



A University of Sussex PhD thesis

Available online via Sussex Research Online:

<http://sro.sussex.ac.uk/>

This thesis is protected by copyright which belongs to the author.

This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the Author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the Author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

Please visit Sussex Research Online for more information and further details



CHARACTERISING SEMANTICALLY COHERENT CLASSES OF TEXT
THROUGH FEATURE DISCOVERY

ANDREW DAVID ROBERTSON

Submitted for the degree of Doctor of Philosophy
Department of Informatics
University of Sussex
May 2018

SUPERVISOR:
David Weir

DECLARATION

I hereby declare that this thesis has not been and will not be submitted in whole or in part to another University for the award of any other degree.

Signature:

Andrew David Robertson

ABSTRACT

There is a growing need to provide support for social scientists and humanities scholars to gather and “engage” with very large datasets of free text, to perform very bespoke analyses. `METHOD52` is a text analysis platform built for this purpose (Wibberley et al., 2014), and forms a foundation that this thesis builds upon.

A central part of `METHOD52` and its methodologies is a classifier training component based on `DUALIST` (Settles, 2011), and the general process of data engagement with `METHOD52` is determined to constitute a continuous cycle of characterising semantically coherent sub-collections, *classes*, of the text.

Two broad methodologies exist for supporting this type of engagement process: (1) a top-down approach wherein concepts and their relationships are explicitly modelled for reasoning, and (2) a more surface-level, bottom-up approach, which entails the use of key terms (surface features) to characterise data. Following the second of these approaches, this thesis examines ways of better supporting this type of data engagement to more effectively support the needs of social scientists and humanities scholars in engaging with text data.

The classifier component provides an active learning training environment emphasising the labelling of individual features. However, it can be difficult to interpret and incorporate prior knowledge of features. The process of feature discovery based on the current classifier model does not always produce useful results. And understanding the data well enough to produce successful classifiers is time-consuming. A new method for discovering features in a corpus is introduced, and feature discovery methods are explored to resolve these issues.

When collecting social media data, documents are often obtained by querying an `API` with a set of key phrases. Therefore, the set of possible classes characterising the data is defined by these basic surface features. It is difficult to know exactly which terms must be searched for, and the usefulness of terms can change over time as new discussions and vocabulary emerge. Building on the feature discovery techniques, a framework is presented in this thesis for streaming data with an automatically adapting query to deal with these issues.

ACKNOWLEDGEMENTS

I am incredibly grateful for the support and guidance of my supervisor David Weir and Jeremy Reffin, both of whom I admire and have taught me a great deal; their advice was invaluable throughout my research.

This work was funded by the Text Analytics Group (TAG) at Sussex University. I thank TAG for providing me with this opportunity and an excellent place to work. I am lucky to have worked with such friendly, helpful, and clever people as the members of TAG. Thanks especially to Simon Wibberley, Jack Pay, Thomas Kober, Julie Weeds, Miroslav Batchkarov, and Matti Lyra for years of collaboration and discussion.

I would like to thank the users of the METHOD52 platform, whose experiences were vital to this thesis. I particularly appreciate the DEMOS side of the Centre for the Analysis of Social Media (CASM) for not only being users but teachers of new users. In particular, Josh Smith who diligently tested new features and collected user feedback for this research.

I appreciate immensely my parents Kim & David Robertson, sister Mandy, and fiancée Mary Browning for their years of support and confidence in everything I do. Writing this thesis was made significantly easier and more pleasant through Mary's encouragement and support each day.

CONTENTS

1	INTRODUCTION	1
1.1	Important Features of a Corpus	5
1.2	Method52	7
1.2.1	Component Contributions	12
1.3	Method52 & Similar Tools	13
1.4	Bespoke Classification & Active Learning	14
1.5	Feature Discovery with Active Learning	18
1.6	Adaptive Streaming	20
2	DATA ENGAGEMENT AS CLASS DISCOVERY	22
2.1	Engagement in Two Mindsets	24
2.2	Engaging with Textual Datasets: An Example	26
2.3	The Search & Explore Cycle	30
2.4	Conclusions	33
3	SURPRISINGLY FREQUENT FEATURES	36
3.1	Related Work	38
3.2	Evaluation Strategy	44
3.3	Measure of Surprisingness	49
3.4	Extending to Phrases	52
3.4.1	Counting Phrases	54
3.4.2	Pruning Phrases	57
3.4.3	Ranking Phrases	58
3.5	Tweet Analysis	61
3.6	Single Large Document Analysis	63
3.7	Establishing the Expectation	66
3.8	Using Existing Method52 Pipeline Architecture	69
3.9	SFPD Comparison with Chi-square and Log-likelihood Ratio	74
3.10	Corpus Homogeneity	81
3.11	Chapter Summary	82
4	FEATURE DISCOVERY WITH ACTIVE LEARNING	84
4.1	Classification in Method52	86
4.1.1	Analysis by Classifier	86
4.1.2	Classifier Training Procedure	91
4.1.3	Classifier Model	95
4.1.4	Active Learning Queries	97
4.1.5	Methodologies for Applying Classifiers to Twitter Corpora	98
4.2	Dataset: Brussels Bombings	99
4.3	Dataset: Father’s Day	110

4.4	Exploration Using Surprisingly Frequent Features . . .	112
4.4.1	SFPD for Corpus Insights	114
4.4.2	SFPD for Classification Insights	119
4.4.3	Domain Adaptation	124
4.4.4	Irrelevance Filtering	127
4.4.5	Vocabulary Overlap	128
4.4.6	Using Discovered Phrases in the Feature Model	133
4.5	Original Feature Contexts	133
4.6	Encoding Context with Syntactic Ngrams	136
4.6.1	Related Work	138
4.6.2	Approach	143
4.6.3	Adapting Dependency Parsing to Twitter	145
4.6.4	Impact on classifier training process	150
4.7	Weighting prior knowledge	157
4.7.1	Frequency-based Weighting	159
4.7.2	Indicativeness Weighting	163
4.8	Chapter Summary	164
5	ADAPTIVE TWITTER STREAMING	166
5.1	Related Work	172
5.2	Dataset: Disrupting Daesh	182
5.3	Adaptive Streaming with SFPD	183
5.4	Topic Structure	190
5.5	Near-duplicate Tweets	194
5.6	Maintaining Precision Through Relevance Assessment	196
5.6.1	Blacklisting Irrelevant Terms	198
5.6.2	Qualifying Relevant Terms & Data Collection .	199
5.6.3	Classifier Scope Problem	200
5.6.4	Classifier Drift Problem	201
5.7	Selective Target & Relevance	202
5.8	Query Term Expiration	204
5.9	Domain Adaptation	206
5.9.1	Continuous Sample Integration	208
5.9.2	Noise Reduction with a priori Topic Knowledge	209
5.9.3	Topic Specificity	210
5.10	Evaluation on Brexit	210
5.11	Chapter Summary	222
6	CONCLUSIONS	225
6.1	Future Work	228
6.2	Limitations	233
	ACRONYMS	234
	BIBLIOGRAPHY	235

INTRODUCTION

Social scientists and humanities scholars increasingly wish to gather and “engage” with large text datasets, to perform highly bespoke analyses. The size of these datasets makes them challenging to study; analysts encounter difficulty finding and isolating the data they are interested in. This necessitates the support of automated tools.

The general objective when analysing these large datasets of unstructured text is to infer some structure that permits downstream (later in the analysis) quantitative analysis of the previously unstructured data. In order to attain this goal, the analyst must first gather the appropriate data, then “code up” the documents according to some structure. In order to demonstrate the kinds of insights which are very specific to the individual researcher and dataset, the analysis must be bespoke, and generally once complete cannot be directly re-applied to another scenario. This runs counter to what is often the expectation in [NLP](#) research, wherein a particular task (such as sentiment analysis) is defined, and the techniques for approaching the task are iterated on over time, but applicable to the same task, over and over again. This thesis is slightly unusual, in that it is not a typical type of this kind of [NLP](#) research, but more a study of how to better support the above type of bespoke analysis. Nor is it a social science thesis, since it still focuses on the *technology* for supporting scholars.

In these bespoke analyses, the nature of the data gathering stage depends on the type of study being undertaken. The researcher may wish to analyse free text responses from questionnaires, or records or other data from a public body such as a police department or health organisation. In these cases, the gathering of data is usually taken care of by the organisation in question. The analyst may alternatively be interested in some section of a historical dataset, such as the transcripts of Old Bailey trials. Here, in order to gather data, the analyst will likely need to determine which documents are actually relevant. This may first be a filter on metadata, such as the date range in which a trial occurs, or the role of the speaker. But the documents may be further filtered by whether they contain certain key phrases, or whether classified relevant by a classifier trained to model the notion of relevance required for the study. And the type of study of

most interest in this thesis is the analysis of social media datasets, for example, the discovery and study of online hate speech, or user concerns regarding crime, or politicians and their policies. These studies require the specification of a query composed of words and phrases, which is presented to the [API](#) of a social media platform in order to collect user messages relevant to that query. The query is rarely sufficient to ensure that all returned documents are relevant, so these documents also must undergo similar relevance filtering.

The relevance filtering could be considered the first part of the “coding up” stage, in which the analyst explores the data and attempts to sort and logically categorise the documents in order to gain insight from the text. The first step in most cases is to code up which documents are relevant to the study.

The coding then proceeds with these new categorisations of documents. For the online hate speech analysis example, subsequent categories of document could be distinguishing actual hateful language versus discussion about hate speech. The goal of the analyst may be to further analyse hate speech discussion or the hate speech itself. If the latter, then further subcategories could define different types of hate speech, perhaps according to the kinds of hateful terms employed, or who it is targeted at, or what it was in response to. An example with mental health support forum data would be finding comorbidities in discussions of mental health issues, or trying to isolate actual cries for help, or treatment discussions. In police crime records, in free-text officer reports, it could be desirable to identify patterns of crime that are not yet catered for in any structured text inputs.

In order to complete studies of the above nature, there is a requirement for tools which permit bespoke analysis. Coding the discovered specific subcategories of hate speech, or the levels of inebriation of defendants at the time of their alleged crime as reported in Old Bailey trials are studies unlikely to be amenable to existing pre-baked solutions. The methods of filtering, splitting, and annotating the data must therefore be flexible, and tailor-able to individual scenarios. The tools must also be able to import a variety of data formats, and scrape data from social media platforms in an ongoing real-time stream. Furthermore, the tools need to be agile and fail-fast in order to support a flexible and iterative approach. The researcher will frequently be exploring the data with initial hypotheses, but ultimately expect to explore and draw insight from the text. Some analysis could lead to

uninteresting, inclusive, or erroneous results, and in this situation, the least effort should be wasted.

METHOD52 (version 2 of METHOD51: [Wibberley et al., 2013, 2014](#)) is a general-purpose tool for collecting and analysing text, which meets the requirements laid out above. Section 1.2 gives an introductory high-level overview of the tool. The bespoke analysis and agility is primarily achieved in METHOD52 through a modular interface, in which the user builds custom “pipelines” of analysis out of components of functionality which collect data, then filter and annotate it according to the analyst’s needs. The most important component for making bespoke categorisations of the data is the classifier, which began as an extension to DUALIST ([Settles, 2011](#)), an active learning environment for training a Naïve Bayes classifier.

A key innovation of DUALIST was the inclusion of a system for presenting individual features to the user, which the user can label alongside their labelled documents. The classifier then incorporates both document and feature labels into its model. [Settles \(2011\)](#) set out to build a system designed to complement the strengths of both learner and annotator. Annotators are shown to be able to label features much faster than documents, and using feature annotations can rapidly build well-performing classifiers.

Despite METHOD52’s suitability, when training new analysts to use METHOD52, and during informal discussions about their usage of METHOD52, analysts report certain difficulties and limitations, which can be categorised into two main areas. Firstly, when training bespoke classifiers (introduced in more detail in Section 1.4), users experience difficulty utilising the feature labelling functionality. It can be difficult to interpret the features and find useful ways to label them. The process of feature discovery based on the current classifier model does not always produce useful results. And understanding the data well enough during exploration to produce successful classifiers (classification tasks that the data supports) is time-consuming. The second area concerns the data collection strategy. The data collection strategy is important because it determines what relevant data is collected, and therefore what information can actually be derived from the text. Social media research is often concerned with data collected in real-time as messages are published online. However, it is rarely possible to know exactly what terms must be searched for in order to acquire all the relevant data, and if it is possible to know, over time what’s needed can change as new vocabulary and discussions emerge. There-

fore, the analyst can lose out on relevant data, and only discover this when the data analysis is complete, if at all.

The aim of this thesis is to explore solutions to these problems and limitations, and iterate on METHOD52's approach. For context, and as part of the methodology for determining how best to derive these solutions, and to clarify how analysts actually engage with data using METHOD52, Chapter 2 first defines the process of using METHOD52 to engage with text data. The general engagement process is determined to constitute a cycle of characterising semantically coherent sub-collections, *classes*, of the text: a continuous loop of discovering and defining classes of document, then isolating them for further analysis. The term "class" is used here to refer to semantically coherent categories of text: if documents can be categorised according to some property (or properties) of their text's meaning, then they can be said to be part of a class defined by that property (or properties). Two broad methodologies exist for supporting this type of engagement process: (1) a top-down approach wherein concepts and their relationships are explicitly modelled for reasoning, and (2) a more surface-level, bottom-up approach, which entails the use of key terms (surface features) to characterise data. This thesis follows the second of the two approaches. A bottom up approach is adopted for supporting class discovery and isolation by producing techniques to better and more explicitly exploit individual surface features like the words and phrases of a text.

With a particular focus on social media data analysis, this thesis therefore attempts to answer: *how can feature discovery be used to support the characterisation of semantically coherent classes of text?*

This research question is divided into three sub-questions:

1. How can features be identified which provide useful bases for characterising classes of text? (Chapter 3)
2. How can feature discovery support the identification, definition, and characterisation of classes of text? (Chapter 4)
3. How can feature discovery support the collection of data relevant to classes of text? (Chapter 5)

After Chapter 2's discussion of the explore-search cycle, Chapter 3 details the contribution of a feature discovery method for Question 1. Chapter 4 addresses Question 2 by introducing feature discovery and incorporation strategies for discovery and isolation of classes of

documents (supported by an active learning classifier training environment). Chapter 5 deals with Question 3 by the contribution of a framework for streaming real-time data from an API using a query whose terms are adapted based on the data collected, using feature discovery strategies.

These contributions are united in the theme of using feature discovery strategies to better define, discover, and isolate classes in the data in a bottom-up approach.

Despite a common theme of feature discovery, these contributions draw on sufficiently different areas of research that it was deemed more logical to split the discussion of related work into separate sections: 3.1, 4.6.1, & 5.1.

Chapter 6 ends with final conclusions.

Given that METHOD52 is in active use for consulting and collaborations by its creators at CASM CONSULTING and TAG¹, the technology created in this research is incorporated into METHOD52, so that it can be used by current and future METHOD52 analysts.

The remaining sections of the introduction each give overviews of particularly important details supporting this thesis.

1.1 IMPORTANT FEATURES OF A CORPUS

In this thesis, and generally in NLP, “features” are properties extracted from documents, about which machine learning algorithms like the Naïve Bayes classifier collect statistics in order to make predictions. The most simple and common feature type to be extracted is the unigram, which is simply an individual token in the text.

The main theme of the techniques in this thesis is the notion that some features are more important than others. The following are examples of why certain features become more important than others in a document collection:

1. After having collected a dataset that is yet to be analysed, without any notion of the topics or sentiment present in the data, certain words/phrases from the text can be indicative of the document classification tasks which are feasible to perform on the data.
2. Machine learning classifiers like the Naïve Bayes Classifier (NBC) must more highly weight those features that are most indicative

¹ The Text Analytics Group at the University of Sussex <http://taglaboratory.org>

of the classifications it is learning in order to perform prediction effectively.

3. In the `METHOD52` active learning environment, features are presented to the user with the aim of finding those features which would best inform the model about classification decisions according to the user's prior knowledge.
4. After having collected a dataset via query to some data source (e.g. the `TWITTER API`), certain words/phrases could constitute a new query to the data source which would permit the collection of additional similar or related data.

Points 1, 2, & 3 exemplify document features being used for better understanding of a dataset through explore/search, and 4 shows their use for ensuring collection of additional relevant data, ensuring that the desired classes can be defined.

How do we decide which features are important? That is partially driven by what we want to gain from the data. There are usually many types of analysis possible given the same data, so our own research goals can drive which aspects of the data are most important. However, the possible analyses are dependent on inherent attributes of the data: a corpus created by querying `TWITTER` for tweets containing "general election" will no doubt contain, in large quantities, discussions about party leaders and election issues. These issues are what make this dataset distinctive from just a broad sample of English text: these features occur more often than one would expect from a broad sample of English.

A feature, like a unigram or bigram, occurs *surprisingly* frequently if it occurs more than one would expect statistically. Inherent in the idea of an expectation is some *reference* point. If we want to determine the features that occur surprisingly frequently in a dataset, we need a reference corpus from which we derive the expectation that was surprised.

The reference corpus can be as simple as a broad sample of English, such as a large collection of English `WIKIPEDIA` articles. Or the reference corpus can be chosen creatively to expose particular traits of the target corpus. For example, the reference corpus could be tweets obtained during January for a given query, and the target could be February tweets, with the intention that only features occurring proportionally more often in February than in January are interesting.

January documents would therefore establish the expectation of a feature's rate of occurrence.

The approach used for finding these features, inspired by keyword extraction research and this notion of surprise, is a contribution discussed in Chapter 3, after the discussion in Chapter 2 which first shows how “engagement” with a social media dataset in METHOD52 becomes a cycle of class discovery and isolation.

Sections 1.5 & 1.6 outline the contributions of Chapters 4 and 5, which both make use of the surprisingness of features.

1.2 METHOD52

METHOD52 is the second version of METHOD51 (Wibberley et al., 2013, 2014). It is a general-purpose tool for collecting and analysing text, built and maintained by CASM CONSULTING. METHOD52 improved upon METHOD51 by abstracting its data schema away from a single data source (TWITTER), then adding methods for uploading user data (CSV, PDF, WORD, MSG), and other data collection capabilities (REDDIT API, web searching & scraping). The contributions of this thesis build upon METHOD52.

Figure 1.1 is a screenshot of METHOD52's User Interface (UI). METHOD52 encourages potentially non-technical analysts to engage with their data and data processing strategy to produce highly bespoke analyses.

METHOD52 users analyse text by constructing programs visually, dragging and dropping boxes that represent components of functionality, and connecting them with arrows in order to set the path that data takes through the components.

METHOD52 provides “components” of functionality such as:

- Read from and write to a database for long-term storage of text and analysis.
- Import text, or collect data from services such as TWITTER and REDDIT.
- Train a Naïve Bayes classifier to classify text with user-defined classifications in an active learning loop, which leverages large quantities of unlabelled data.
- Geolocation of tweets and users.
- Volume over time analysis.

- Network influence analysis.

Components may generate output which can be annotated onto documents as they pass through the components. Filters can be defined over the annotations on documents, so that the user can control which components are passed specific kinds of documents.

The user builds “jobs”, each comprised of a pipeline of components connected with arrows through which documents are passed to produce some form of analysis.

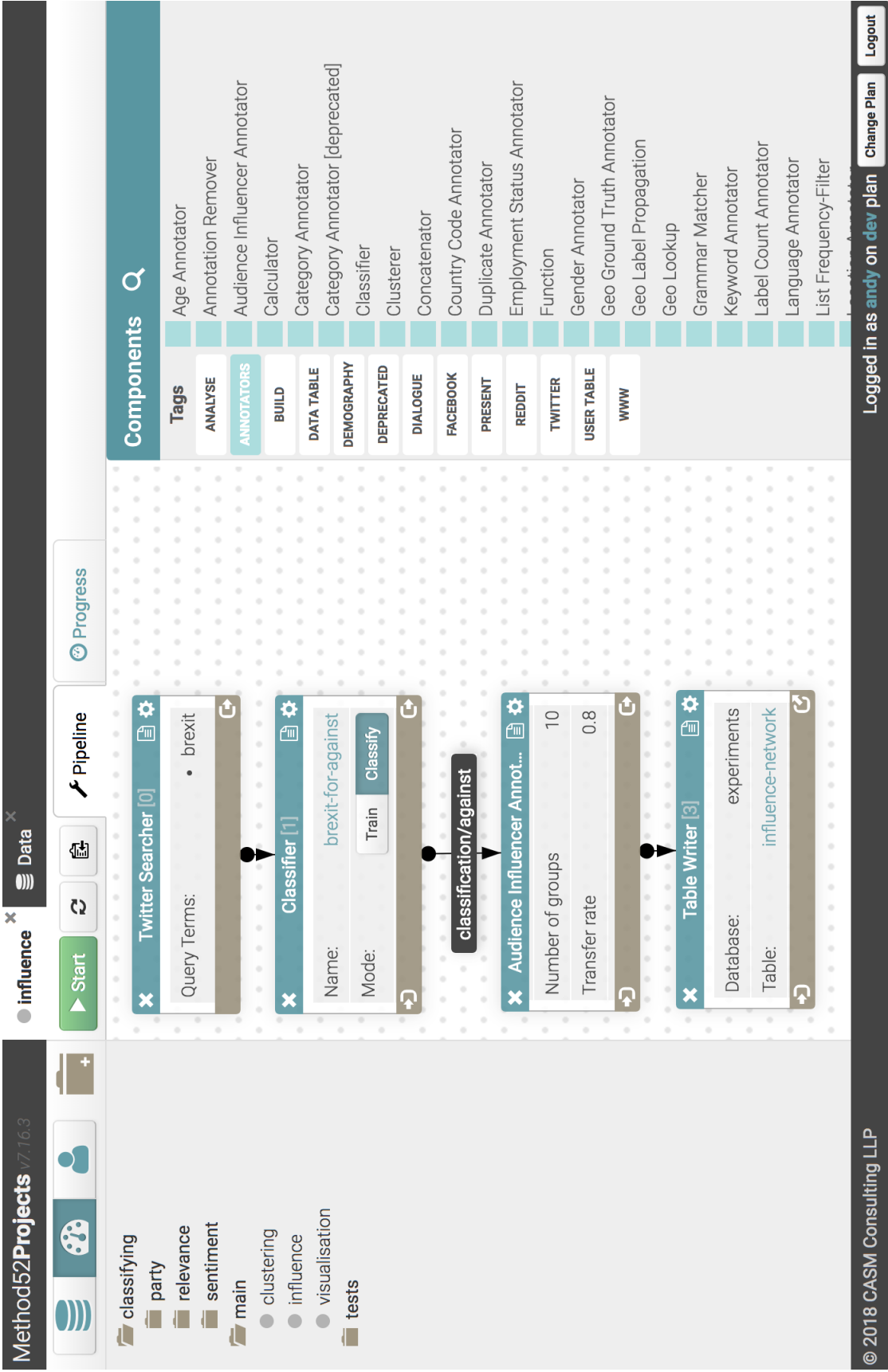


Figure 1.1: Screenshot of the METHOD52 UI. Shows a job tab with its pipeline of components. An influence network is inferred and recorded from historical tweets mentioning “brexit” that are classified as being against Brexit.

Figure 1.1 shows an open job tab, where the functionality of an example job is defined. A `TWITTER SEARCHER` component queries the `TWITTER API` for historical tweets mentioning the term “brexit”. The resulting tweets are categorised as for/against Brexit by a user-trained `CLASSIFIER` component, which like other annotation components, annotates a new field to the document representing its output. The for/against annotation is used as the basis for filtering along an arrow such that only tweets classified as *against* Brexit are passed to the next component. The `AUDIENCE INFLUENCER` component infers an influence network based on the dynamic interactions of users (as opposed to their static follower/friend network) using the `AUDCLUS` method (Lin et al., 2015). The resulting influence network over anti-Brexit users is recorded in a database table by the `TABLE WRITER`.

Under a different job setup, the user could train the `CLASSIFIER` in an active learning loop with bespoke classifications by providing hand-annotated training data.

Also shown in Figure 1.1 is the project explorer pane on the left side, which allows the organisation of jobs into project folder hierarchies. In the job tab, components can be selected from the component list on the right. A job is run by pressing the green “start” button, and its progress can be monitored using the “Progress” tab.

network/clusterBelongingness	network/clusterInfluence	twitter.user/id
3.64287571045057e-9	2.1462697091352243e-8	8405325321
0.9999999887540073	2.101609383601913e-8	
2.8956783171722074e-9	2.2493947881068527e-8	
0.00038040983662387723	0.00022147430055759662	10135337202071920641
0.00021645291993406193	4.6858860271890057e-8	
0.0002917022244657989	5.73825773575995e-8	
0.00018347504818947007	4.921563941314518e-8	21198790611
0.00018511577278254536	4.3916120636535935e-8	
0.0002913629444558336	0.00022702508012405578	
6.025842062755526e-8	0.0010088876431829593	32344296903437
1.2682560569096951e-33	0.00009840416068217446	
0.9999998045523446	0.00010096025251757114	
0.00035199735697964387	0.0002214717003135004	6025808832794624
0.00021670745818685395	4.6656472714415695e-8	
0.0003427372586423433	5.477672607573074e-8	

Figure 1.2: Screenshot of the METHOD52 database table viewer. When the job in Figure 1.1 has completed, and the user clicks on the output table called “influence-network”, they are presented with the table viewer tab. Each row in the table is a TWITTER user with annotations produced by the Audience Influencer component. These annotations give measures of how much each user belongs to each cluster (“audience”) and how much each user influences each cluster. This data could then be used as input to another METHOD52 job, or exported to CSV for use in other software.

METHOD52 is an adaptable platform. Its ability to generalise to new domains and types of analysis has been tested in many projects. Example studies include:

- Analysing the presence and nature of racial and ethnic slurs on social media (Bartlett et al., 2014).
- Discovering the supporters of Islamic State (IS) through analysis of their social media strategies, and monitoring their disruption by TWITTER (Conway et al., 2017).
- Analysing attitudes on TWITTER after extremist attacks, and after Brexit, studying the presence of Islamophobic, racist, xenophobic, or hateful views (Miller et al., 2016).

- Analysing the nature of political discussions online (Krasodonski-Jones, 2016; Smith, 2017).

Many projects that utilise METHOD52 are concerned with text from TWITTER², or similar short comments from other social media platforms. This fits with the focus of this thesis.

Standard public access to TWITTER'S API is assumed, so the techniques explored are tailored to this level of access, not to the full TWITTER fire-hose³, the use of which would be impractical for many researchers in any case.

1.2.1 Component Contributions

The technological contributions of this thesis are now built into components of METHOD52. Components that were added or modified for the contributions of this thesis are summarised below.

Surprising Phrase Detector This is a new component that implements the feature discovery method that this thesis contributes in Chapter 3.

Duplicate Annotator When computing word frequency statistics for feature discovery, it became necessary to be able to filter out documents with near-duplicate texts. Therefore, this component was added, which implements a well-established near-duplicate detection algorithm (see Section 5.5).

Classifier This component was an existing and core component of METHOD52, which allows the analyst to train and deploy bespoke classifiers. The component features an active learning loop with its own method of feature discovery. Chapter 4's contributions are modifications to how it discovers and uses features, and methods for combining its use with the Surprising Phrase Detector. Section 4.1 describes the state of this component before the modifications introduced by this thesis.

Surprising phrase tracker Chapter 5 introduces a framework for adaptively querying TWITTER in real time, based on features discovered in the collected data. The framework depends on the

² A social media platform on which users publicly share short messages ("tweets"), that can optionally contain references to topics or other users (marked by a # or @ tag respectively) <http://twitter.com>

³ The TWITTER "fire-hose" is the complete, unfiltered stream of all tweets being published, which is not available for free through the public API

interactions of several METHOD52 components (including the Classifier), but this new component is at the core, which tracks the query and modifications to it, and wraps the functionality of the Surprising Phrase detector.

Twitter Steamer This existing component handles contacting TWITTER with a keyword query and collecting the ongoing stream of matching tweets. This thesis modifies it such that it is no longer fixed to its initial query, but instead can receive new query instructions while running, and modify the TWITTER query, with only optional user interaction.

1.3 METHOD52 & SIMILAR TOOLS

There are many tools that can be used to analyse social media text. METHOD52 aims to provide an environment which balances the need for computational capabilities with the requirement that the system can be used by analysts without programming skills or a background in NLP. Furthermore, it encourages analysts to directly observe the data, tailor a bespoke technique to match the data, and consider the feasibility of the type of analysis that they are performing (Wibberley et al., 2013).

There exist tools with richer computational capability (e.g. more choice of algorithm for a given task) than METHOD52, such as GATE⁴, RAPIDMINER⁵, and STANFORD NLP⁶. However, in exposing implementation details like tokenisation, feature weighting, and algorithm choice as the units of the analysis, researchers with an NLP background are more conceptually aligned with these tools than, for example, social scientists. Components in METHOD52 are more likely to be presented as functional sub-analyses of a problem, rather than their literal computational task or implementation.

On the other end of the scale, tools such as NVIVO⁷, are very much aligned with concepts non-technical analysts are familiar with in qualitative research, but in terms of computational capabilities much beyond annotation, filtering, and sorting they are lacking.

⁴ <https://gate.ac.uk>

⁵ <https://rapidminer.com>

⁶ <https://nlp.stanford.edu/software>

⁷ <https://qsrinternational.com/nvivo>

Tools such as `DISCOVERTEXT`⁸ and `LEIPZIG CORPUS MINER`⁹ offer a range of computational capabilities similar to `METHOD52` (such as active learning classifier training), which are also more conceptually aligned with the type of analyses a social scientist might perform. However, a strength of `METHOD52` is its visual representation of the analysis being performed as pipelines of components. Analysts can see how the data arrived at its current state (`RAPIDMINER` also features a pipeline view).

In order to tailor analyses to a particular problem, `METHOD52` analyses usually include a bespoke classification component, which requires the analyst to manually label some amount of their data in order to train a classifier to fit their particular problem. In this way, analysts are directed to read the texts and produce analyses that are tailored to their data, rather than generic solutions which may or not be producing useful distinctions in the data. This contrasts with tools such as `LIWC`¹⁰, which compares text to fixed sets of words that are intended to indicate concepts such as emotions or thinking styles. Another similar example is `OPINIONFINDER`¹¹.

The active learning environment provided by `METHOD52` for training the classifier component is based on a tool called `DUALIST` (Settles, 2011) and is introduced in the next section.

1.4 BESPOKE CLASSIFICATION & ACTIVE LEARNING

This section introduces `METHOD52`'s classifier and active learning frameworks, since these play a key part in the isolation of classes in `METHOD52`, and are used extensively in this work. Some of the contributions of Chapter 4 make additions to the classifier component.

The classifier component in `METHOD52` is a Naïve Bayes Classifier (`NBC`), which is a simple probabilistic machine learning method. Due to its computational simplicity, it can be used interactively in real-time to repeatedly train on a growing set of hand-labelled data. And despite its simplicity, it has been shown empirically that the `NBC` can achieve good performance despite its naïve assumptions (Settles, 2011).

These factors were utilised by the `DUALIST` framework (Settles, 2011), a platform in which the user is tasked with labelling both

⁸ <https://discovertext.com>

⁹ <http://lcm.informatik.uni-leipzig.de>

¹⁰ <https://liwc.wpengine.com/>

¹¹ <https://mpqa.cs.pitt.edu/opinionfinder>

features and documents in order to rapidly train an NBC that performs well on the evaluation data. The classifier presents to the user those documents whose categorisation the classifier is most uncertain about, and those features whose presence and/or absence in a document correlates most with reduced classifier uncertainty. The classifier uses the user-labelled features in an expectation-maximisation step to incorporate unlabelled data.

METHOD52's classifier training procedure is based on the DUALIST system, implementing this processing of feature and document querying and labelling (shown in Figure 1.3). Technical details of METHOD52's classifier component before additions made by this thesis, and details of exactly how the component differs from DUALIST can be found in Section 4.1.

The screenshot displays the METHOD52 classifier training interface. At the top, a list of tweets is shown with three label buttons: 'attack' (pink), 'sympathy' (yellow), and 'other' (grey). The 'sympathy' button is selected for most tweets. Below the tweets is a 'Submit Labels' button. At the bottom, a table shows the extracted features for each category.

Attack	Sympathy	Other
on	just_get	do_you
to	affairs_exec	do
referendum	with_@ronamacleod	be_a
danish	prayers_are	hitchens_do
demand	affected_by	a_servant
danish_politicians	sad_about	servant
politicians_demand	the_families	servant_of
brussels_attacks	thoughts	you
airport	prayers_go	you_want
attacks	my_heart	want_to
dead	prayers	to_be
brussels_airport	sending_love	of_brussels
eXPlosions	praying_for	peter
injured	our_thoughts	peter_hitchens

Figure 1.3: Screenshot of the labelling portion of the METHOD52 classifier training screen. Both tweets and features (here unigrams and bi-grams) are presented to the user for labelling according to the user-defined scheme of “Attack”, “Sympathy”, or “Other”.

It is unreasonable to assume that a classifier trained and evaluated on one dataset (source) will perform equally well on a different dataset (target), because the vocabulary of features in the new dataset will present different statistics for their level of indicativeness of the classifications. The more dissimilar the data, the worse the mismatch, since there is more likely to be a greater vocabulary mismatch (Ben-David et al., 2010). This problem gives rise to methods of *domain adaptation*, such as finding and utilising those features that are used similarly across both source and target data in order to find a projection of

the source feature space that is more suitable for the target data (Li, 2012).

Instead of relying on domain adaptation, METHOD52 takes the approach of providing tools for the user to build *bespoke* pipelines of functionality, including user-trained classifiers (chained or simultaneous) in order to make the best use of the data available. This flexibility enables the user to adopt strategies to which their problem and data are most amenable by building different formations of classifiers. Some of these strategies are discussed by Wibberley et al. (2014) and described in Section 4.1.5.

The METHOD52 classifier training environment provides the means to annotate an evaluation set, which is then used to keep a current evaluation of the classifier’s performance. For each classification label, the user is shown the model’s precision, recall, and f-score. The overall classification accuracy is also given. The performance is updated whenever the user submits new training data.

Actual	Predicted	attack	sympathy	other
attack		112	0	0
sympathy		12	49	0
other		23	1	3

Document Category Layout

Advanced EM Settings

Label		Precision	Recall	F-Score	Accuracy	Coded	Prior Multiplier
attack	<input type="button" value="Sample"/>	0.762	1.000	0.865		32	<input type="text" value="1"/>
sympathy	<input type="button" value="Sample"/>	0.980	0.803	0.883		10	<input type="text" value="1"/>
other	<input type="button" value="Sample"/>	1.000	0.111	0.200		11	<input type="text" value="1"/>
Unlabelled		5114	Features	18	0.820		sent out:10

Figure 1.4: Screenshot of the METHOD52 classifier evaluation portion of the training screen. The upper portion of the table shows how many are classified correctly or incorrectly for each classification, and the lower portion gives details of precision, recall, f-score, accuracy. From here, the user can also click the sample button to take a random sample of documents which the classifier determines belongs to the selected classification.

1.5 FEATURE DISCOVERY WITH ACTIVE LEARNING

This section outlines the contributions of Chapter 4 toward the following research question mentioned early in the introduction:

How can feature discovery support the identification, definition, and characterisation of classes of text?

A core part of data exploration in `METHOD52` is the classifier component. It allows the user to inspect random samples of documents, to devise a categorisation scheme (explore/define classes) and to train a classifier through active learning in order to apply the categorisations to large amounts of data (isolate classes).

Due to the way that documents are presented to the user during active learning, the user will encounter text that challenges the boundaries of the classifications, necessitating clear class semantics in order to obtain a well-performing classifier. Typically, users will need to re-define their classes several times as they become better acquainted with the data. Clearly, if there were some way to sooner gain an understanding of the data, the number of iterations of class definition, testing, and revision could be reduced, thereby speeding along the process of classifier construction and further data understanding.

Chapter 4 elaborates on this issue and approaches it from a feature discovery perspective. In particular, Section 4.4 introduces feature discovery strategies that speed corpus understanding by providing a rapid overview of the data, allowing more efficient discovery of classes (definition of class semantics) within the data.

Chapter 4 also addresses the problems with the current feature labelling functionality. Class isolation is improved by dealing with weaknesses in the feature discovery and incorporation mechanisms within the active learning system.

Features f labelled as indicative of classifications c are used to build an initial classifier by instructing the classifier to assume it has seen α occurrences of f in documents classified with c . This initial classifier is used to calculate the class probabilities for each of the unlabelled documents, which together with the hand-labelled document classifications make a new training set upon which the final classifier is trained (full details in Section 4.1.2).

The assignment of a fixed α value for each feature is in part to blame for some user confusion concerning performance drops when labelling infrequent features, and many features with varying levels

of indicativeness for their classifications. This problem is examined in Section 4.7, which introduces methods for assigning dynamic α values.

Furthermore, it is difficult even understanding or selecting the features that are presented through the active learning procedure. Terms presented in isolation often simply lack the context to be interpreted. One cannot always know the particular sense of the term that is being used in the text, and even it is somehow obvious, it is not always clear how it could be indicative of a particular classification. This *annotator awareness* issue is primarily covered in Section 4.5, using simple user-interface features.

Even if the meaning and indicativeness is clear, it is also frequently the case that a term could be indicative of multiple categories depending on context. This *conditional indicativeness* problem is examined in Section 4.6, and approached by extracting syntactic dependency ngrams that encode context more logically in order to increase the likelihood that a feature can be found that is suitably indicative of a classification, and that does not present conditional indicativeness.

Even if the features presented to the user and their method of integration into the model is improved, there is still the problem that the classifier's method of feature proposal is locked into the analyst's current notion of the classification boundaries. It is only as powerful as the current classification definition (class semantics). Yet, we have already established that this definition undergoes frequent revision as the analyst becomes familiarised with the data. Whilst the work described above streamlines each step of this process, we must also focus on reducing the number of these iterations.

Section 4.4 uses surprisingly frequent phrase analysis to add a more classification-definition-agnostic process to exploring terms in the unlabelled data, thereby helping the analyst to understand the data and reduce the likelihood of committing to a flawed classification definition, without having to read hundreds of randomly sampled documents. This therefore reduces the number of steps in the revision process. The phrase analysis is also used to examine documents under the current classification definitions in a more engaging manner than simply checking the current category f-scores.

1.6 ADAPTIVE STREAMING

This section describes the Chapter 5 contributions for the aforementioned research question shown again below:

How can feature discovery support the collection of data relevant to classes of text?

The data collection strategy is fundamentally important to classifying data, because it determines the space of possible class semantics that can be derived from the text.

A currently popular source of study is social media. Social media research is often focused on real-time messages, or at least messages collected during a phase of the study. Furthermore, the most popular social media site for study is TWITTER. Therefore, this thesis focuses mostly on TWITTER as an example. However, the proposed adaptive streaming framework is applicable to similar scenarios where keywords and phrases are used to collect data.

TWITTER provides a few APIs for collecting tweets: the TWITTER sample¹², search¹³, and filter¹⁴ APIs. The sample API provides a 1% real-time sample across all tweets being published. The search and filter APIs require the user to specify a query, and only those tweets matching the query are returned. The search API returns historical tweets, and the filter API sets up a real-time stream of tweets matching the query.

These APIs permit users two types of strategy for collecting tweets on a topic of interest. The first possibility is to extract from the 1% sample those tweets which contain terms of interest. Alternatively, the user can provide a set of terms as a query to either the search or filter API. The first option can be useful for certain ways of implementing topic tracking (Magdy and Elsayed, 2014) as described in Section 5.1. However, if the topic of interest is not discussed sufficiently frequently, it may not occur in the 1% sample, and even if it does, data could be lost since the sample is not directed at the topic. The search/-filter approach more directly obtains the tweets of interest.

The problem when finding tweets on a given topic is that the analyst must know the query terms that will produce the data they de-

¹² <https://developer.twitter.com/en/docs/tweets/sample-realtime/api-reference/get-statuses-sample>

¹³ <https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets>

¹⁴ <https://developer.twitter.com/en/docs/tweets/filter-realtime/api-reference/post-statuses-filter>

sire, which are difficult to know ahead of time. Often, after collecting data in this manner and performing their analysis, researchers discover references to topics and subtopics that they would have liked to be included in the data more fully, or they find evidence that the topic of interest drifted away from the vocabulary that was used in their TWITTER query. This is because the vocabulary of a topic drifts over time, and new, related, or sub-topics arise naturally in discussion (Magdy and Elsayed, 2014). Sometimes TWITTER users will happen to reference the relevant query terms together with their sub-topics, but often they will not. The problem grows more serious as the data is collected over longer periods of time, since this allows more time for related events or discussions to arise. This implies that a great deal of relevant data could be lost.

One approach to addressing this problem is to implement an automated process for analysing the text in order to find the terms that would be of interest to researchers, and then immediately incorporate them into the query to the TWITTER API. There are many challenges involved in achieving this, including ensuring that the automated process does not become overwhelmed with irrelevant terms, and how to define relevance to the process in the first place. Chapter 5 analyses these issues and presents a METHOD52 framework for adaptive streaming/querying, the essence of which is based in surprisingly frequent phrase analysis for semi-automatically adapting and updating a keyword query.

This chapter examines what is meant by “engaging” with text data using `METHOD52`. While not a contribution in itself, this chapter provides necessary context for later chapters in understanding how `METHOD52` jobs are built and how this relates to characterising classes of text data.

When users build jobs in `METHOD52`, the visual pipeline of their analysis is saved until they purposefully delete it. So it is possible to see to some extent how a problem was tackled using `METHOD52`. A number of early case studies are also described by [Wibberley et al. \(2014\)](#). This chapter describes in detail an illustrative application of `METHOD52` in section 2.2 which demonstrates the use of `METHOD52` for analysing online conversation. This example is then compared to similar use cases.

The design of `METHOD52` allows us to gain a view of how analysts engage with data using it. Every job built by a user is constructed and saved as a visual pipeline of processes, so it is possible to go back and observe what has happened to the data (for any unclear jobs, informal clarification is sought from their creators). For example, see Figure 2.1 for a job that determines whether `TWITTER` users are expressing gratitude through two layers of classifiers. Informed by inspecting the jobs created by users in existing and past projects using `METHOD52`, this chapter establishes generalisations for how `METHOD52` is used.

Studying social media comments is akin to entering an enormous and busy pub, and then trying to discover, pick apart, and understand the conversations encountered. Social media data is conversational: users’ comments can be seen and responded to by the other users. Unlike datasets such as news articles, the data arises in a social space. It is impossible to be certain who would turn up on the day of our pub visit, and what conversations would arise as we listen. Not only are we uncertain of the topics, but also *how* the topics will be discussed. The words and phrasing used will depend on the individuals and their relationships with each other. We also do not know what happened in people’s lives today which will influence what they discuss and how they discuss it. Events may be large/important enough that they influence multiple lives and lead to wider discussion. People

do not construct an agenda of conversations and carefully edit the points to be raised; they react to the conversations present and the events that sparked them. These conversations represent *classes* of utterance. Where, as mentioned in the introduction, the term “class” is used to refer to semantically coherent categories (or “conversations”) of text. If documents can be categorised according to some property (or properties) of their text’s meaning, then they can be said to be part of a class defined by that property (or properties).

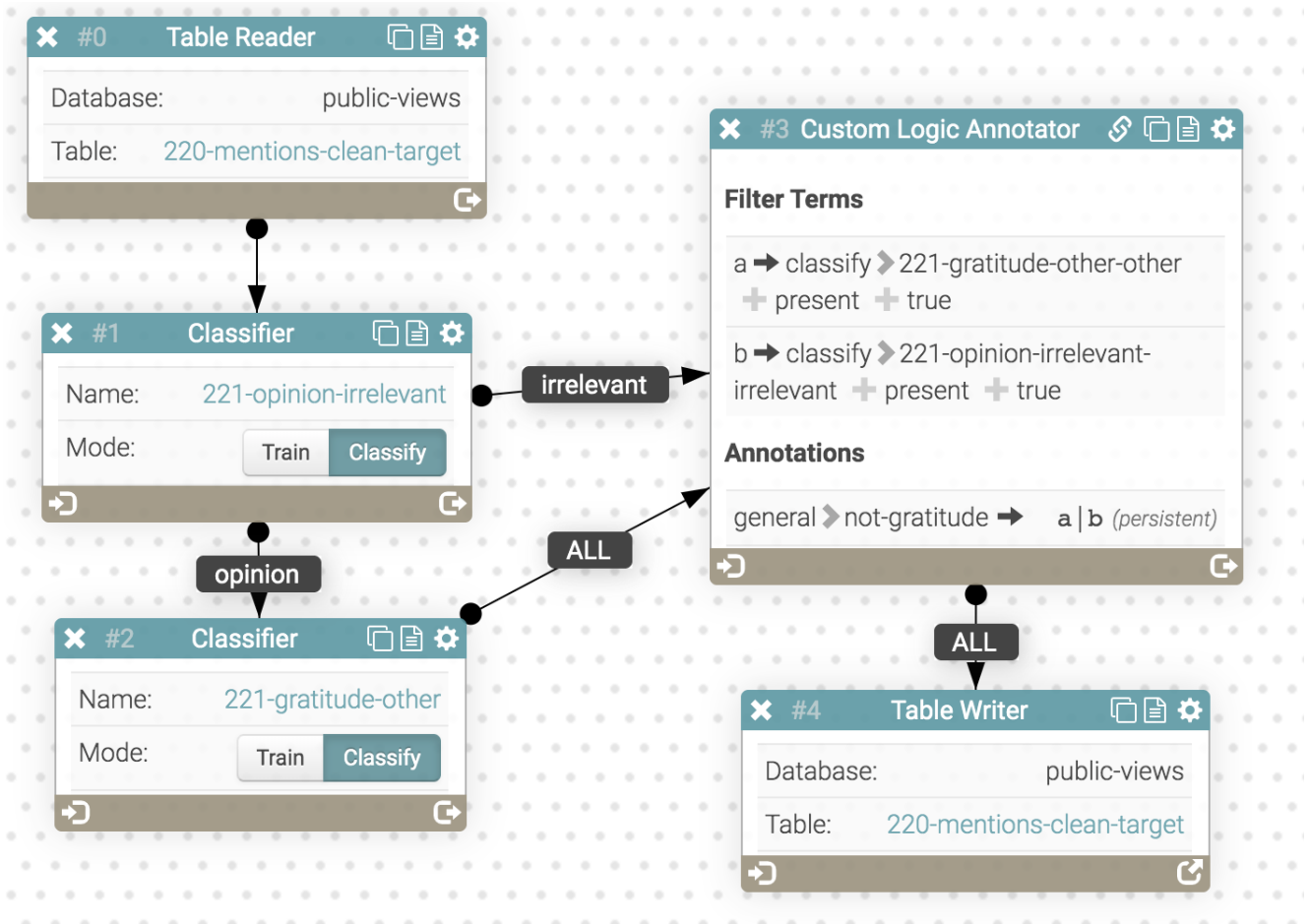


Figure 2.1: Screenshot of a METHODO52 job from a project in collaboration with HMIC. Tweets directed at police force accounts are first classified as whether they contain an opinion or not, then opinion tweets are classified as either expressing gratitude or not. Then all tweets are saved back to the input table with the new classification annotations.

To study these conversations and engage with the datasets they produce, METHODO52 users seem to approach the problem in one of two modes, which will be referred to as “explore” and “search”. In “explore” the analyst attempts to discover information about online con-

versations, to discover and define classes. The difficulty in this step is the vast amount of conversation to sift through to find something of interest amongst the noise. `METHOD52` provides components to fetch data from social media [APIs](#), and to sample or cluster the data in different ways in order to aid exploration. For “search”, once the analyst discovers a conversation that seems interesting to study, or if the analyst already has some idea what types of conversations are of interest, the problem instead becomes identifying and isolating instances of these conversations among all other conversations (classes). `METHOD52` provides various methods for filtering documents, the primary of which is a classifier component which the user trains in an active learning loop.

In reality, both “explore” and “search” steps arise in the process of studying data, in a kind of loop or cycle. If we begin searching with a particular kind of conversation in mind, we first must find examples of it. However, the successful isolation of this class of data could still produce millions of messages. We may already have in mind likely sub-conversations, and if so we begin to try to isolate them, but otherwise must proceed to the alternate explore step, in order to discover sub-conversations. Once evidence of interesting sub-conversations is found, we cycle back to the search step to isolate other instances of the sub-conversations in the data. This cycle continues until the analyst has a sufficiently clear conception of the dataset (the engagement is concluded).

Section [2.1](#) first establishes the type of study that `METHOD52` users have typically conducted, in order to demonstrate the scope of this style of engagement. Then in Section [2.2](#) this engagement style is clarified with an example study which is typical of the social media studies of interest, and which exemplifies the cycle introduced above. Section [2.3](#) examines the explore-search cycle in the general case. Then Section [2.4](#) concludes with how the remainder of the thesis seeks to improve `METHOD52` to better support this process of engagement.

2.1 ENGAGEMENT IN TWO MINDSETS

This section identifies two broad mindsets of dataset engagement in studies undertaken by social scientists and humanities scholars when studying large text datasets with automated tools. And this thesis is primarily focused on one of them, since it is by far the most represented projects using `METHOD52`.

More common among humanities scholars, the first type is concerned with individual interesting examples. This mindset tends to occur when the researcher has access to a large collection of (often historical) records, and they are accustomed to a research methodology based on “close reading”.

Simply, close reading involves the study of single example passages of text in a careful, considered manner. Close reading contrasts with the concept of “distant reading”, which attempts to consider the collection as a whole, perhaps aggregating statistics over features of the text.

Researchers interested in close reading are less interested in even a 90% correct algorithm, because when individual examples are sampled the problem arises of explaining the 1 in 10 tagging mistakes throughout the text. When the data is studied in aggregate, it is possible to observe patterns despite the relatively small number of errors. Close readers are more interested in an accurate, careful analysis of individual passages.

This mindset is not the primary concern of this thesis. However, it is worth considering how this type of work might adapt to large collections of records. Some researchers may favour a random walk, looking at records for passages that spark their interest. However, this could be a long process. Ideally, automated tools would step in to assist the analyst’s exploration. The tool should increase the likelihood that interesting passages are presented to the analyst. It is this aspect of the first mindset that is most similar to the second mindset, and most able to take advantage of the present work.

The second mindset is most usual among social scientists, and is concerned with utilising large datasets for more quantitative methods of studying social phenomena. This is more like distant reading, and amounts to finding interesting classes of documents, rather than individual instances. It may be the case that the researcher is interested in characterising the collection (or sub-collection) of documents, rather than individual examples. Classes of documents are interesting because the frequency of instances representing the class lend weight to the conclusion that the class exists as a coherent trend or social phenomenon rather than a single example, which could be explained as an outlier (i.e. a quantitative analysis).

The difference between the two mindsets can be roughly characterised with the following metaphor. Imagine a book which represents the hypothetical full and complete analysis of a dataset. The first

mindset is searching for individual interesting pages or passages representing complete analyses of aspects of the dataset. The second is interested in a detailed contents page, the breakdown of the analysis into semantically meaningful classes and subclasses. Both mindsets may benefit from an index, through which individual features can point to smaller classes of passages.

This work is primarily interested in the second mindset. It is more common with social media data; the discovery and isolation of conversations fits this mindset well, since conversations are collections of documents representing a common theme or type. They can represent quantitative evidence of social media phenomena. These conversations, or types of conversation, will be referred to as *classes*, because they represent categories of text which share sufficient coherence in their semantics as to be considered as a single theme. Example classes might be: “islamophobia”, “support for Barack Obama”, or “witness information in a train derailment”.

2.2 ENGAGING WITH TEXTUAL DATASETS: AN EXAMPLE

What follows is a description of an example dataset analysis illustrative of the types of study that social media analysts aim to engage in using `METHOD52`. The example will illustrate what is meant by semantically coherent categories of text (*classes*), and why they emerge with central importance when engaging with the data. It is shown that the process of engaging with the data in terms of these classes occurs in an explore-search cycle (which is generalised in Section 2.3).

A very common problem is the desire to investigate the online reaction to some real-world event or ongoing hot topic. A typical example is a terrorist attack, a natural disaster, or a political election, debate or issue.

One such problem is discussed in more detail in Section 4.2, wherein the aim was to examine the reactions of `TWITTER` users to the 2016 Brussels terrorist bombings. The analyst does not necessarily begin with a finding that they are expecting to prove, instead they wish to discover properties of the data to report on. They want to know: what reactions are present in the data? What topics were discussed? How did the bombings influence the `TWITTER` users’ opinions?

8.4 million tweets were obtained containing the word “Brussels” using the `TWITTER API` from shortly after the last bomb detonation

on the 22nd March, 2016 until the evening of the same day, with the aim to investigate the nature of TWITTER discussion of the bombings. The collection is first filtered such that all those tweets whose language is not detected as English by TWITTER are removed (since there were no analysts with other language proficiencies that would be useful, though METHOD52's technology would support them), so the analysis concerns only English-speaking TWITTER.

The initial query "Brussels" is quite vague, but the timing of the collection helps to make more specific the data we hope to encounter. With TWITTER being a very event-driven reactionary platform (Wiberley et al., 2014) (explained further in Section 4.1.5), our confidence is increased that using a query term related to an event of international significance in the short period following it, raises the likelihood that the documents will contain sufficient relevant data.

The query is sufficiently general that we have not attempted to isolate a class of document (or conversation) that we expect to find concerning the attacks. This is the explore step. With the large collection of documents, we need a window into the data in order to begin defining observable social phenomena. Technically, before this explore step we have already performed a search step, by defining a class of "conversations pertaining to the Brussels attack" and attempting to isolate it using timing and a keyword query to a data source.

One simple way to begin the explore step is a random sampling of documents. Documents are selected at random and shown in batches to the analyst. After reading hundreds of example tweets, it became clear that there were many relevant to the attacks. The random sample was overwhelmed by tweets which constituted expressions of sympathy for the victims of the bombings. An alternative here would be to perform some form of clustering, then interpret the clusters by reading documents within the clusters.

Due to the abundance of tweets expressing condolences, any other type of relevant tweet was being suppressed from view. It may be of interest to measure the sympathetic response of TWITTER users, or we may wish to examine other types of document. Regardless, the next logical step then is to isolate those tweets which express condolences. Therefore, we find ourselves in the search step.

In order to isolate the condolence tweets, they must represent a coherent class of document. Even if the isolation procedure were entirely manual, the criteria for isolation would constitute the definition of the class of tweets representing condolences. An obvious tool for

organising documents into classes is a classifier. `METHOD52` provides an active learning interface for training a bespoke Naïve Bayes Classifier ([NBC](#)). A classifier was trained to distinguish condolences from the rest of the data. This step can act as a verification for the existence of the class in question; if it is possible to model the classification, then it likely holds weight as a coherent class of document. The inverse conclusion is less strong, however, since if the classifier cannot be made to model the desired classification, this may simply be a limitation of the model, rather than a flaw in the class definition.

Once the condolence documents have been isolated, they can be studied apart, or can be filtered away so that the other documents can be more effectively explored. This style of analysis of identifying patterns, and peeling them away to discover previously overshadowed patterns (here, classes) is shown to be effective by [Wibberley et al. \(2014\)](#) and is described further in Section [4.1.5](#).

In either scenario, the explore step is once again reached, because the objective becomes again to discover and define more classes within the resulting subset of data. Not only is the exploration accomplished through random sampling, but also the biased sampling of an active learning procedure. Documents are presented to the user of whose classification the classifier is most uncertain. This means that the analyst can also view interesting edge-cases, and documents which may not fit their current class definitions so easily.

In this study, the condolences were filtered away, and the remaining data was explored for more phenomena. This process was repeated several times, which lead first to distinguishing factual updates about the actual attacks from all other relevant comments. This class of documents contained information about suspects, victims, and other aspects of the investigation.

In the next explore step, after isolating and filtering away this “update” type of tweet, another class of news-like tweets was discovered dominating the data. These, instead of being updates about the actual attacks and investigation, were concerned with consequences of the attacks, such as service closures. Included in this class were details of helplines for those affected by the attacks.

These tweets contrasted with another class emerging which showed the personal opinions of the `TWITTER` users on a number of topics in light of the attacks. Searching for and filtering away the consequence tweets in favour of the personal opinions, the explore step began again.

Again random sampling was used to investigate the classes of personal opinion being expressed. Three substantial classes were identified, those discussing Donald Trump, Barack Obama, or Islam. In the search step, these three were isolated using simply the presence of keywords in the text. This produced three datasets representing three classes of personal reactions to the bombings concerning one of these three topics. The explore step occurs again on each dataset, discovering that each can broadly be divided into two classes of documents either in support or criticism of the topic in question in light of the bombings. From here, either the proportion of support or criticism for each topic could be quoted, or the cycle can continue attempting to analyse the classes of support and criticism.

This short study is typical of how analysts “engage” with a social media dataset using METHOD52, examining it for social phenomena. It shows that a central concept at each stage of the analysis, is the idea of classes of text, whether attempting to discover them, or apply their definition to structure subsets of the data for further analysis.

A similar procedure is found in many METHOD52 projects. [Wibberley et al. \(2014\)](#) describe a study in which the goal was to inspect tweets mentioning Father’s day, in order to find users to whom it would be appropriate to send Father’s Day marketing messages. The authors describe a “Russian doll” methodology of using classifiers to one-at-a-time pick out patterns in the tweets and explore the subset of data produced (much like the Brussels example). Again similar methodology arose in a study of the usage of racial and ethnic slurs on TWITTER using METHOD52 ([Bartlett et al., 2014](#)), in which the authors describe using classifiers to pick out relevant tweets, then identify types of slur, then identify categories of slur usage. And at each stage annotators manually inspect and categorise samples of slur usage. Additionally, the [HMIC](#) project that contributed to the 2016 Police Effectiveness, Efficiency and Legitimacy ([PEEL](#)) programme’s national overview report¹ exhibits a similar process at various stages. The relevant METHOD52 jobs show classifiers that break down the data in stages. For example, Figure 2.1 showed the user first separating out opinion-sharing tweets, before classifying them as whether or not they exhibit gratitude. Other jobs in the same project show samples being taken of the data at different stages for inspection.

¹ <https://www.justiceinspectorates.gov.uk/hmicfrs/publications/peel-police-effectiveness-2016/>

2.3 THE SEARCH & EXPLORE CYCLE

The example in Section 2.2 illustrated a typical procedure for “engaging” with a social media dataset using METHOD52. It describes a tension between the need to discover potential classes of text for findings to be data-led, and the desire to apply hypotheses about the existence of classes in the data. This tension leads to a methodology which alternates between the two goals as the data is explored. The cycle between the two is visualised in Figure 2.2. This section generalises the cycle, discussing what each step can involve, sub-steps within the explore and search steps, and where in the cycle studies begin and why, and advantages of and complexities with the explore-search characterisation of engagement.

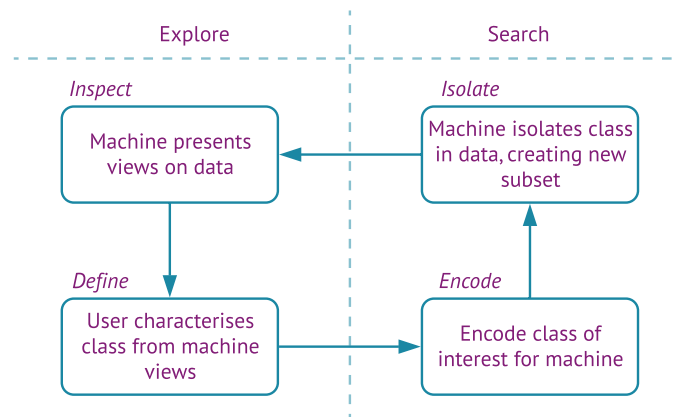


Figure 2.2: The search/explore cycle.

The first goal was referred to as the “explore” step and the second, the “search” step; the analysis in each step respectively is either discovering or isolating semantically coherent categories, *classes*, of text. A given study may have components of either, but typically exhibits a cycle of both.

In the Brussels example, the data is first collected on a very broad term, and the study sought to discover which conversations were present in the data. Therefore, the first phase was the data-lead “explore” step, in which the analyst attempted to discover classes of text. The method chosen for this was random sampling of documents. However, there are other options for this phase. Unsupervised methods are of particular use, since they discover structure in the data without the requirement of the analyst to have already defined classes. Examples could be some form of clustering or topic modelling, which

can be used to first group documents and guide discovery of potential classes.

The explore step can itself be conceptualised as two steps. Figure 2.2 defines them as “inspect” and “define”. In the inspect step, the machine presents the data to the user. This includes the actual data collection, and the methods used to present documents to the user, whether that is random sampling, clustering, topic modelling, etc. Then, in the define step, the user utilises the views on the data in order to construct class definitions which fit the data.

Figure 2.2 also divides the search step into two sub-steps. In the “encode” step, the analyst’s definitions of classes are encoded for the machine. This could mean simply defining the class in terms of the presence particular key terms, or could be for example in the form of training examples for a supervised learning technique. For most of the encoding steps, the Brussels example used training examples selected through active learning for an NBC. In order to separate the personal reactions into those relating to Donald Trump, Barack Obama, or Islam, the keyword method was used. The “isolate” substep is simply the machine’s application of the class encoding to the data.

Supervised learning methods most often fall into the search phase, since the supervision aspect implies the encoding step. This is then followed by an “isolate” step as this now fixed encoding is applied to the data, isolating the classes which were encoded in the encode step.

Figure 2.2 does not mark any step as the initial step, because the part of the loop which comes first depends on the goals of the analyst. In Section 2.2, the goal was only vaguely defined: to collect data on the term “Brussels”, the analysts knowing that much of the data on the day of the attacks would be about the attacks, in order find interesting conversations arising concerning the attacks. Alternatively, the analyst could specifically conduct a study into Islamophobia online, in which case, a class of document is already defined: a tweet which exhibits Islamophobia. The class could be more general: the characterisation of Muslims online. In these cases the first challenge is the isolation of these classes; the researcher only hypothesises that the Brussels data would contain them, which implies beginning with the search step.

Conceivably, a study could commence at any of the four sub-steps in Figure 2.2:

Inspect As in the Brussels example, if acquiring a dataset that is itself interesting, that is the analyst has not yet formed class hy-

potheses, but wishes to collect and explore the data in order to learn more about it, then they are beginning with the inspect step.

Define If the analyst does not first explore a dataset, but instead proceeds directly to establishing a class hypothesis about the data, which they wish to search the data for subsequently, then they begin with the define step.

Encode When starting with a hypothesis about a dataset's classes (e.g. replicating existing work), and attempting to train a bespoke learner to encode this hypothesis, the analyst is beginning with the encode step.

Isolate The analyst begins with the isolate step if they are applying a pre-trained algorithm, e.g. a positive/negative sentiment classifier, out-of-the-box to a dataset.

The cycle represents the overall aim of analysis, the high-level procedure. In practice, there are interesting complexities in the structure of the methodology.

In the encode step, when training bespoke classifiers, it is frequently the case that the first attempts do not perform as expected, since understanding of the dataset and the classes it best supports increases during the engagement process. This may be discovered on test data, before the analyst proceeds to the isolate step. However, it may also only be decided after proceeding through isolate to inspect that the data being presented is now too noisy to proceed. In this case, the analyst must return to an earlier encode step to try something different.

There are two major advantages of this methodology over a strategy that relies solely on pre-baked (non-bespoke) methods of isolating classes (e.g. pre-trained classifiers). The first most obvious is that the encoding and isolating can adapt to the discoveries in the explore step, tailoring the research to the data. [Wibberley et al. \(2014\)](#) has shown this to be often necessary in order to understand social media data in the right context, with the "Twitcident" principle (explained further in [Section 4.1.5](#)).

The second related advantage, is *fail-fastness*. When each iteration of isolating classes is followed by viewing the data and refining the encoding or applying new class definitions, the analyst can react to previous cycles producing incoherent results. This is not possible if applying a long chain of pre-baked methods.

More positively, the encoding step often produces insights about the documents being annotated. In the Brussels example, the classifier is trained in an active learning environment. We could say that the encode step here actually nests an entire explore-search cycle. The user is presented with a sample of documents of which the classifier is most uncertain (inspect), the analyst adjusts his/her understanding of the class definition according to the documents seen (define), then supplies training annotations (encode), and the system uses this new information to re-train, then re-classify the data (isolate) before presenting further samples to the user again.

Discoveries in any of the steps may lead the analyst to determine that the data collection method should ideally be altered. For example, the analyst may discover that more relevant data could be acquired by the addition of new terms to an [API](#) query. The additional data could alter or invalidate progress in the preceding steps.

Despite these complications, this characterisation of engagement with social media documents using `METHOD52` shows classes of text as the central concept in an investigative cycle of discovering and isolating them in the data as the analyst gains a deeper understanding of the data.

2.4 CONCLUSIONS

This chapter demonstrated the role of the discovery and isolation of classes in the style of engagement which `METHOD52` jobs often follow. Having identified that this type of analysis is based fundamentally on the building of a structure of classes, and having developed a description for this process, the purpose of the remainder of this thesis is to explore practical, adaptive methods for enhancing `METHOD52` to more effectively support the needs of social scientists and humanities scholars during this engagement process.

There are two broad styles in which one can approach structuring and defining a system of classes for a dataset. The first is to try to explicitly model classes and their relations, perhaps hierarchically. This approach could be described as top-down, and is akin to inferring an ontology. The idea of the semantic web would be an example of this type of approach. Alternatively, one might take a more bottom-up approach, in which the focus is on defining those properties which are important aspects of the classes, and by defining these properties, the class definitions emerge implicitly from the presence of these features.

An example which exhibits this approach is the “folksonomy”. Folksonomies are typically systems which permit the annotation of items (e.g. documents) with tags or properties which characterise the items in some way. Classes of item then arise implicitly from collections of items annotated in a similar way.

This thesis adopts the bottom-up approach. The bottom-up approach is interesting because it is mirrored by the machine learning algorithms: patterns are inferred over the properties of training documents. These properties are most often simply the words and phrases in the text, potentially matched with a class label annotated by the user (for supervised learning). Furthermore, the individual words are often relied upon by analysts to understand the views presented to them by the machine. For example, after applying topic modelling to a dataset, it is difficult to understand how the model categorises the documents, so often we will attempt to determine the words which are most indicative of, or relevant to, a topic in order to summarise this information for the analyst (Sievert and Shirley, 2014), e.g. using word clouds.

Given the bottom-up word-based focus, a fundamental aspect of the techniques explored in this thesis is the extraction of important words and phrases from the text. Therefore, Chapter 3 is concerned solely with establishing the technique which will be used for this task. It should be noted, however, that the technique while fundamental to many parts of the later chapters does not account for the entirety of their contributions.

The active learning system of METHOD52 based on DUALIST, together with METHOD52’s pipeline architecture, is fundamental to the overall explore-search cycle. Therefore, Chapter 4 begins with this tool. The technique established in Chapter 3 is applied to better support the discovery and isolation of classes in addition to other methods, whose focus remains with the individual features of text.

The data collection strategy (in the inspect sub-step of the explore step) must also be considered because it affects the possible classes that can be extracted from the data. Often, to collect social media data, the analyst must provide words or phrases to the social media company’s API. Texts are then returned which match those terms. Even use of modern search engines follows a similar structure. We are accustomed to finding data through use of queries composed of key words and phrases. If a dataset is constructed on the keyword “Brussels” it is likely to be feasible to define classes concerning dis-

cussions relevant to Brussels. It is less likely that it would be possible to derive a class for attitudes toward “The X Factor” in such a dataset. Therefore, the potential for classes of interest are defined by the set of query terms provided to the [API](#). Which means that again, these surface features are critically important, and the [API](#) query they form is important.

Additionally relevant to the data collection strategy, is the fact that discoveries in other steps of the explore-search cycle can necessitate changes in the data collection strategy (the query terms supplied to the [API](#)), or at least reveal the possibility of missing out on relevant data if the change is not possible.

Given the importance of the data collection method, Chapter [5](#) explores strategies using the key phrase extraction technique from Chapter [3](#) for automatically adapting the social media data collection based on the discovered terms in the collected data in order better support class discovery by capturing additional relevant material.

SURPRISINGLY FREQUENT FEATURES

Chapter 2 described how when analysing large sets of social media text using METHOD52, the analyst can be thought of as attempting to discover and define *classes* of data. Once the semantics of a class are defined, the analyst wishes to isolate for analysis those documents which represent given classes, and this effort continues in an explore-search cycle as subsets of the data are defined and subjected to the same treatment. This thesis adopts a bottom-up, surface feature based approach to improving this process, asking: *how can feature discovery be used to support the characterisation of semantically coherent classes of text?* This approach often relies upon a method for extracting and ranking important words and phrases in a corpus. For this reason, the first sub-question outlined in the introduction was: *how can features be identified which provide useful bases for characterising classes of text?* The features of interest here are words and phrases. Therefore, the contribution of this chapter is to introduce a new method for extracting these phrases.

In some cases, the machine has defined classes which the researcher must interpret (the “inspect” step, see Figure 2.2). For example, after applying a clustering algorithm, any document can be described as belonging to one or more clusters. The analyst must then interpret the semantics of the class that each cluster represents, a process which frequently turns to displaying important words and phrases for a given cluster/topic, for example in a word cloud.

In other cases the researcher is responsible for having defined a class, through observation of the text, or knowledge of the text domain (the “define” step). The user then wishes to isolate that class in the data. In order to accomplish this at scale, a machine learning classifier can be trained to model the desired classes (“encode” step), and isolate the documents representing them (“isolate” step). In the currently dominant trends of machine learning, individual surface features play a key role here in how the machine models classes, in both the discovery (e.g. clustering/topic modelling) and isolation (e.g. classifiers) of classes. Furthermore, surface features are important for user interpretation of machine-defined classes, and interpretation of document collections in general. Lastly, these word and phrase fea-

tures are also important for defining the queries to data collection APIs for even generating the datasets to begin with.

Through discussions for collaborations using METHOD52 with social scientists and humanities scholars, we (the TEXT ANALYTICS GROUP) find anecdotally that there often seems to be a fundamental mismatch between how the machine models the classes, and how non-technical analysts understand the machine to work. The analysts interpret the machine’s process as building an explicit semantic model representing the meaning of concepts and the relationships between them, so that it can perform clustering, classification, etc. with full understanding of the domain, as they as humans would.

Given the importance of word/phrase features, it was decided that a feature-driven, bottom-up approach to supporting the explore-search cycle would be fruitful. Therefore, most of the thesis is concerned with practical methods of enabling METHOD52 techniques to better discover and exploit features.

In Section 1.1, the idea of some features in a corpus being more important than others was introduced, as was the notion that features which occur more often than one would expect (*surprisingly* frequent features) are likely to be important features. This is a well-established idea (Rayson and Garside, 2000), but which has no single simple approach. How do we define our expectation? By what measure do we make the comparison between observed and expected?

This chapter’s purpose is to introduce and describe the technique used for extracting key phrases, which will be referred to as Surprisingly Frequent Phrase Detection (SFPD). Design decisions as a result of the initial problems from early prototyping are also discussed in this chapter. In addition, the ways in which the algorithm is adaptable to different scenarios is considered. SFPD will be used in large parts of following chapters. Chapter 4 applies feature discovery techniques to support characterisation of classes of text, and in Chapter 5 SFPD is at the core of a framework for discovering new words and phrases for automatically adapting queries to TWITTER in order to collect additional relevant data.

Section 3.1 discusses related work in keyword extraction and corpus analysis. Section 3.2 discusses the overall strategy for evaluation in this thesis, since finding surprisingly frequent features has a substantial role in the following chapters. Section 3.3 introduces and justifies the chosen measure for feature surprisingness, and Section 3.4 describes how this measure is extended to additionally produce *multi-*

word terms. Sections 3.5 & 3.6 are concerned with the early issues that arise when adapting the technique to different types of data (contrasting with Chapters 4 & 5, which while similar, focus specifically on the active learning and adaptive streaming capabilities). Section 3.7 deals with how to establish and exploit our *expectation*, the reference point to which feature observations are compared using our measure of surprise. Section 3.8 shows how the existing structure of METHOD52 can be used to further exploit the surprisingly frequent feature analysis. Section 3.10 discusses the role of corpus homogeneity when using this technique.

3.1 RELATED WORK:

KEYWORD EXTRACTION & ANALYSIS OF CORPORA

Keyword extraction is a well-established method for comparing, summarising, and generating insight from datasets (Rayson and Garside, 2000; Witten et al., 1999), which remains popular (Abilhoa and De Castro, 2014; Marujo et al., 2015; Jordanous and Keller, 2016). Therefore, Section 3.3 draws on keyword extraction work to establish a method for extracting key terms from datasets.

In order to build key term lists, a measure must be selected for judging the importance (“key-ness”) of terms. This section therefore describes measures used in previous work.

Keyword extraction techniques can first be broadly divided into two categories: *document-based* techniques whose aim is to extract keywords on a per-document basis (such as Timonen et al., 2012; Marujo et al., 2015), and *corpus-based* methods which extract keywords from a corpus as a whole (Abilhoa and De Castro, 2014; Jordanous and Keller, 2016). The focus of this section is on corpus-based techniques, because the primary goal of keyword extraction in this thesis is to help analysts understand a corpus, not to provide keywords for every single document, of which there could be millions.

This section first focuses on corpus analysis techniques generally, then examines work that corresponds to the most common scenario in this thesis: corpora composed of many small documents such as tweets.

Analysing corpora for key terms is essentially a comparative process. When searching for a key term, one is looking for a term that is important to the corpus. This is similar to asking: what about the

use of this term is important compared to its use in other contexts? I.e. how does its use compare to our expectation of its use?

Our “expectation” is usually defined using some other corpus. Rayson and Garside (2000) explicitly divide the task of comparing corpora into two broad categories:

1. Comparison of a sample corpus to a larger corpus.
2. Comparison of similar sized corpora.

In 1, the larger corpus essentially provides a norm or standard against which any number of smaller samples may be compared. This standard is also called the “background”, “normative”, or “reference” corpus. By comparing multiple samples against a common standard, we also technically gain an understanding of how the smaller samples compare to each other by comparing their differences with respect to the larger corpus. In 2, we are directly comparing similar sized corpora.

Rayson and Garside (2000) raise awareness of four specific issues when comparing two or more corpora:

1. Representativeness
2. Homogeneity
3. Comparability
4. Reliability of statistical tests

Representativeness is of particular importance for a corpus functioning as a standard/norm. In order to actually function as a standard, i.e. to expose those features of the sample corpus that deviate significantly from the norm, the standard must be highly representative of the norm. Otherwise, any significant gap in the standard will lead to features in the sample corpus seeming to deviate from the norm, when in reality they do not.

The homogeneity of a corpus is especially important when comparing corpora of similar size. For corpora that are not sufficiently homogeneous, any results derived from their comparison may be due only to particular sections of a corpus that are unlike other sections in itself and the other corpora. Kilgariff (1997) presents a method for measuring corpus similarity and homogeneity using a χ^2 -based statistic. The author argues that similarity of corpora can only be understood relative to its homogeneity.

In order for the differences between corpora to be explainable by their inherent differences rather than quirks of the data, the corpora should be comparable. They should be sampled in the same way.

And finally, the authors call attention to the reliability of the statistical test used for the comparison in relation to the size of the corpora.

Kilgarriff (1997) shows that the χ^2 statistic in its basic form is inappropriate for the task (despite its popularity) since words do not occur randomly in text, which violates the statistic's assumptions. The score increases with the frequency of terms, which naturally increases with the size of the corpus anyway. In order to overcome this problem, the measure is calculated over equally sized samples of the corpora being compared, and the measure is then normalised by the number of words used in the comparison. It is left as future work to determine how to interpret similarity between corpora of different sizes.

Dunning (1993) state that in a "moderate-sized" corpus, words that have a frequency of less than 1 in 50,000 make up about 20-30% of typical English language. In other words, rare words are common; they do not follow a normal distribution. This violates the base assumption made by the χ^2 test and z-score tests, implying that the measures are not reliable for very rare terms. Dunning (1993) instead proposes the Log-Likelihood (LL) ratio measure, which does not rely on the assumption of a normal distribution. Daille (1995) shows empirically that the LL method is effective for finding terms, and approximates reasonably well human judgements of importance.

Knowing the weaknesses of a measure is not necessarily a reason to abandon it. Instead, the weaknesses can be offset with other information. Granger and Rayson (1998) use corpus linguistics software developed by Rayson and Wilson (1996) to compare corpora of native and non-native English. The system uses the χ^2 measure, and offsets its unreliability concerning common terms and very rare terms by showing the dispersion value and concordances of a term so that the distribution of the corpus can be taken into consideration.

It is the LL method that Rayson and Garside (2000) apply to field reports of a series of ethnographic studies at an air traffic control centre. They use a 2.3 million word subset of the British National Corpus (BNC) which is derived from transcripts of spoken English as the normative (or background) corpus. The measure reveals semantic categories that include important objects, roles and functions in the air traffic control domain. This method has also been applied to previous work (Rayson et al., 1997), which studied social differen-

tiation in the use of English vocabulary, using the demographically-sampled spoken English subset of the [BNC](#). The [LL](#) method has since become quite popular. Examples of use include: determining daily community trends on [TWITTER](#) ([Java et al., 2007](#)), term extraction in ontology learning ([Gacitua et al., 2008](#)), and building models of creativity ([Jordanous and Keller, 2016](#)). All showing that insight can be generated through keyword extraction.

When determining trends in project management literature using keyword analysis on journals in the field, [Crawford et al. \(2006\)](#) tried both a χ^2 approach and the [LL](#) ratio, and decided that keywords identified by the χ^2 statistic had a greater correspondence to a natural language understanding of keyness in the text, which shows that the suitability of a measure may be task-dependent or dependent on the motivations of the analyst, despite potential theoretical concerns.

[Kilgarriff \(2001\)](#) also describes the Mutual Information ([MI](#)) statistic ([Church and Hanks, 1990](#)), which when comparing corpora states how much information a word provides about a given corpus with respect to the joint corpus. Unlike χ^2 , whose main issue is over-weighting common terms, [MI](#) tends to over-emphasise rare terms. This is a problem given that many words will have a small number of occurrences, as will most bigrams, trigrams, etc.

[Kilgarriff \(2001\)](#) suggests that it would not be surprising to find that no single measure works well for both high frequency terms and low-to-mid frequency terms, given that whatever makes either category of term distinctive or interesting may be different. The Mann-Whitney test is suggested as an alternative to χ^2 for the reason that it did not add undue weight to high frequency items. However, as noted by [Baron et al. \(2009\)](#), when the Mann-Whitney test was applied to compare American English to British English using the Brown corpus and Lancaster-Oslo-Bergen corpus ([LOB](#)), 60% of the terms were still marked as significant. Furthermore, words with less than 30 occurrences were excluded to avoid zeros in the test measure, which meant ignoring 92% of the joint corpus.

Fisher's exact test can be used on contingency tables with low expected frequencies, but the measure is computationally expensive since it requires the computation of factorials ([Baron et al., 2009](#)). [Weeber et al. \(2000\)](#) combine the log-likelihood measure with Fisher's exact test to cover all frequency ranges when extracting side-effect related terms from medical paper abstracts.

A different approach involves first analysing the corpus for “topics”, then separately finding keywords that summarise those topics (Zhao et al., 2011; Sievert and Shirley, 2014). Topic modelling is accomplished frequently with Latent Dirichlet Allocation (LDA) (Blei et al., 2003), which models documents as distributions over topics, and topics as distributions over words. The number of topics must be decided in advance, as well as parameters α and β for the per-document topic priors and per-topic word priors respectively. A similar approach can be taken after any kind of clustering or grouping of documents. Santhan et al. (2017) uses an MI measure to find terms which summarise the results of a clustering algorithm.

Sievert and Shirley (2014) devise a statistic “relevance” which is used to rank terms within a topic, to provide the user with a better understanding of the topic. The statistic is a linear interpolation between the likelihood of a term appearing in the topic and its “lift”. Where lift is defined as the ratio of the in-topic likelihood to the prior probability of the term over the entire corpus (similar to MI). Adjusting the interpolation parameter λ provides a convenient method for placing more importance on either the frequency within topic (in our case, the target corpus) or the surprisingness of a term’s frequency in-topic given its frequency across all topics (our background corpus). A similar measure is applied to a corpus of tweets by Zhao et al. (2011), but where the background corpus is a separate normative corpus.

The approach in this thesis is similar, but instead of using topic modelling to produce target corpora, METHOD52 provides user-trained classifiers and simple keyword-based filters, which allow the user to carve out topics or other categorisations from the data. The documents in these categorisations can then be used as target corpora.

Other approaches emphasise co-occurrences of words in tweets, using them to build graphs of terms, where terms have more highly weighted connections if they frequently co-occur. Then keywords are extracted using features of the resulting graph. For example, Abilhoa and De Castro (2014) use measures of graph centrality to find important keywords.

Once a method for obtaining keywords is established, the next issue is to establish how to find multi-word phrases of importance, since individual words are rarely the complete story. A common technique is to simply extract all ngrams (up to some computationally feasible n) and count statistics over their occurrences in the same fash-

ion as for unigrams. A serious problem with this method is that this data will usually be sparse. The greater the number of tokens in the ngram feature, the more likely that the feature will be sparse enough to not occur in the comparison corpus, raising the likelihood that the feature will seem to occur surprisingly often.

An alternative strategy is to perform the analysis of keyword importance on unigrams alone. Once a keyword is discovered as important, find the possible phrases in which it occurred that might explain its surprisingly frequent occurrence. [Baroni and Bernardini \(2004\)](#) employ such a strategy. The authors present a set of tools for collecting a corpus from the web. Using a small set of highly relevant seed terms, they perform a `GOOGLE` search. New unigram seed terms are extracted from the search results by comparing the frequency of occurrence of terms in the results with their frequency in a background corpus. The comparison is made with the log odds ratio measure.

Once unigram terms have been determined to be of interest, they are expanded to multi-word terms. The contexts of the unigrams are examined for the frequent phrases that occur containing them. In order for a candidate multi-word phrase to be approved, it must:

- not be part of a longer multi-word term with frequency above $k \cdot f$, where k is a constant between 0 and 1, and f is the frequency of the current term.
- not contain a shorter multi-word term with frequency above $\frac{1}{k} \cdot f$.
- not contain stopwords except common connecting phrases that occur between approved unigrams (e.g. “of the”).

A similar approach is adopted in this thesis and described fully in Section 3.4.

[Saleem et al. \(2017\)](#) argue that keyword extraction approaches, and those that require manual annotation, such as `METHOD52`’s bespoke classifiers, introduce the biases of the annotators and do not take into consideration that vocabulary can be very similar across fundamentally different data. Their examples compare the documents from hate groups and support groups for the same topics, which are obviously fundamentally different, but the authors show a large vocabulary overlap between them. Their solution involves modelling the language in defined communities (e.g. subreddits¹ on `REDDIT`), rather

¹ Online communities dedicated to a common purpose/interest on `REDDIT`

than having analysts manually annotate documents as hateful or supportive. However, the existence of such clearly-defined communities in a given dataset is not guaranteed. For our main data source, TWITTER, there are no analogously defined communities. Instead, the effectiveness of the methodology and the categorisations of the data must be part of the bespoke analysis.

3.2 EVALUATION STRATEGY

This section describes the overall strategy, and its motivation, used for evaluating the techniques introduced in this thesis.

This thesis is concerned with improving researcher ability to engage with and generate insight from text data, by using more explicit feature discovery strategies to improve the process of discovering and isolating classes in text data. It was desirable to select an evaluation strategy which complemented how METHOD52 is used and is useful. METHOD52's strength is its collection of methodologies and tools, and their adaptability and generalisability for different research and data. It provides the general "job" interface, which allows the user to construct pipelines of processes for defining classes, then isolating them, and repeating the process on subclasses. Section 4.1.5 describes the adaptable methodologies refined by [Wibberley et al. \(2014\)](#) utilising METHOD52's classifiers and pipeline building framework to structure classification tasks for solving different types of problem. Classifier-based strategies are of most interest here since the aim of Chapter 4 is to improve their capability to execute the explore-search cycle.

METHOD52 tries to encourage analysts' engagement with the data. The analyst must construct their processing pipeline in a graphical interface which visualises the actual processing strategy. METHOD52 follows what DUALIST started, and tries to combine the strengths of both the machine and researcher. This is accomplished partly through the active learning classifier component based on DUALIST, but also with the adaptable pipeline structure, which the analyst can use to customise a strategy of different types of filtering and annotation.

The aim of the METHOD52 platform is not to have the most cutting-edge accurate classifier algorithms; if they were evaluated on standard sentiment or topic classification tasks, they would be outmatched by more complex state-of-the-art techniques. This approach is a purposeful choice. METHOD52 is designed to push the user toward read-

ing and analysing the data themselves in a productive manner (e.g. instead of reading millions of tweets), which is supported by technology, rather than trying to work with a black box that simply produces a number. This contrasts with systems that simply strive to provide, for example, the most accurate classification accuracy for a specific classification problem and dataset, where the classification results alone constitute the analysis. The surprisingly frequent phrase detection system, for example, need not find all and only the most interesting features of a dataset, it should simply provide a good enough signal that it can characterise aspects of the data.

Furthermore, there are important practical concerns when building this type of tool:

- Tools must exhibit timely executions using minimal resources to encourage interactivity and collaborative use.
- Tools must have fairly intuitive explanations for their workings, so that users without a background in computer science can understand and explain at a high-level the processes that lead them to their research conclusions.
- Tools must be modular and flexible enough to be built together in different flows in order to be adaptable to new problems.

The issue, therefore, is *not* to evaluate how well the system performs sentiment analysis, topic tracking, or corpus linguistics. The task is to determine its capability to generate insight from datasets. Evaluating such a capability is incredibly difficult. Every analyst has their own purpose in mind, and follows their own line of inquiry. Therefore, the approach for evaluating `METHOD52` and any new additions to the platform in the past has been through case study and theoretical argument. `METHOD52` has been applied and refined through use in many projects (Wibberley et al., 2014; Bartlett et al., 2014; Miller et al., 2016; Krasodonski-Jones, 2016; Conway et al., 2017). This is a qualitative approach to evaluation because it reasons about the underlying problems of current user approaches and how they can be alleviated by these techniques, using project outcomes, logical argument, informal discussion with project leads rather than seeking to statistically quantify performance of individual tools on fixed tasks.

In evaluation of this type, a clear advantage is that we are immediately confronted by whether a system can help produce insights,

since it is applied directly to a scenario for which it is designed. We can also tailor solutions to actual arising problems in projects. However, the drawback is that it is more difficult to obtain a conclusive result. We do not have a measure that states exactly how much better a tool performs, whose statistical significance we can ascertain. Furthermore, embedded in a project with different specific goal, there is often not the luxury of time and resource to perform the entire project several times with completely different instantiations of the technique being evaluated. Additionally, without the vast resources required to undertake many case studies with many other collaborators, conclusions will necessarily be drawn from few data points.

With relation to keyword extraction for the purpose of adapting a TWITTER query (the task primarily considered in Chapter 5), [Wibberley et al. \(2013\)](#) discuss the idea of manually updating a TWITTER query based on high Information Gain (IG) features² according to a trained relevancy classifier model. The authors suggest a method for estimating the relevancy precision of any such feature: using a TWITTER query that contains only the proposed feature, collect a sample of tweets containing the feature, and use the proportion of those tweets marked relevant by the classifier as the relevance precision for the feature. This can then be compared to the relevance precision of the original query terms.

The motivation for augmenting the API query is to increase recall (discussed further in Chapter 5), to widen the scope of the data that is considered relevant. Therefore, a key weakness of this evaluation technique is that the quality of new query terms is assessed in terms the previous narrower definition of relevance. Terms that obtain relevant documents but whose vocabulary is markedly different to those in the original relevant training data, will be unfairly penalised by this measure. For this reason, the evaluation technique is potentially useful for excluding very irrelevant terms (explored in Section 5.6), but poor as an intrinsic measure of the performance of SFPD as part of an adaptive querying method; if we were to evaluate SFPD by e.g. the average relevancy precision across all proposed features, not only would it be an underestimate (by a degree dependent on the task and classifier training), but also it would ignore whatever utility or insight that the user did gain from the newly queried data, or how much more rapidly insight was gained. Without embedding the exploration in a project with a desired outcome, it is difficult to inter-

² Features whose presence or absence in a document on average most reduce uncertainty in the classification decisions.

pret the measure usefully. Is 50% good or bad? It would likely be task-dependent. Does the utility of *SFPD* even correlate with relevancy precision? Studies would be needed to answer whether the measure is even an appropriate proxy for utility to the user. So despite this task being exactly the task in Chapter 5, the quantitative measure suggested by *Wibberley et al. (2013)* is not suitable for the needs of this thesis.

A similar argument holds for other instantiations of intrinsic measures of performance for our goal. The topic tracking task in Text Retrieval Conference (*TREC*)-2012 (*Soboroff et al., 2012*) and subsequent years define topics to be followed in a fixed dataset. Are they the kind of topics which when tracked produce useful insights? Are the topics too focused or broad? Does producing a system that better tracks these topics actually help a user generate insight in the general case? These questions are difficult to answer by producing a score according to some dataset, or set of human judgements. Instead, if the technique is embedded in a project with separate aims, the technique's usefulness can be tested in a realistic setting. Its usefulness can be assessed in the context of user motivations and requirements.

As an example of case study analysis *Rayson and Garside (2000)* show lists of keywords produced by their measure of keyness on particular corpora, with particular applications in mind. The authors argue how these terms can lead to different insights for the *given applications*. This shares the qualitative properties of *METHOD52*'s case study based evaluation. A similar approach is taken by *Jordanous and Keller (2016)*. *Rayson (2008)* purposely constructs a keyword discovery technique that demands user intervention for the analysis of terms, explicitly agreeing with the conclusions of other work that the analyst should not be excluded from the analysis.

The software and methodologies in this thesis were not developed to solve an existing task like topic tracking, which itself has a fixed notion of what it means to follow a topic. Neither were they developed to produce a final score, or collection of keywords that can be reported as the complete analysis, which itself could be evaluated intrinsically. Instead, they were developed to increase the potential for insight gain in discovery and isolation of classes in text data. This thesis investigates problem of providing tools for exploratory purposes.

A practical, but significant problem in instead attempting a quantitative evaluation for tools designed for exploration, is the sheer number of *options*. Exploration begins with interactive play and experi-

mentation; the user has many parameters and strategies at their disposal, and as they experiment with them, they gain insight. But to individually devise a quantitative study for the contribution of each parameter and strategy to each type of insight at each stage of the study (if such an experiment is even possible in each case) would be impossible in the time devoted to this work. Instead, a case study conducted in collaboration with the users makes it possible to follow the steps of experimentation, and observe strategies used and where tools were used successfully or problems were encountered and how they were overcome.

Given the ambition to provide a framework that allows users to explore and adapt to their data and problems, a substantial part of the evaluation of these approaches should consider the analyst's possible requirements. Instead of producing an evaluation of the form "this approach is best", which is often an impossible assertion in evaluation by case study, it is more desirable to make observations such as "when the analyst requires *A*, *B* is a problem that arises. *C* is a possible solution, but has the following drawbacks". The latter type of evaluation shows at a higher level how to reason around the problem space, rather than trying to fit a broad solution that works best on average.

Given that the overarching aim for both Chapters 4 & 5 is to augment user ability to explore and engage with their data, and produce a practical system that is tried and tested, this thesis adopts the evaluation by case study approach (except where it is appropriate to do otherwise, as in the evaluation of adapting dependency parsing to TWITTER language in Section 4.6.3).

The thesis contributions are not only the technology which is produced, but also the lessons from the behaviour of the technology under different circumstances. Due to the use of theoretical argument or user experience for evaluation, strategies are produced that can help guide analysts' use of the new METHOD52-based tools. The intention is also to take lessons from the thesis to augment documentation and tutorials for METHOD52 users.

Chapters 4 & 5 will introduce datasets and projects that involve METHOD52 and the trialling of the thesis approaches. While these projects may be individual studies, the issues discussed are typical problems for which METHOD52 is designed. Each chapter will discuss problems that were encountered and how the approach was improved. Equally important are discussions on how the techniques are

adaptable to new tasks. Approaches will be evaluated by examining the benefits that they produced, and the problems encountered in their use.

Particular projects are referenced where appropriate, along with accompanying public reports or papers where possible. Otherwise, smaller studies were also carried out specifically for this thesis, which will be fully described.

In order to help guide and contextualise the theoretical discussion and smaller studies undertaken in this thesis, informal feedback was sought from analysts using `METHOD52` via the project leads overseeing its use. The feedback was acquired by open-ended meetings or calls lasting up to an hour with the project leads (from `TEXT ANALYTICS GROUP` or `CASM CONSULTING LLP`). The following questions were asked about a given new/changed `METHOD52` feature, encouraging open-ended answers:

1. Was the new feature used?
2. Why was it used or not used?
3. How was it used?
4. What problems (if any) arose using it?
5. What insights (if any) did the users gain while using it?

3.3 MEASURE OF SURPRISINGNESS

This section describes the measure chosen for extracting key terms from a corpus, and the rationale for this choice.

The purpose of `METHOD52` keyword analysis is as part of a wider exploration. Much like the way classifiers are used to divide up and examine text data, so are the keywords; they can be used to provide a view on the data that improves the user's capability within the active learning process, and also used as a means for collecting additional similar data. The keywords can be used, with sample contexts, to explain the properties of corpus. Therefore, there is less emphasis on providing a measure that produces some kind of canonical keyword list that flawlessly accounts for the eccentricities of the data (as might be the goal of other keyword extraction research), and instead more focus on an adaptable measure that gives the user intuitive options to draw out insights from the data, since ultimately the analysis and interpretation of the datasets is in the hands of the researcher. The

keywords and phrases are evidence for classes that can be defined over the data, and will be used to support techniques for improving the explore-search methodology in `METHOD52` in subsequent chapters.

It is far more common to be using automated tools (including `METHOD52`) to analyse big data rather than a few documents, since the analysis of a few documents can be achieved manually. So it was at least necessary to choose a measure that does not overestimate the surprisingness of common features, which excludes the χ^2 measure (Kilgariff, 2001). The Mann-Whitney test also seemed to overestimate significance (Baron et al., 2009).

A common theme in the microblog topic tracking literature is the use of temporal features when evaluating terms: the frequency of terms are tracked over time to identify more bursty terms, ones that may be more indicative of emerging topics (see Section 5.1). Temporal features are, however, not used in our approach. While results with them were shown to be mixed (Kim et al., 2012), the main reason for not including them in our measure of surprisingness was flexibility; it is not often that only emerging or “bursty” terms are of interest if the goal to develop an understanding of a given dataset (e.g. its themes, trends, etc.).

The relevance measure proposed by Sievert and Shirley (2014) is designed to rank those terms that are most highly associated with a target collection of documents that represent a topic (“topic” according to `LDA` topic modelling). In `METHOD52`, when data is collected it is usually categorised by the terms found in the messages, or by classification decisions, or simply by the query terms that produced the data. This shows clear parallels with summarising topics; the measure could help discover relevant terms in whatever categories the analyst has chosen to sort the data.

The relevance measure is an interpolation between the likelihood $P(f|t)$ of feature f in a target category t and its “lift”: the ratio of the target likelihood to its marginal probability across all categories. The measure is calculated as follows, where λ is the interpolation parameter, a number between 0 and 1:

$$Relevance(f) = \lambda \cdot \log(P(f|t)) + (1 - \lambda) \cdot \log\left(\frac{P(f|t)}{P(f)}\right) \quad (3.1)$$

The measure can be generalised by estimating $P(f)$ from something other than just all categories together; instead, we could use a

separate reference corpus or other categories individually for a direct comparison. This makes the *lift* part of the measure essentially the **MI** measure of importance/surprise (Church and Hanks, 1990).

Deciding how to estimate the prior probability of a feature is basically the process of establishing an *expectation*, as it defines what the measure would consider *surprising*. This is discussed further in Section 3.7.

An important goal of feature discovery in this work is flexibility. The system should not be so constrained that it cannot be generalised and applied to new analyses and data. The interpolation parameter λ permits a certain level of flexibility. The researcher can choose how much importance is placed upon the likelihood within a category, or the surprisingness in relation to the reference corpus. This level of interactivity lends some adaptability to this measure, since it allows the user to easily tailor the measure of surprise to problems where either the likelihood or lift is more indicative of importance, or even gain several different views on the same data.

The drawback is that the **MI** component is considered unsuitable for very low frequency features (around a frequency of 5) (Kilgariff, 2001). This issue can be avoided by filtering out terms below a certain frequency cut-off. This solution is only not suitable in very small corpora, where the key terms may actually only occur a few times. The interpolation parameter λ can be used to place more importance on the more frequent surprising terms. Varying it can tailor the level of importance to each new dataset.

Dunning (1993) introduced the Log-Likelihood ratio measure after criticism of the χ^2 and **MI** methods' issues with very common and rare terms. The Log-Likelihood method has been empirically shown to be effective for finding terms, and to approximate reasonably well human judgements of importance (Daille, 1995). And as shown in Section 3.1, the measure has since been used successfully. However, some research using this measure has found the need to filter out features with very low frequency anyway (Jordanous and Keller, 2016). However, unlike the relevance measure, there is no interactivity. Instead, it gives a single decision about the most surprising terms. This is desirable if the end-goal is for the key terms to be the complete analysis, a list of terms that canonically define the corpus and that can be directly compared to those in another corpus. But the aim here is to provide a tool that aids the researcher's engagement with and ana-

lysis of the data, a tool which improves the active learning process and enables streaming with an adaptive query.

Given their strengths, the relevance and LL measures were the most considered candidate measures. The relevance measure was selected for use in this thesis, in favour of its flexibility. However, any approach in this thesis that does not explicitly concern the interpolation parameter λ could be changed to feature a different measure if found to be superior in the future.

The measure can be applied to any feature extracted from a corpus. But unless otherwise stated in this thesis, it is being applied to unigram features.

3.4 EXTENDING TO PHRASES

It is not sufficient to simply rank the unigrams of a corpus. Alone, a unigram may not represent the full story as to what makes it interesting. It may sometimes be the case that only the additional context of surrounding words present the full picture. Furthermore, query terms for social media APIs can be more than just single words, so given that the term surprisingness measure is also a core part of the adaptive querying method in Chapter 5, it should be extended to phrases for this reason also. This section details the method by which surprisingly frequent words are extended to phrases.

It is difficult to gain accurate statistics over terms that consist of more than a single word without access to corpora consisting of billions of words. Bigrams, trigrams, etc. will often seem far more surprising than they should be when they do occur, since it's likely that they won't have happened to occur in the comparison corpus, due to their sparsity. This is especially true of a measure that may overestimate the surprisingness of rare terms. After having applied the surprisingness measure, the analyst is presented with a list of words sorted by their surprisingness. The same technique could be applied to bigrams, or trigrams. But this greatly increases the terms and their counts that have to be tracked, and the larger the n in n -gram, the more likely the occurrence of a feature will seem surprising, simply by virtue of n -gram features being inherently sparse.

Instead of trying to find n -grams that are surprising according to our measure, we can find phrases that explain the surprisingness of unigrams: what phrases did surprising unigrams commonly occur in, which seem to account for much of their usage? This approach

was taken by [Baroni and Bernardini \(2004\)](#), who recursively searched for valid $n + 1$ grams from the original important unigrams, which occur with sufficient frequency to be considered interesting. Valid and interesting is defined as follows:

- A phrase cannot contain stopwords except common “connecting phrases” that occur between surprising unigrams (e.g. “of the”).
- A phrase must have a frequency above a certain threshold.
- A phrase cannot be part of a longer phrase with frequency above $k \cdot f$, where k is a constant between 0 and 1, and f is the frequency of the current phrase.
- A phrase cannot contain a shorter multi-word term with frequency above $\frac{1}{k} \cdot f$.

This essentially means that for a valid phrase to be proposed as of interest, its frequency must account for more than a specified fraction of the occurrences of its shorter form. For example, if $k = 0.5$, then a phrase must account for at least half of the occurrences of the next smallest ngram.

This idea is implemented in this thesis with a few differences (more detail in [Section 3.4.3](#)):

- A method for sorting the phrases generated for unigrams in order to permit the user to select a fixed smaller number of phrases per surprising word, and to ensure that those phrases chosen are more likely to be useful. Phrases are compared by identifying the largest common sub-ngram and sorting by the frequency of occurrence of the divergent sub- $n + 1$ grams.
- It is difficult to know sensible “connector phrases” across all possible datasets that `METHOD52` will encounter. Therefore, instead of disallowing stopwords, the number of stopwords and their position in a phrase only affects its sorting order. If two phrases being compared share the same frequency, then the phrase that contains fewer stopwords is sorted above the other. If this value is also equal, then the phrase that has the most stopwords at its beginning and end is sorted below.
- Instead of excluding sub-phrases of interesting phrases, their sorting order is affected such that they will never appear above their longer counterpart.

Similarly to Baroni and Bernardini (2004), the phrase frequency threshold and k threshold are exposed to users as parameters to be altered to suit their data. Typically, various values are tried and one or more may produce different interesting results. In this thesis, k values experimented with are usually all values between 0 and 1 in steps of 0.1. The phrase threshold is set to 0.3 where unspecified, since this generally produces satisfactory results in experimentation. Occasionally 0.4 is used.

The remainder of this section details the implementation of these differences and the overall phrase discovery process for SFPD, since its use is fundamental to many techniques throughout the thesis. Also note that while in Chapter 4 SFPD is used mostly in conjunction with classifiers, and data that has been isolated based on the classifier's predicted classes, given its origin in explaining topics, it can easily be applied to aid feature discovery in the output of topic modelling, or clustering techniques.

Once surprising words are discovered, SFPD can be thought of *conceptually* as a three stage process. For each surprising word to be expanded to phrases:

1. Build a data structure to count the occurrences of phrases containing the word (Section 3.4.1).
2. Prune the data structure according to any thresholds or filtering criteria (Section 3.4.2).
3. Select the top N remaining phrases according to some sorting measure (Section 3.4.3).

In practice, the pruning can occur during the build process so that unnecessary ngrams are never explored and stored in memory. However, the three phases will be described separately in their respective subsection below.

3.4.1 Counting Phrases

The data structure used for counting phrase occurrences was inspired by the Trie (Fredkin, 1960). A Trie is a type of search tree, with current applications such as prefix-based search of strings. It is an efficient structure for lookup of prefixes, and storing elements with overlapping prefixes.

Figure 3.1 visualises a character-based Trie for representing strings (as is most common). Instead of storing words like “tea” and “ten” separately, the place in memory of their common prefix “te” is shared, by breaking down each word into its constituent characters. Therefore, the path from the root node to each other node represents a word. The number at each node (omitted if zero) specifies how many times the string represented by the path from the root was observed in the data. For example, Figure 3.1 claims that “tea” occurred once in the data. For example, Figure 3.1 claims that “tea” occurred once as a word, as did “team”, but “teal” occurred twice.

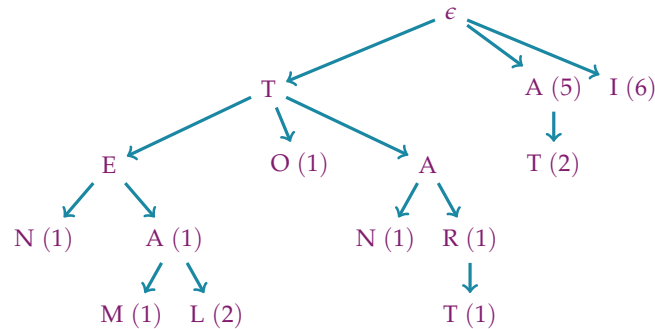


Figure 3.1: A character-based Trie representing the occurrences of the words: TEN, TEA, TEAM, TEAL, TO, TAN, TAR, TART, A, AT, I. The ϵ represents the empty string. The path from root to each node represents a string, and the number in brackets represents the number of occurrences of that path’s string in the data. This means that the L node shows that TEAL occurred twice.

Figure 3.2 shows a word-based interpretation of the Trie data structure. Unlike character-based Tries, where nonterminal nodes do not necessarily describe valid words, every node in word-based Tries will describe a valid ngram, where “valid” signifies that it occurred in the data. This is because if we observe the 4-gram “brown dog went home”, this means we also observed the trigram “brown dog went”, and so on.

The ngrams in Figure 3.2 are illustrative examples of contexts of the word “dog”. The diagram reveals the problem of analysing the contexts of “dog” with this basic word-based interpretation of the Trie: the “dog” node is not shared for any ngram that does not start with “dog”, making it difficult to perform frequency analysis of similar ngrams with “dog” appearing in various locations other than the start (hence why Tries are also referred to as *prefix trees*).

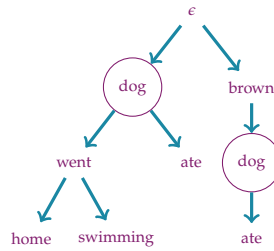


Figure 3.2: A word-based Trie of the ngrams “dog went home”, “dog went swimming”, “dog ate” and “brown dog ate”. The diagram reveals the problem of analysing the contexts of “dog” with a basic Trie: the “dog” node is not shared for any ngram that does not start with “dog”, making it difficult to perform frequency analysis of similar ngrams with “dog” in various locations.

The solution involves two factors. Firstly, we redefine an ngram by positioning the original word (keeping with the “dog” example) at the start, then following it with any terms from the original phrase that preceded “dog”, and only then follow with the terms that occurred after “dog”³. The preceding terms are placed in reverse order, maintaining the property that the closest terms in the ngram to the root (“dog”) are the closest children in the graph. Therefore, an ngram centered on “dog” such as “a smelling dog toy” would be reordered to “dog smelling a toy”. In order to encode the distinction between a transformed ngram like this one and an ngram that naturally occurred with this ordering, we must use *typed* edges between nodes. The basic Trie structure consists only of “forward” (or “downward”) edges, which encode the meaning that the parent node is followed by the child in an ngram. But now “reverse” (or “upward”) edges are introduced, which denote that the parent node is *preceded* by the child in the ngram.

The second factor of the solution must account for the fact that with the introduction of reverse edges, the Trie no longer accounts for all the sub-ngrams that we observe by virtue of observing a given ngram. In order to solve this problem, we must add a pointer to the path of forward edges for each step in the reverse path. Figure 3.3 illustrates how such a tree is built (henceforth named context tree). In particular, Figure 3.3d shows this replication of the forward path for each stage in the applicable reverse path. The replication permits us to represent the fact that observing “big brown dog ate” includes the observation of “dog ate”, “brown dog ate”, and “big brown dog ate”. Figure 3.4 further extends the example.

³ This ordering is a convention; we could have placed the following terms before the preceding ones.

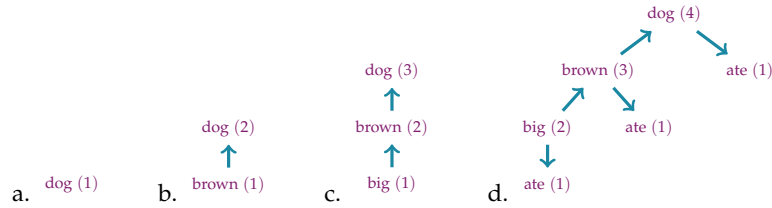


Figure 3.3: These diagrams show the construction of the context tree for “dog” as each new context is discovered. Assume that the following contexts concerning “dog” are discovered in the order given . *a*: dog. *b*: brown dog. *c*: big brown dog. *d*: big brown dog ate. Note that the one occurrence with the added “ate” must be attached to each relevant upward edge, because an occurrence of “big brown dog ate” also implies an occurrence of “dog ate” and “brown dog ate”.

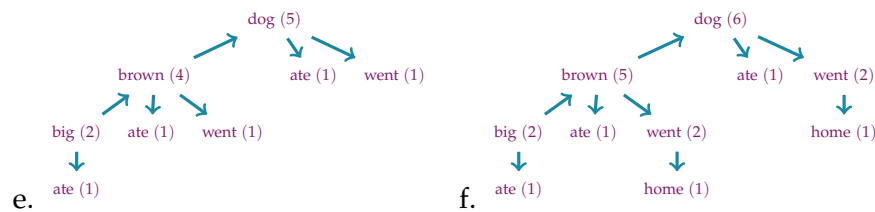


Figure 3.4: Adding the following contexts to the context tree in Figure 3.3d. *e*: brown dog went. *f*: brown dog went home.

Figure 3.5 provides a tip for reading paths in the trees.

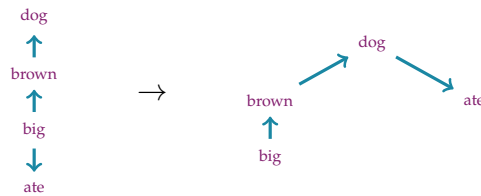


Figure 3.5: In Figure 3.4f, the longest path (and therefore phrase) represented by the node “ate” is “big brown dog ate”. One way to visualise this more easily is to identify the first downward edge, then imagine the downward sub-tree to be an immediate child of the root node (dog), placed on the right. The resulting graph can now be read left-to-right (following the arrows) like a normal phrase.

This concludes the description of the ngram-counting data structure.

3.4.2 Pruning Phrases

The next stage is to prune the trees in order to discard ngrams that are likely to be uninteresting. It is possible to prune during the con-

text tree creation stage by considering all ngrams of the same n before moving onto $n + 1$. This requires either maintaining a tokenised corpus, or re-tokenising each context of the root word, but saves building paths that would have been pruned earlier.

As with Baroni and Bernardini (2004), possible ngrams are pruned according to whether their frequency as a proportion of their parent $n - 1$ gram is above some threshold. This is essentially thresholding over the conditional probability of the added word, given that we've observed the $n - 1$ gram. If the probability is above some user-defined minimum, then the ngram remains unpruned. This threshold will be referred to as the *phrase pruning threshold*.

Additional methods of frequency-based pruning are considered in order to solve sparsity problems in Section 3.6.

3.4.3 Ranking Phrases

The final stage of SFPD is to select the top N phrases. Therefore, some measure of comparison is required for the nodes in the context tree. Figure 3.6 shows a more filled-out version of our context tree toy example. In order to illustrate the measure of phrase comparison simply, assume that the nodes in Figure 3.6 are present after pruning has already occurred. Once pruning is complete, all the phrases remaining are the equivalent (aside from differences in pruning strategy) of those that would be proposed by the system of Baroni and Bernardini (2004) if starting with the same root words. However, the aim of this work is to support analysis of corpora for the improvement of active learning, and to present sensible terms for adaptive querying, not to inundate the user with many phrases for every surprisingly frequent word. Therefore, instead of presenting all possible phrases, only the top N are presented (where N is a user-defined parameter).

The measure of comparison between possible ngrams must be valid across all ngram sizes. Larger ngrams are naturally more sparse, so a comparison based on raw frequency would be biased toward the shorter ngrams. But the frequency of ngrams intuitively has some bearing on how well it explains the surprisingness of the root word, since the cause of the surprisingness is presumably some context that seems to occur more often in this particular corpus, and the surrounding phrases represent this context.

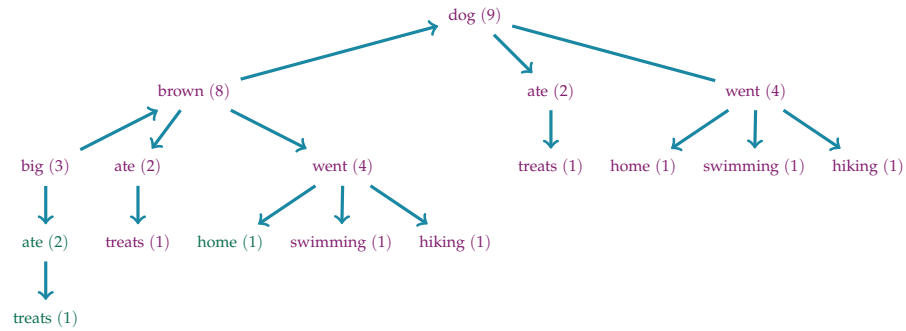


Figure 3.6: A context tree made by adding the following contexts to Figure 3.4f: “brown dog went swimming”, “brown dog went hiking”, and “big brown dog ate treats”.

Ngrams are made comparable here by considering their common sub-ngram, and inspecting the frequency of the sub-ngrams from this point that lead to the ngrams in question. For example, in Figure 3.6, the green-highlighted “treats” node represents the ngram “big brown dog ate treats” and the green-highlighted node “home” represents “brown dog went home”. Their common ngram is “brown dog”; this information is found by locating the nearest common ancestor node of the two nodes: “brown”.

At the nearest common ancestor, the divergent paths that lead to the descendent nodes “treats” and “home” are inspected. “Home” arises from the “went (4)” branch, and “treats” from the “big (3)” branch. These divergences will always be as comparable as we can get, since they represent one extra word from their common sub-ngram, so not only are we comparing same-size ngrams, but also ngrams of similar contexts. It is assumed that, because the “went” branch accounts for more occurrences of “brown dog” than the “big” branch and therefore better “explains” the occurrences of “brown dog”, its children inherit this property. Therefore, any child node of “went” will always be sorted above those of “big”.

This implies, for example, that the green-highlighted “ate (2)” (despite being the same size ngram and occurring more frequently) will be sorted below “home”.

If the diverging branches share the same frequency, then the branch that contains fewer stopwords is sorted above. If this value is also the same, then the phrase that has the most stopwords at its beginning and end is sorted below. Since Baroni and Bernardini (2004) excluded phrases based on the presence of stopwords, the intuition is that increased presence of stopwords reduces the likelihood that the phrases will be important, but it was hypothesised that there could be circum-

stances under which such phrases could be interesting. In the unlikely event that the ngrams are still equal according to these measures, then their ranking is arbitrary.

The notion of what constitutes a stopword from one domain to another can vary, and we cannot know what type of word usage is important to users ahead of time, so it is prudent to avoid completely throwing away potentially interesting phrases based on a fixed notion of non-relevant words. A client of `CASM CONSULTING` is currently interested in how industrial strategy is discussed online, and is using `SFPD` to investigate. They highlighted some phrases as interesting which began and/or ended with stopwords. In an informal discussion about their use of `SFPD`, it was suggested that this small additional context showed important information about the usage of terms. The usage information was often what raised interest in the term in the first place.

After initial experimentation with single large documents, and large sets of tweets, this phrase expansion strategy seemed to produce important phrases where possible. As very short documents, tweets present a few issues that need to be addressed for even basic corpus analysis. These are discussed in Section 3.5. Single large documents especially require some additional complexities at the pruning stage to avoid falsely detecting many terms as surprising. These complexities are discussed in Section 3.6.

Table 3.1 shows a selection of phrases from the top 50 that were generated from tweets collected on the day of the 2016 Brussels bombings (more details in Chapter 4), all of which reveal possible avenues of exploration in the data that an analyst might follow. For example, below they are categorised under classes of document which were found in the data for which they seemed to be relevant.

Condolences
condolences to the victims of the brussels
heartbreaking
thoughts and prayers are with the people of brussels
bourse square
pray for brussels
so sad to hear about brussels
solidarity with
Attack information and fallout
paris attacks suspect salah abdeslam
zaventem airport
explosions
maelbeek metro station
maalbeek metro station
key suspect in paris attacks captured in brussels raid
eurostar
jihadists
daesh
kalashnikov rifle
counterterrorism raid in brussels
airport several injured
salah abdeslam
unexploded suicide vest
eu foreign policy chief federica mogherini
tihange nuclear power plant
Related Topics
brexit
trump
nigel Farage

Table 3.1: Selection of phrases from the top 50 generated by [SFPD](#) from tweets collected on the day of the 2016 Brussels

3.5 TWEET ANALYSIS

The challenges in applying a measure of surprise to terms in a corpus of news articles are quite different to those in applying it to a corpus of tweets. This section examines the specific issues with applying [SFPD](#) to tweets.

In a collection of large text documents, a concern would be the possibility that a single document will use a new term enough that the document becomes the sole reason that a term occurs surprisingly frequently. This problem does not exist for collections of tweets. Tweets

are limited to 140 characters⁴, therefore no single tweet can have so many occurrences of a term that it would become an issue for any sensible sized tweet corpus.

The shortness of text presents a different problem however. No single tweet can be analysed in isolation, because so few words can occur in within the character limit that their rate of occurrence will often immediately seem surprising. If there are only 15 words in total in the tweet, then occurring three times would give a term a 20% rate of occurrence. If this rate is compared to its rate in a large reference corpus, such as a WIKIPEDIA article collection, a rate of 20% will most definitely be surprising, since it is unlikely that there is a single term that occurs in 20% of all WIKIPEDIA or any large reference corpus. In order to overcome this problem, the SFPD component incorporated into METHOD52 provides the user with three methods of joining the contents of tweets, which can easily be used individually or combined:

- Combine tweets that appear within a fixed time window.
- Combine the last N tweets.
- Combine tweets according to some logical criteria, such as a classification decision, or the presence of a key term.

If the corpus is constructed from too few documents, the overestimation of surprisingness is still likely to occur. Depending on the motivations of the research, integrating over *too many* tweets could hide important but temporally short-term features. Therefore, the method of tweet integration should be chosen carefully.

Which method is appropriate is task-dependent. If the TWITTER query concerns a topic that is trending on TWITTER, it is feasible to set only a fixed time window over which to integrate the tweets, since we can rely on the popularity of the topic to provide regular and large collections of tweets. The more popular the topic, the shorter the time window can be before the corpus acquired is too small.

When dealing with a non-trending topic, which perhaps only generates tweets in smaller chunks with much more irregularity, performing SFPD on a fixed time interval is not sensible; this would lead to irregularly sized corpora, which may be incredibly small. An altern-

⁴ After work was completed for this thesis, and during final drafting, TWITTER raised their character limit to 280 characters. This is still short enough that the following arguments should still apply.

ative is to wait to collect a fixed size corpus first, no matter how long it takes.

If the results of [SFPD](#) are required in real-time in order to update knowledge about the topic (e.g. in order to adapt the `TWITTER` query), then an additional time-based batching criteria should be specified. For example, wait until either 1000 tweets have been collected, *or* 30 minutes have passed before processing the batch. This is because *some* data could be better than no data if we're trying to follow topic drift. This increases the risk of introducing noise, since we may be using corpora of sizes smaller than desirable. This could necessitate stricter thresholds on the [SFPD](#) criteria for including phrases in order to counter-balance the additional noise.

Either instead of, or in addition to, the above, it is possible to combine tweets based on some aspect of the tweet itself. For example, [Duan et al. \(2012\)](#) group tweets together that share hashtags or `URLS`. Alternatively, tweets can be grouped by some inferred label, such as the outcome of a relevancy classifier (see Section 3.8). Again, careful attention should be paid to ensure that the resulting sets of tweets are not too small.

Lastly, it is of course possible to abandon the real-time nature completely, where we collect a fixed set of tweets and combine all into a single corpus. This is the approach used by the `CASM CONSULTING` client investigating industrial strategy discourse.

3.6 SINGLE LARGE DOCUMENT ANALYSIS

While the main focus of this work is social media analysis, and in particular `TWITTER`, the core technique should be adaptable to different scenarios. A very different scenario compared to tweet analysis is the analysis of single large documents, instead of corpora constructed from a multitude of documents. This section details a case study in which [SFPD](#) was applied to single large documents, and describes how issues were dealt with to adapt it.

The [SFPD](#) method was applied to funding bid documents (using a large `WIKIPEDIA` sample as the reference corpus) in order to generate search engine queries that would acquire related work from the web⁵. Potential goals included:

⁵ Experimental work undertaken in collaboration with Jack Pay, research fellow in the Text Analytics Group at the University of Sussex.

- Ascertain whether the proposed research was too similar to existing work or had its own niche.
- Find related work that should be considered and studied when undertaking the proposed research.

The individual surprising words that were produced were largely sensible; they were words taken from the names of related fields, techniques, or jargon. However, when [SFPD](#) attempted to expand them to phrases, there seemed to be no setting of the phrase pruning threshold that produced desirable results. Either it was too relaxed, and the system would output long, overly inclusive phrases, or it was too strict, and no phrases occurred frequently enough to warrant expansion much beyond the original unigram.

When inspecting the over-long phrases, a source of the problem became clear: with only a single, fairly large document, despite its theme being more coherent so it need not be as large as the tweet pseudo-documents, the frequencies of the surprising phrases were fairly low. High enough that the surprising words were discovered, but low enough that even a fairly low-count phrase could account for most of the unigram's occurrences, leading to over-generation of long phrases. For example, if a word occurs five times in the document, and a bigram including this word occurs twice, then this bigram already accounts for 40% of all of the occurrences of the term. [TWITTER](#) experiments (e.g. the industrial strategy work, or Brussels study in [Section 4.2](#)) on the other hand show that 30% is a fairly conservative phrase pruning threshold.

The simplest response is to set a minimum frequency required of a phrase in order for it to be a candidate; we could ignore all phrases that had not occurred more than twice. But there is the worry with this approach that any number chosen slightly too high would lead to ignoring important phrases surrounding infrequent terms.

A slightly more permissive approach is to adapt the phrase pruning threshold based on the number of unique phrases of a fixed size that a term occurs in. A phrase that occurs four times, whose root word occurs only eight times, is more interesting if there are four other phrases each only occurring once which the root word occurs in. But if there are only two possible phrases each occurring four times, then neither phrase seems as special. A simple method for encoding this dynamic threshold is shown below:

$$t_d = \max\left(\frac{1}{C}, t_u\right) \quad (3.2)$$

Where t_u is the user-defined phrase pruning threshold, and C is the number of phrases being considered to extend the smaller phrase, i.e. the total number of children of a given node in the context tree. This formulation ensures that the threshold is higher when the number of context tree children is smaller, and lower when there are more choices for expanding a phrase. This property achieves the goal of making it harder for a potential phrase expansion to be considered sufficiently interesting when there are fewer alternative expansions in the dataset. The *max* function ensures that the threshold is never reduced below the user-defined threshold.

The drawback of this approach is that it does not account for absolute frequency. Regardless of the number of occurrences of a phrase represented by a node in the context tree, it is only the number of different sibling nodes that influence the threshold. However, if a phrase occurred twice, and it has only two child nodes each occurring once, then the threshold is 0.5. And the same is true if the phrase occurred a thousand times if there are only two child nodes, each occurring 500 times. Intuitively, it seems as though one occurrence out of two, is far less reliable an indicator that the phrase should be expanded.

This can be adjusted by having a number of tiered thresholds on the absolute frequency of the phrase. In the equation below t_1, t_2, t_3 are user defined positive integers, which represent the term frequency at which a more conservative threshold will be chosen for the phrase pruning threshold. Otherwise, the dynamic threshold described above comes into effect.

$$t_d = \begin{cases} 1 & \text{if } F < t_1 \\ 0.75 & \text{if } t_1 \leq F < t_2 \\ 0.5 & \text{if } t_2 \leq F < t_3 \\ \max\left(\frac{1}{C}, t_u\right) & \text{otherwise} \end{cases} \quad (3.3)$$

Where F is the total number of occurrences of the phrase or term that is under consideration for expansion. In other words, when a child node in the context tree is being considered as an expansion, F is the frequency of the parent node.

Experiments on the single large documents showed that terms were less likely to have been expanded too far into phrases when these

thresholds were set. This means we can relax (lower) the user-defined minimum phrase pruning threshold t_u ; we no longer have to keep it high to filter out the noisy features, since they are pruned using the new dynamic threshold.

While the phrases produced with the new threshold were desirable according to human judgement, the study's approach of querying the search engine BING⁶ mostly turned up well-known academic work with which researchers established in the field would already have been acquainted. The results would probably be more useful to new researchers looking to fill out their knowledge of a field. This is likely due to the nature of a search engine, since a highly cited and known paper is more likely to arise in search results than a less popular but related paper. And companies offering commercial products are much more incentivised to ensure that their products appear in search results, but they only constitute noise in this scenario.

3.7 ESTABLISHING THE EXPECTATION

Established earlier was the idea that in order to measure a notion of *surprise*, we must first establish an *expectation*, we can then be surprised when our expectation is contradicted. To find just how surprisingly frequently a word occurs, we need to have established an expected frequency for that word. How much more frequently the word occurs than our expectation determines how surprised we are at its frequency in a dataset. The expectation is defined by taking the frequencies of words in a reference/background corpus.

Given that the notion of surprisingness is relative to the background corpus, it follows that the selection of a background corpus is important. It is also problem-specific, because what the analyst wishes the system to find surprising (and therefore important) is problem-specific. This section explains some important basics for reference corpus selection, but more detailed discussions are found in later sections (4.4.3 & 5.9) in the contexts of different techniques employing SFPD.

As a default, the SFPD tool incorporated into METHOD52 uses a reference corpus drawn from English WIKIPEDIA⁷. This represents a broad and general sample of the English language, enabling the SFPD to determine whether, in a given target dataset, any features occur

⁶ <http://www.bing.com>

⁷ <http://wikipedia.org>

proportionally more than in a general sample of English. However, this reference corpus can be swapped out by any dataset, making the notion of surprisingness a more specific comparison than just with general English, or even with a different language entirely.

In `METHOD52` this is achieved simply by connecting another data-source component to the `SFPD` component, and labelling its output with a background data annotation as shown in Figure 3.7.

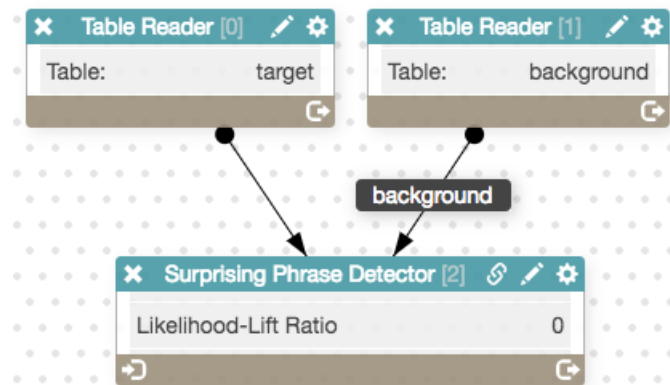
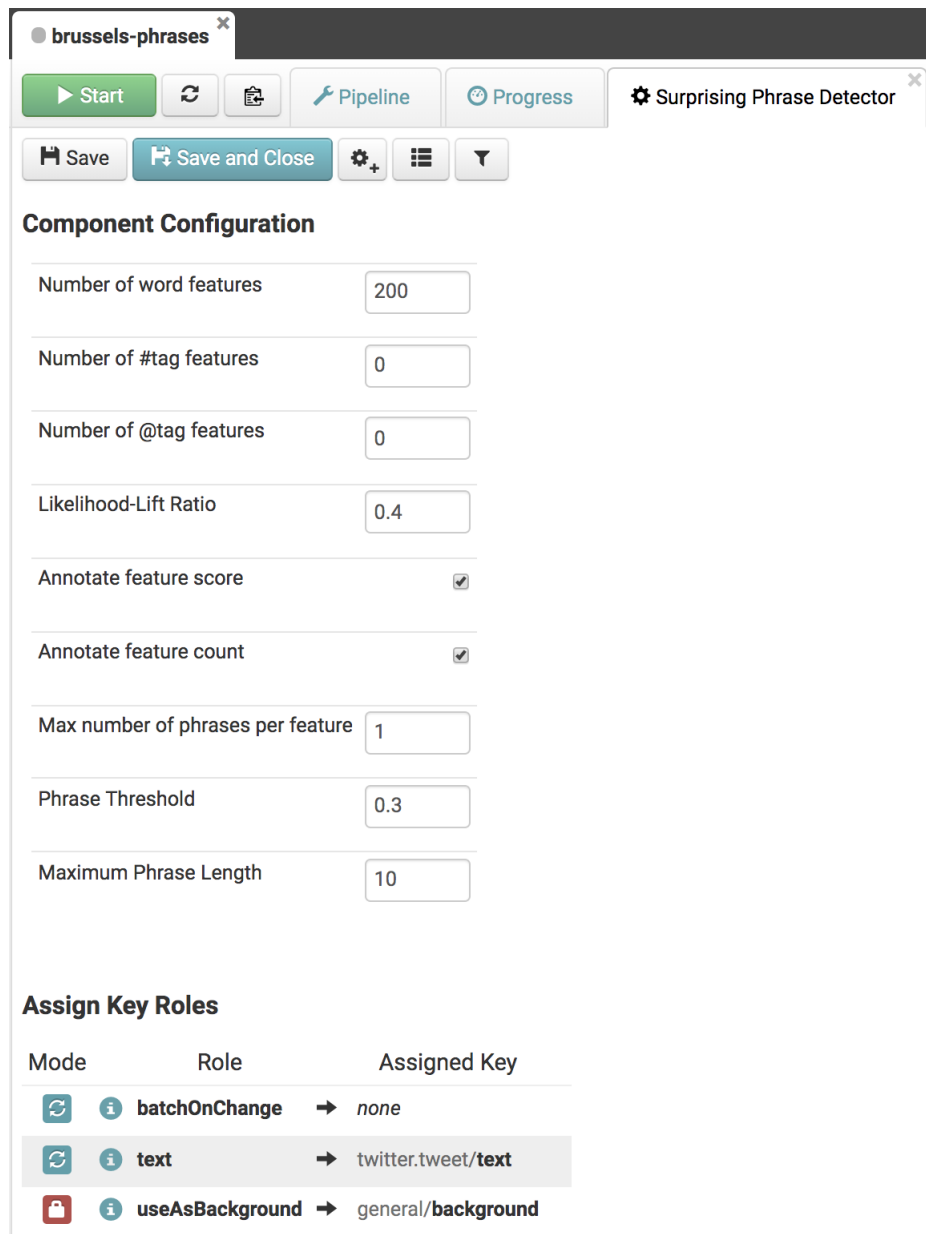


Figure 3.7: Pipeline screenshot showing annotation of the “background” label, so that the `SFPD` component knows to treat data from the `TABLE READER` on the right as background data (the reference corpus).



Component Configuration

Number of word features: 200

Number of #tag features: 0

Number of @tag features: 0

Likelihood-Lift Ratio: 0.4

Annotate feature score: ☒

Annotate feature count: ☒

Max number of phrases per feature: 1

Phrase Threshold: 0.3

Maximum Phrase Length: 10

Assign Key Roles

Mode	Role	Assigned Key
	batchOnChange	none
	text	twitter.tweet/text
	useAsBackground	general/background

Figure 3.8: When the cog icon of a component is clicked, the component configuration screen appears. The basic settings of the [SFPD](#) component are shown here. More frequency filtering options are available if the advanced options button is pressed (cog+ icon). Note the selection of `TWITTER` text for analysis and any documents with the “general/background” annotation will be considered background documents.

While the default `WIKIPEDIA` reference dataset can be sufficient for finding interesting terms in a `TWITTER` dataset (see Section 4.2), if a corpus is particularly full of chatty language, or anything that is just an artefact of the documents being `TWITTER` text, the reference may need to be more specialised, because the surprising phrases

produced become filled with colloquial and chat terms, which occur surprisingly frequently when compared to WIKIPEDIA text.

If the goal is simply to decrease the level of surprisingness of TWITTER language, the simplest solution is to use the TWITTER sample API to generate a reference corpus. The sample API allows us to stream a 1% sample of all messages being tweeted, without having to specify a query.

Beware that even a broad 1% sample can be biased enough to not be useful as a reference corpus if not taken correctly. For example, if there is a very popular trending topic on TWITTER on a given day, and the sample is taken across only that day, then the reference corpus language will be dominated by the terms that describe the trending topic. In turn, this will down-weight the surprisingness of those terms found in any target corpus. This problem can be avoided by taking a sample of tweets over a longer time period, thus minimising the effect of any particular trending topic. Section 4.4.3 discusses this type of domain adaptation in the context of improving the active learning process of classifier training.

SFPD can be adapted to other languages by simply changing the expectation. If we were to maintain the English WIKIPEDIA as our reference, then most words in an Arabic language corpus would seem to occur surprisingly frequently, since our expectation is defined by having seen predominantly English words. Instead, an Arabic corpus can be uploaded to METHOD52, or collected from social media, to be used as the reference corpus, the definition of our expectation. This method was applied using METHOD52 by Conway et al. (2017). Section 5.9 discusses this and other domain adaptation issues in the context of adaptive streaming.

3.8 USING EXISTING METHOD52 PIPELINE ARCHITECTURE

By incorporating SFPD into the existing METHOD52 architecture, the SFPD analysis can be used very creatively. Many of the options are explored in the context of active learning in Chapter 4, and adaptive streaming in Chapter 5. Therefore, below follows a high-level summary of SFPD's utility in the context of existing functionality of METHOD52.

Social media data can be tracked in real time. Wibberley et al. (2014) show that it is often necessary to analyse sudden spikes in the volume of tweets in isolation, since they are often reactions to real-world

events and signify the introduction of different vocabulary or vocabulary use. METHOD52's volume-over-time component generates an interactive graph displaying the tweet volume over time (as shown in Figure 3.9). Spikes in volume can be identified, and the data corresponding to the spike can be extracted. The spike data can be made to constitute the target data for SFPD, so that the phrase extraction is more focused on the language in the high-volume data. The reference corpus can be changed from WIKIPEDIA to all collected tweets in the set, in order to focus the expectation on what about the spike data is surprising compared to whatever background tweet collection is being monitored.

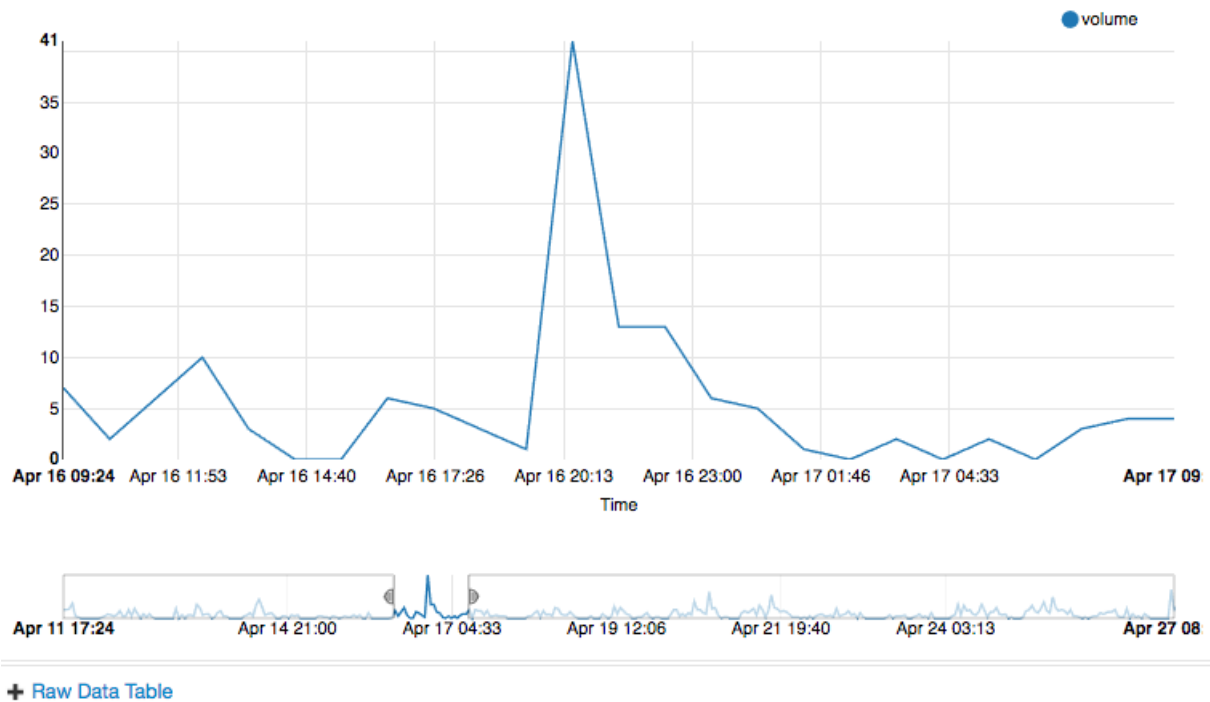


Figure 3.9: User interface for volume over time analysis in METHOD52. The lower graph is a selection tool. It shows the volume-over-time plot for all documents passed into the component, and the user can select portions of it for closer inspection. The selected portion above is a spike of particularly high volume compared to the rest of the data. The user can extract all documents within a selected portion by using the “Raw Data Table” button.

SFPD can also be used in conjunction with the user-trained classifiers. Classifiers are used to split up and filter data. These splits can be used as another method to focus the target data and find more specific surprising phrases. For example, the target data can be set as only those documents that are classified as relevant, or as a given

topic or sentiment. The use of [SFPD](#) in coordination with `METHOD52` classifiers is fully explored in Section 4.4.

Documents can be filtered by the presence or absence of specific phrases using the keyword annotator component. If the right phrases can be identified, this can be a useful tool to increase precision or recall for tasks where classifiers alone struggle to learn the necessary distinctions.

[SFPD](#)-identified phrases have been successfully used to filter and sort documents in projects. `CASM CONSULTING` investigated social media use by, and concerning, the police as a contribution to the 2016 Police Effectiveness, Efficiency and Legitimacy ([PEEL](#)) programme's national overview report⁸. One of the tasks was to identify user criticism/support of the police in various topic areas, such as racism, corruption, and communication strategy. When identifying tweets about racism of the police, general keyword filtering using terms like "racism" and "racist" produced very noisy results; the tweets were frequently not about the police or only discussing racism in the abstract. Very specific terms like "racist police officer" produced suitable data, but in very small quantities. It proved difficult to train a classifier to make the distinction because of the sheer amount of noise.

Instead, [SFPD](#) was applied in two ways. Firstly, all collected data was used as target data for [SFPD](#). The resulting phrases were inspected for candidates to grow the list of high-precision specific terms, in order to increase recall there. Next, the data was filtered keeping only those documents using general terms like "racism" to produce the target data. The resulting phrases were inspected for terms that were indicative of *noisy/irrelevant* data. Then any tweet that mentioned these noisy terms can be filtered out before reaching the classifier. For example, it was determined using [SFPD](#) that the racism of "tories" was frequently discussed in the data, which was irrelevant for the purposes of the project. With enough such examples, noisy data can be filtered out using these discovered phrases. Therefore, it is possible to retain the high-recall low-precision terms like "racism", but maintain higher precision by then filtering out the specifically noisy terms. The `METHOD52` job using the result of this process is shown in Figure 3.10.

`METHOD52` provides geolocation capability, which can be used to isolate target data to tweets originating from a common location. And tweets can easily be grouped by the presence of particular terms

⁸ <https://www.justiceinspectorates.gov.uk/hmicfrs/publications/peel-police-effectiveness-2016/>

in their text. Documents from social media especially can also be grouped by the users that publish them.

Combinations of all of the above permit a flexible way to drill deeper into the data, each time selecting surprising features according to some expectation and target criteria, until the user has a good understanding of the dataset.

Any of these methodologies can lead to the discovery of terms which the analyst might wish to use as a query for collecting more data. It is difficult, if not impossible, to predict all of the terms in use on social media which are relevant to a study. Unfortunately, if this process is entirely manual, the phrases will rarely be discovered in time to affect the project's data collection strategy. This motivates the need for a semi-automatic adaptive querying process, which is the purpose of Chapter 5.

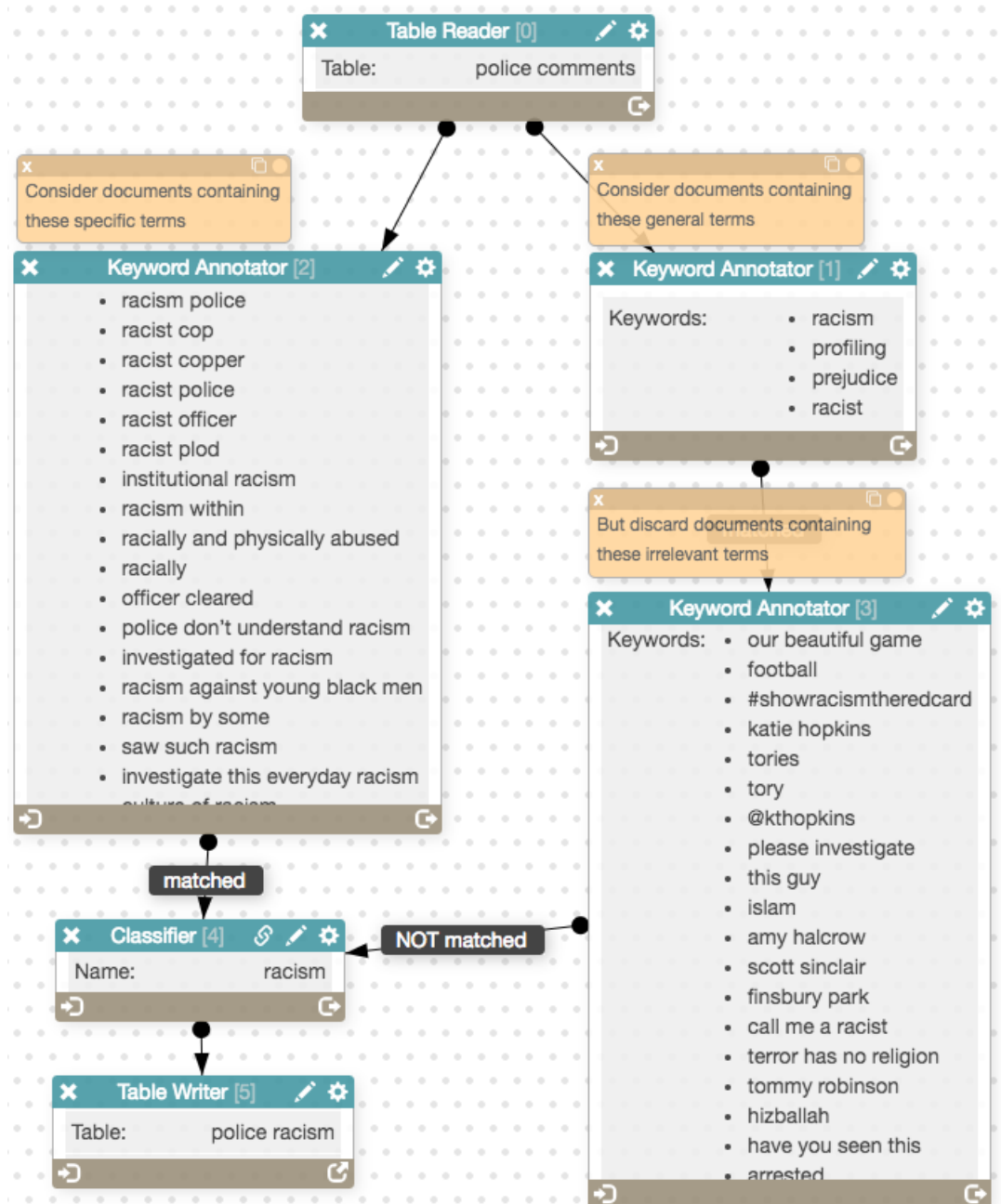


Figure 3.10: Screenshot of the pipeline which finds documents that are discussing police racism. Tweets mentioning police follow two parallel processes. On the left, specific terms that frequently seemed to be indicative of tweets discussing police racism are used to identify documents that are then passed to the classifier. On the right, general noisy terms are used to identify candidate documents, but they are filtered first by the presence of terms that were often indicative of irrelevant documents.

3.9 SFPD COMPARISON WITH CHI-SQUARE AND LOG-LIKELIHOOD RATIO

This section shows how SFPD compares to two popular methods of key term extraction: the χ^2 and the Log–Likelihood (LL) ratio measures. These methods do not inherently involve methods of phrase expansion; therefore, they compare more directly to the initial surprising word detection step (using the relevance measure of Sievert and Shirley (2014)). So to aid comparability to the overall SFPD approach, the same phrase expansion method is applied to words proposed by the χ^2 and LL methods (the originally proposed unigram words will also be shown).

The analysis of the Brussels bombings data resulted in a subset of tweets characterised by being personal reactions or opinions concerning the attacks. This subset was further divided by keywords to produce subsets about Barack Obama (then president of the us), Donald Trump and Islam, because these were discovered as major topics which were very polarising of TWITTER users. The comparison in this section utilises the set of tweets mentioning Donald Trump, which after de-duplication contained 28,905 tweets. The reference corpus was a 100K random sample of all the collected Brussels tweets.

Tables 3.2, 3.3, & 3.4 below show the top 30 words (and their top phrase expansions) for the LL, χ^2 , and SFPD methods respectively. The SFPD table shows the results with the likelihood-lift parameter set to 0.5. All of the following values were tried and 0.5 seemed to provide particularly interesting results: 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9. Figure 3.11 shows a screenshot of what the user sees when they inspect a database table containing the phrases output by the SFPD component.

In each table for each measure, the SFPD phrase expansion has clear utility in terms of contextualisation of what makes certain words interesting. For example, both LL and SFPD discover “vote”, and the phrase expansion tells us that it’s about voting for Donald Trump as president. It explains why the generic term “mr” is suggested by LL (and SFPD with likelihood-lifts 0.6, 0.7, & 0.8), and we find that “mr trump” is a common way of referring to Donald Trump by his supporters (and therefore a useful bigram for a feature labelling when training an anti/pro Trump classifier). Other seemingly generic terms like “blames” and “incites” receive contexts that clarify their meaning. Names are often expanded to their full form or even context

(e.g. Megyn to “Megyn Kelly”, or Shepard to “Shepard Smith slaps down Donald Trump”). However, some such as “Newt” are missed because “Newt Gingrich” is only used in full form four times in the dataset. The more frequent and generic terms are less likely to be expanded since they will have no clear top phrase in usage (e.g. “great” or “yes”).

The χ^2 measure seems to produce many high-frequency overly generic terms, such as swear words, text speak (e.g. “b”), and interjections, which are very unlikely in most circumstances to be useful in an analysis of the tweet content. The phrase expansion method also cannot produce useful top phrases for these terms since they are used in such many, varied ways. SFPD is more likely to produce these kinds of terms as the likelihood-lift parameter is closer to 1, since this assigns more weight to just the likelihood of a word in the target data. Although, for this dataset, for each of the top 30’s of SFPD with likelihood-lifts ranging from 0 to 0.9, the swear words and “yeah” variants still did not occur. In fact, the only overlapping words with the top 30 χ^2 across these ten trials were: “president”, “think”, and “said”. However, there were similar variants in similar contexts for some words, e.g. “blames” present in one and “blaming” present in the other. The main interesting term proposed by χ^2 in the top 30 that *wasn’t* proposed by SFPD (or LL) was “build a wall”, which refers to discussion of Donald Trump’s proposed border wall.

Due to χ^2 favouring overly generic terms, its top 30 missed many interesting terms that the other two measures discovered. LL and SFPD can have much more overlap. For example, with likelihood-lift set to 0.7, SFPD discovers all but eight of the terms discovered by LL’s top 30. These missing eight are: “talking”, “january”, “leader”, “look”, “nato”, “ago”, “u”, and “wrong”. Two of which are particularly interesting: “january” hints at discussion surrounding whether Donald Trump predicted the Brussels attacks earlier in January, and “nato” points to discussion about his suggestion to disband NATO. “Wrong” is frequently used when arguing whether Trump is wrong about something, but is becoming quite generic, as are the other terms.

SFPD is more likely to produce overly generic terms as the lift-likelihood is increased. For example, when maximising overlap with LL at 0.7 we see “said”, “think”, “like”, “stop”, “want” and “saying”, which seem to reflect no specific theme in the data. But by setting the value lower at 0.5 as shown in Table 3.4, we can produce a good num-

ber of more specific, interesting terms that [LL](#) does not find, which index into interesting discussion. For example:

incites hatred & violence & banning all discussion about Trump's attitude toward Muslims inciting hatred.

advisors calls to pay more attention to expert advisors.

a fear monger criticism of Trump

airtime Trump's critics suggest that he is given too much airtime for his sentiments

prescient some of his supporters believe Trump to have been prescient regarding the attacks

doing the tango users show support for Trump while criticising President Barack Obama for being pictured dancing the Tango soon after the Brussels attacks

djt "Donald J Trump" a way of referring to Donald Trump that was actually missed when originally dividing the dataset by keyword.

general/phrase
trump
doing the tango
hilary
donald trump
incites hatred&violence&banning all
trumps
romney
capitalize on
airtime
an apology
shep smith
shepard smith slaps down donald trump from brussels : no
vote trump
trump2016
features donald trump trashing brussels
maria
djt

Figure 3.11: This is the screen that users see when the click on the output table of an [SFPD](#) job. Any further annotations the user makes on the phrases would appear as additional columns.

word	phrase expansion
trump	trump
donald	donald trump
vote	vote trump
cruz	cruz
right	right
like	like
need	we need
america	america
president	president
trumps	trumps
blaming	blaming trump for brussels
mr	mr trump
ted	ted cruz
blame	to blame
hell	hell hole
said	said
stop	stop
want	want to
usa	usa
saying	saying
leader	leader
look	look at
think	think
nato	nato
muslims	muslims
u	u
january	in january
ago	months ago
wrong	wrong
talking	talking about

Table 3.2: Top 30 words proposed by the log-likelihood ratio method from a dataset of personal reactions to the Brussels bombings, concerning Donald Trump. The reference dataset is a sample of all tweets collected containing the word “brussels”. Words are expanded using the phrase expansion method described in Section 3.4.

word	phrase expansion
f***ing	f***ing
gonna	gonna
having	having
b	b
crazy	crazy
yeah	yeah
thanks	thanks
country	our country
wall	build a wall
f***	f***
think	think
dangerous	dangerous
blames	blames trump for brussels
happen	happen
needs	needs to
start	start
ask	ask
worst	brought out the worst in cruz and trump
yes	yes
damn	damn
instead	instead of
worse	worse
comment	comment on
great	great
said	said
happens	what happens when you
president	president
responds	responds to brussels attack by insulting the 'city' of
things	things
beginning	just the beginning

Table 3.3: Top 30 words proposed by the χ^2 method from a dataset of personal reactions to the Brussels bombings, concerning Donald Trump. The reference dataset is a sample of all tweets collected containing the word “brussels”. Words are expanded using the phrase expansion method described in Section 3.4. Asterisks not present in original tweets.

word	phrases
trump	trump
tango	doing the tango
hilary	hilary
donald	donald trump
incites	incites hatred&violence&banning all
trumps	trumps
romney	romney
capitalize	capitalize on
airtime	airtime
apology	an apology
shep	shep smith
shepard	shepard smith slaps down donald trump from brussels : no
vote	vote trump
trump2016	trump2016
features	features donald trump trashing brussels
maria	maria
djt	djt
spew	to spew
newt	newt
cruz	cruz
megyn	megyn kelly
prescient	prescient
advisors	advisors
stump	stump
monger	a fear monger
dishonest	dishonest
slaps	smith slaps down donald
right	right
electing	electing
brussels	brussels

Table 3.4: Top 30 words proposed by the [SFPD](#) method (likelihood-lift: 0.5) from a dataset of personal reactions to the Brussels bombings, concerning Donald Trump. The reference dataset is a sample of all tweets collected containing the word “brussels”. Words are expanded using the phrase expansion method described in Section [3.4](#).

3.10 CORPUS HOMOGENEITY

The homogeneity of a corpus is the property of how consistent its vocabulary is. The more sections of a corpus that present very different statistics over their vocabulary when compared to the corpus as a whole, the more heterogeneous the corpus is.

Kilgarriff and Rose (1998) argue that the homogeneity of a corpus is an important property to be known when comparing the similarity of corpora, because it would be unclear how to interpret any differences when comparing to a heterogeneous corpus. SFPD entails the comparison of the target and reference corpora, so this section discusses the relevance of corpus homogeneity.

The issue is most salient when comparing corpora of similar size for their differences. If they are similar in size, then large heterogeneous sections could exaggerate differences, or make it seem like there are similarities that do not truly apply to the rest of the corpus. When comparing to a large normative reference corpus, we expect the corpus to be a broad sample, heterogeneous by its very nature, expecting that the target corpus only differs in its parts that diverge from the norm.

For our use of SFPD, however, we are less interested in finding a direct measure of similarity between corpora, and less interested in finding some set of terms that completely and accurately describes all and only what makes a corpus interesting. The focus is instead on aiding user exploration. The user is expected to interact with and explore the data, using the discovered features as a guide for how to define the semantics of classes of documents, or understand a class of documents generated by the machine. Therefore, there is no current focus on providing this feature for the user. However, it is possible to achieve the method described by Kilgarriff and Rose (1998) using existing features of METHOD52 and a spreadsheet program. The necessary steps are outlined below:

1. Divide the corpus into slices. METHOD52 has a batching component which can split documents into several of the same size.
2. Create two separate corpora by randomly allocating the slices into two documents. There is a METHOD52 component which annotates a random number, which can be used as a basis for filtering documents.

3. Compare the resulting corpora by comparing the most frequent terms of each constructed corpus to each other. These features can be extracted using the [SFPD](#) component with likelihood-lift $\lambda = 1$ (features are then ranked solely by their likelihood). The comparison could be done by inspection, or like [Kilgarriff and Rose \(1998\)](#), use a measure such as the Spearman's rank correlation coefficient or χ^2 to quantify the difference between the top features of each corpus. `METHOD52` can export the features and their counts as a csv file for any program designed to import csv and calculate these statistics.
4. Investigate multiple random allocations of the document slices. From these tests, if a quantitative measure is used, an average and standard deviation can be derived. The more overlap discovered in these comparisons, the more homogeneous the corpus.

3.11 CHAPTER SUMMARY

This chapter proposed a new method for key phrase extraction: Surprisingly Frequent Phrase Detection ([SFPD](#)). [SFPD](#) was introduced to tackle the research question: *how can features be identified which provide useful bases for characterising classes of text?* Furthermore, it serves as a fundamental technique for later chapters.

- Previous keyword extraction work was examined.
- The evaluation strategy for the techniques introduced in this thesis was discussed. With some exceptions that will be made clear, evaluation will primarily be driven by theoretical argument and presenting actual exemplar output, which will be supported by informal/anecdotal discussion with project leads about the experiences of `METHOD52` analysts.
- [SFPD](#) uses a measure of keyword relevance which includes a term for keyword likelihood and lift, and through adjustment of the likelihood-lift ratio parameter λ , either can be emphasised.
- [SFPD](#) includes a method for expanding keywords to multi-word phrases. The frequencies of ngrams containing proposed keywords are analysed to find phrases that most explain the occurrence of the keywords.

- The first hurdle with applying this method to tweets is the shortness of documents. Instead, tweets are analysed together as large corpora. Methods for batching tweets together were described.
- When analysing single large documents, low frequency phrases can be key phrases. Dynamic phrase pruning thresholding was introduced as a method to consider both absolute frequency and the number of choices for keyword expansion when finding key phrases.
- When determining the surprisingness of a word, there must be some expectation of how often the word should have occurred. This expectation is defined using a reference corpus. A reference corpus can be creatively chosen to emphasise different types of word, instead of a fixed user-defined threshold.
- An overview of the benefits for embedding [SFPD](#) in the existing `METHOD52` platform is given, which demonstrates the variety of strategies that could include [SFPD](#) analysis.
- [SFPD](#) was compared to two popular methods of keyword extraction: χ^2 and Log–Likelihood ([LL](#)) ratio.
- Corpus homogeneity arose as an important issue in corpus comparison studies. A method for determining homogeneity is given, but is argued to be less important for this work.

Chapter 2 showed that the discovery and isolation of classes of document are important aspects of the engagement that analysts undertake when using `METHOD52`. This thesis considers the research question: *how can feature discovery be used to support the characterisation of semantically coherent classes of text?* This is a bottom-up approach toward supporting this type of engagement, which focuses on the use of word and phrase features, which is why the previous chapter focused first on establishing a method of phrase discovery, which can be drawn upon when needed (in this chapter, Section 4.4). This chapter now deals with the next sub-part of the research question, which was shown in the introduction and is the core of the overarching question:

How can feature discovery support the identification, definition, and characterisation of classes of text?

As previously suggested, bespoke classifiers are a core part of the explore-search methodology of discovering and isolating classes. Their very purpose is to divide the data into user-defined classes, and therefore worth focusing on. The `METHOD52` classifier component was originally an extension of `DUALIST`. [Settles \(2011\)](#) built `DUALIST` to take explicit feedback from the user regarding the indicativeness of individual features for a given class, and demonstrated the approach to be effective. Therefore, it even shares aspects of the bottom-up feature-driven approach.

As an environment for training bespoke classifiers, `METHOD52`'s classifier component primarily tackles the “search” methodology of the explore-search cycle: the process of isolating a given class of data. The user trains the classifier to label documents with a class whose semantics the user knows, and is attempting to train the classifier to model. The process of presenting documents and features to the user through active learning does contribute to the exploration step though by exposing the user to the data and its features, much as would be expected in an inspect step (it could even be thought of as a nested explore-search cycle, as mentioned previously).

Firstly, due to its importance in this research, the current active learning classifier training process in `METHOD52` (originally based

on DUALIST (Settles, 2011)) is described in detail in Section 4.1 for context. Then Sections 4.2 & 4.3 discuss datasets that will serve as the primary motivators for the techniques described in this chapter.

For this chapter’s contribution, Sections 4.4, 4.5, 4.6, & 4.7 analyse weaknesses of the METHOD52 approach and how feature discovery methods can better support characterisation of classes of text.

In practice, the existing feature discovery mechanisms for analysis by classifier present a number of problems. In summary:

1. High Information Gain (IG) features are presented to the user according to the current classifier model. Since the current model is usually still undergoing improvement, these features are not always useful choices. Furthermore, classification in METHOD52 is an *exploration*: the user does not know that a certain classification scheme is possible to learn on a dataset until they have tried it. Therefore, feature discovery which is entirely dependent on a possibly flawed classification model is not a complete/best solution. This exploratory process of training a classifier, determining that it is flawed and then discarding it can be extremely time-consuming.
2. It can be difficult to interpret the meaning of isolated words, so it can be a struggle to label any as indicative of a particular classification. And worse, user interpretation of features may not reflect reality, so their labelling leads to worse performance. Suggested features may also not encode enough context to distinguish them from uses that are indicative of other categories, so even if interpreted correctly, the user may be unable to label it for a given category without harming performance on other categories.
3. The method for incorporating the knowledge generated from user-labelled features is very simple. A fixed number of “hallucinated” counts is added as evidence for a feature appearing in documents under the selected classification. This simplicity can be inappropriate, since features are rarely equally indicative of a classification. This especially leads to probabilistic problems when adding evidence for rarely occurring terms.

Feature discovery strategies are therefore approached from three broad perspectives throughout the remainder of the chapter:

Feature exploration Provide various views on possibly important features, especially views which force the user to confront their assumptions and examine how their classifier is splitting the data. Also provide views on the data that speed the process of classifier building by finding enough information to reduce the number of failed classifiers (Section 4.4).

Feature extraction Improve the actual form of features that are extracted for classification and their display to the user (Sections 4.5 & 4.6).

Feature incorporation Improve the way that hand-annotated features are incorporated into the classifier model (Section 4.7).

The “feature exploration” strategies are most aligned with the “explore” methodology, and the feature extraction and incorporation discussions are mostly aligned with the “search” methodology, since their aim is to improve the active learning system’s ability to isolate classes.

4.1 CLASSIFICATION IN METHOD52

The active learning environment of METHOD52’s classifier component is the main subject of improvement in this chapter. Therefore, for context purposes, this section describes in detail METHOD52’s current classifier training framework (Wibberley et al., 2013, 2014) and its DUALIST foundations (Settles, 2011; Settles and Zhu, 2012).

4.1.1 *Analysis by Classifier*

Building a pipeline of METHOD52 classifiers to perform some bespoke analysis is an iterative process. The high-level procedure is depicted in Figure 4.1, and is explained in this section with references to the numbered steps in the flowchart.

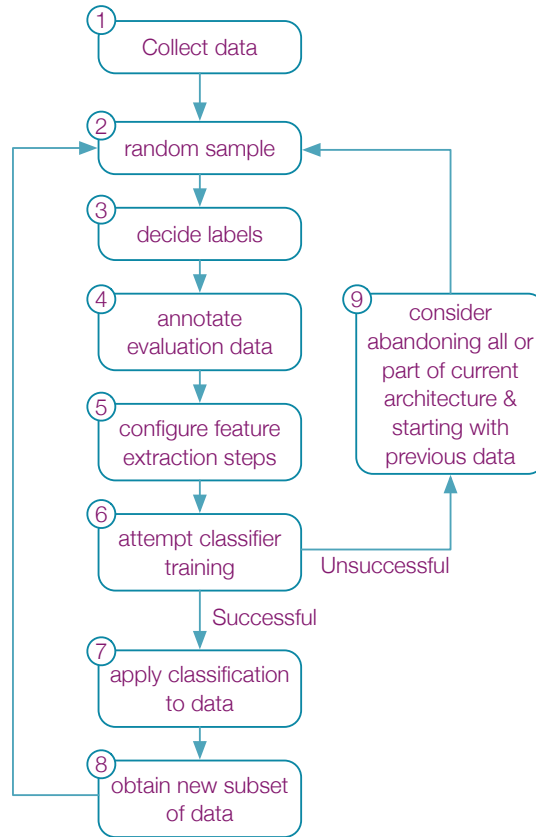


Figure 4.1: The iterative process of defining classification tasks, training classifiers which produce new views on the data, and repeating on the newly generated subsets of documents.

Given that some text data has been collected (1), METHOD52 provides the user with a random sample of documents from the dataset (2). The user should familiarise themselves with the data by reading the documents in the sample. From this reading, the user must decide on some initial idea for the classification labels that the classifier will attempt to assign over the documents (3). This process of random sampling is a small step in the direction of supporting more of the exploration methodology, since before requiring the user to decide on their classes of interest, they are prompted to inspect the data for classes which the data may best support.

Once classification labels are decided, the user must begin annotation of the random sample with these labels, since these documents will serve as an evaluation set for the classifier (4). When the evaluation set is labelled, classifier training can begin.

The user must decide any feature extraction steps (5). The features that METHOD52 allows the user to extract are word-based unigrams, bigrams, and trigrams. Other feature extraction options in-

clude whether to normalise URLs or numbers, whether to lowercase features, and whether to remove stopwords.

Once the feature extraction steps are defined, the user can attempt to train a classifier to learn the distinction that was defined in step ③. The classifier training procedure ⑥ is fully explained separately in Section 4.1.2. If the training successfully produces a well-performing classifier (according to the evaluation set), then the classifier can be applied to all of the data ⑦, in order to produce subsets defined by the classification labels ⑧, on which the process can be iterated again (back to ②). So the user returns to *exploring* the new subset, and *searching* for more subclasses.

If the classifier training failed to produce a classifier that performs up to expectation ⑨, then the classifier may need to be discarded, and the user required to find a better split of the data. A worse alternative is that previous classifiers may also need to be discarded and the document subset definitions they imposed, if the current line of inquiry is not producing interesting splits of the data.

There are two main ways that the user may decide that a classifier is not performing sufficiently well. The first method is checking whether the classifier's performance metrics are above a desired threshold. After ④, the user has annotated an evaluation set, and during training the system will report the current classifier's performance on that dataset (see Figure 4.3). Of course, before sufficient data has been annotated, the performance will certainly be too low, so the user must first continue to annotate data until performance is no longer increasing. For this reason, it is important to continually report the classifier's performance.

The second method for checking performance is to apply the current classifier model to some new data sample, and manually check the correctness of these documents. The user can achieve this by pressing the "sample" button next to any category, then a sample of new data will be extracted and all of the ones that classifiers determines to belong in the given category will be displayed to the user in a tab just like the "Training" tab. The user can then annotate these documents to determine the precision of that category. Then these new annotations can also be submitted as training data by pressing the "submit" button.

brussels-analysis
Stop
↺
📄
Pipeline
Progress
brussels-adr-model1

Sample
Training

Label	Agreed	Disputed	Unlabelled	Kappa (micro)
attack	112			
sympathy	61			
other	27			
		0	0	
				0 (0)

10 records per page
Search:

Showing 11 to 20 of 200 entries
← Previous
1
2
3
4
5
Next →

🐦 Another tragedy . Thoughts are with the people of Brussels and of course the families of those who lost a relative.

andy
attack
sympathy
other

🐦 Soz next time il send screenshots from the ISIS group chat yeah ? <https://t.co/a7cKs38Mu3>

andy
attack
sympathy
other

🐦 Another tragedy . Thoughts are with the people of Brussels and of course the families of those who lost a relative.

andy
attack
sympathy
other

🐦 So sad about Brussels. Sending love. H

andy
attack
sympathy
other

🐦 But this is actually kind of ridiculous. Flags at half-mast for Belgium while other countries hardly get a mention in the news. [#Brussels](#)

andy
attack
sympathy
other

🐦 WATCH: Video shows the chaotic scene after explosions at the [#Brussels](#) airport. Learn more: <https://t.co/7trzLtku3w> <https://t.co/HWpSVYKGbj>

andy
attack
sympathy
other

🐦 My thoughts and prayers go out to all those affected in Brussels [#Brussels](#) [#BelgiumAttack](#)

andy
attack
sympathy
other

🐦 .@tedcruz: "In the wake of Brussels...we need a commander in chief who does everything necessary to defeat the enemy." <https://t.co/FGXNHYKhtf>

andy
attack
sympathy
other

🐦 The Brussels Airport departure hall, before and after Tuesday's explosions <https://t.co/ofcDgNlrLJ> <https://t.co/9waUZwlNMs>

andy
attack
sympathy
other

Figure 4.2: Screenshot of the “Sample” tab in the classifier component. In this tab, one or more users build a gold standard evaluation set, which the classifier will be tested on during training. The table at the top tracks how many documents have been labelled in each category, and an inter-annotator agreement score if multiple users are collaborating.

brussels-analysis

Stop Pipeline Progress brussels-adr-model1

Sample Training

Label	Precision	Recall	F-Score	Accuracy	Coded	Prior Multiplier
attack Sample	0.762	1.000	0.865		32	1
sympathy Sample	0.980	0.803	0.883		10	1
other Sample	1.000	0.111	0.200		11	1
Unlabelled	69947	Features	18	0.820		sent out:10

Get Contexts

10 records per page Search:

Showing 11 to 20 of 63 entries

← Previous 1 2 3 4 5 Next →

Training Data:

- #BREAKING: Two loud explosions at #Zaventem airport in #Brussels <https://t.co/JFw9RGLjnh> attack sympathy other
- I liked a @YouTube video from @phillyd <https://t.co/USLtbUMyJQ> Let's Talk About Brussels: The Brussels Terror Attacks attack sympathy other
- my heart goes out to the people of #Brussels this morning attack sympathy other
- My thoughts and prayers are with the people of Brussels attack sympathy other
- Ahmadiyya Muslim Community offers deepest condolences to the victims and all affected by the Brussels atrocities. attack sympathy other
- Looks like Trump plan to curtail influx of Muslims and taking fight to ISIS isn't looking so stupid. #Brussels #tcot <https://t.co/RE2FJt6MyR> attack sympathy other
- #Brussels #PrayforPeace <https://t.co/2GpQyUCvAo> attack sympathy other
- Stop moaning about linking Brexit to the Brussels attacks. This is why we want to leave! Brexit isn't a silly little game! attack sympathy other
- BRUSSELS - 36 dead - Over 200 injured - Bombings at airport and metro - Condemnation around globe - High security alert around Europe attack sympathy other
- Praying for all in #Brussels. Stay strong! attack sympathy other

Submit Labels

Figure 4.3: Screenshot of the “Training” tab in the classifier component. Once the user has labelled some separate evaluation data (see Figure 4.2) they can proceed to this “Training” tab, and label documents and features (below off-screen), which the classifier will actually use in its training. The table at the top of the tab tracks the classifier’s performance on each of the categories using recall, precision, f-score and accuracy. There is a “sample” button next to each category, which allows the user to see a sample of new documents that the current classifier determines belongs to a given category.

Attack	Sympathy	Other
HTTPLINK	my	do_you
in	of_#libya	do
raid	alcoholic_and	you_want
police	gray_a	you
in_brussels	tad_alcoholic	riiiiiigggh
paris	and_crazy	me_riiiiiigggh
paris_attacks	demonstrates	me_darcy
on	alcoholic	her_cereal
HTTPLINK_HTTPLINK	strong_respectuos	darcy_what
linked	detailed_understanding	dinner_her
linked_to	march_th	brussels_me
after	raid	want_for
to_paris	the_people	what_do
fired	not_speak	cereal
shots	HTTPLINK_#teambambiblacks	cheese_and
belgian	dictators	darcy
during	a_tad	her
suspect	@vicplusjean_gray	cheese
shots_fired	tad	dinner
gunman	break	want
attacks_HTTPLINK	@number	ops_zurich
at	@number_gov	morning_looking
killed	everything	cross_runway
brussels_raid	prayers_are	winds
officers	affected_by	runway_ops
one	sad_about	brussels_cross
operation	the_families	zurich_strong
brussels_attacks	thoughts	only_moderate
airport	prayers_go	strong_winds
attacks	my_heart	good_so
dead	prayers	winds_HTTPLINK
brussels_airport	sending_love	good
explosions	praying_for	for_dinner
injured	our_thoughts	cereal_cheese

Figure 4.4: During training, this interface is presented below the documents to be labelled. Features are suggested to the user which seem to be correlated with a given classification. The user can label these features as indicative of a classification (or input new ones at the top); these are highlighted grey.

4.1.2 Classifier Training Procedure

As in DUALIST (Settles, 2011), training proceeds in an active learning loop, in which the system repeatedly prompts the user for information that would help improve the current model, then re-trains with the new information and prompts the user again. Figure 4.5 flow-charts the individual steps of the loop.

The METHOD52 classifier training system presents the user with two labelling activities (numbers refer to Figure 4.5):

1. Label individual *features* with the classification of which the feature is indicative (12). Features are presented to the user ranked by IG, i.e those features whose presence/absence most reduces classification uncertainty are ranked highest (8). Features are

mon features, and a random sample of documents to the user ((6) & (7)), because estimations of classification uncertainty would be extremely inaccurate with so little data.

A Naïve Bayes Classifier (NBC) uses counts of the occurrences of features in documents labelled with classifications, to learn how the frequencies of features correlate with the document classifications. The NBC model details are explained in Section 4.1.3.

When the user labels a *feature* as indicative of a classification, the classifier essentially “hallucinates” having seen that feature α times (by default $\alpha = 50$) in documents labelled with that classification (these counts are also referred to as “pseudo-counts”). The user need not only label features that the system proposes; they can input their own.

The system then uses the labelled documents and features, together with the unlabelled data in an Expectation-Maximisation (EM) step, in order to train an NBC ((1) - (3)). In these steps, a classifier is first trained with only the hallucinated feature counts and is used to assign classification probabilities to all unlabelled documents. Then the final classifier is trained on the hand-labelled documents *and* the probabilistically labelled data. The counts in the probabilistically labelled data are down-weighted to 10%, in order to place more importance on the user-annotated data.

This training happens whenever the user submits new labels (11). The classifier’s performance on the evaluation data is then revealed (4).

The following summarises those aspects of METHOD52’s classifier training system, which are not present in the DUALIST framework. These features were already present in some form in METHOD52 before the work this thesis. I built the backend code for the classifier, feature extraction, and evaluation process prior to the thesis, the other features and frontend were built by other members of the TEXT ANALYTICS GROUP. Details of changes made to the classifier and feature extraction process specifically for this thesis are given in the relevant sections.

Custom feature extraction In order to better adapt to different tasks, the user has the ability to customise the feature extraction process. Tokenisation can be performed by regular expression or by the tokeniser in the TWEETNLP package (Gimpel et al., 2011; Owoputi et al., 2012), which is specifically designed for TWITTER language. The user can choose to normalise or filter out

stopwords, punctuation, numbers, and URLs. The user can elect to extract one or more of unigrams, bigrams, and trigrams as features, and whether to lower-case all features.

Re-scaling class-conditional probabilities When building classifiers with METHOD52, the user is usually in the situation where there is a large unlabelled set of data, and a small collection of hand-annotated training data. The small size of the labelled set means that the $P(\text{feature}|\text{classification})$ estimates are very sparse and noisy.

Lucas and Downey (2013) introduced a method called Multinomial Naïve Bayes with Feature Marginals (MNB-FM) for alleviating this problem. The technique involves rescaling the class-conditional probabilities $P(\text{feature}|\text{classification})$ using the more robust marginal probability statistics $P(\text{feature})$ derived from the *unlabelled* data.

This method is included as an option in the METHOD52 classifier framework.

Classification sampling When training a classifier, it is extremely beneficial to know the errors the classifier is making, so that the user can focus on providing additional training data that helps it overcome these errors. If the evaluation set was used for this purpose, a classifier would be produced that over-fits the test data with very poor generalisation capability.

Instead, METHOD52 provides a procedure for acquiring a random sample from the unlabelled set that has been classified with a specified classification by the current model. With this functionality, the user can see mistakes made without looking again at the evaluation set. In doing so, the user also acquires another estimate of the *precision* of the classifier for the given classification (the proportion of documents that the model believes should be classified c that actually should be labelled c).

Prior multipliers The user may wish to maximise recall of a given classification in order to be sure to never miss relevant documents, at the expense of introducing more false positives. Alternatively, the user may wish to maximise the *precision* in order to be more certain of the classification, at the expense of increased false negatives.

These requirements can be realised using METHOD52’s “prior multipliers” feature, which allows the user to bias the classifier towards predicting particular classifications more or less frequently by multiplying the class priors by a specified factor. This feature can often be used to alleviate performance problems as a result of an unbalanced training set (a dataset that is biased towards one or more classifications).

Evaluation When using METHOD52 classifiers for research (e.g. when using classifiers to generate aggregate statistics over datasets) it would be difficult to justify the validity of the results in some cases without discussing the classifiers’ performance. For this reason, METHOD52 allows users to set up an evaluation set of documents. After the user submits any new annotations, the classifier is re-evaluated using this data, and the results supply the user with the following information:

Classification recall For each classification c , the proportion of documents that *should* have been classified c that actually were assigned c by the classifier.

Classification precision For each classification c , the proportion of documents that the classifier labelled c which actually should have been assigned c .

Classification f-score For each classification, the harmonic mean of its precision and recall.

Accuracy The overall accuracy of the classifier: the proportion of all documents that were correctly classified.

Collaborative annotation METHOD52 provides the facility for multiple annotators to collaborate on training the same classifier and building the same evaluation set. This greatly speeds the process of producing well-performing classifiers. Additionally, inter-annotator agreement measures are calculated and presented to the user.

4.1.3 Classifier Model

This section describes the Naïve Bayes Classifier (NBC) model that the METHOD52 user trains. The following equations are adapted from those given by Settles (2011), as the basic NBC model in METHOD52 is the same as that in DUALIST.

The **NBC** assigns the classification to a document with the greatest conditional probability of classification given the document $P(c|d)$. Equation 4.1 is the standard Multinomial Naïve Bayes (**MNB**) formula. The conditional probability of classification c given a document d is equal to the product of the conditional probabilities¹ of feature given class for every feature $f \in d$, and the prior probability of classification c , all divided by $Z(d)$, a normalisation constant summing over all classifications. Duplicate features contribute equally ($f(d)$ is the frequency of f in d).

$$P(c|d) = \frac{P(c) \cdot \prod_{f \in d} P(f|c)^{f(d)}}{Z(d)} \quad (4.1)$$

Equation 4.2 describes the method for estimating the probability of a feature given a classification $P(f|c)$ (a value required by Equation 4.1). The probability is essentially the fraction of times that f occurs in documents labelled c in our training set D .

$$P(f|c) = \frac{S(f, c) + \sum_{d \in D} P(c|d) \cdot f(d)}{Z(f)} \quad (4.2)$$

For hand-labelled documents $P(c|d) \in \{0, 1\}$, but for the probabilistically labelled documents (during **EM**) $P(c|d)$ is estimated by the initial **MNB** model. Estimates derived from unlabelled data are also weighted by a factor of 0.1 in order to avoid overwhelming the training signal from the labelled data. $Z(f)$ is shorthand for a normalisation constant summing over all features in the vocabulary.

$S(f, c)$ is a smoothing term or prior, which is typically a uniform prior such as the Laplacian (a value of 1 for all features), but like **DUALIST**, we adopt a Dirichlet prior; the prior for f under c is increased by α (typically $\alpha = 50$) when the user labels f as indicative of c , as shown in Equation 4.3.

$$S(f, c) = \begin{cases} 1 + \alpha & \text{if } f \text{ is labelled with } c \\ 1 & \text{otherwise} \end{cases} \quad (4.3)$$

In practice, logarithms are used to prevent underflow.

¹ The Naïve Bayes independence assumption is that $P(d|c) = \prod_{f \in d} P(f|c)^{f(d)}$

4.1.4 Active Learning Queries

During the active learning process, the classification system will query the user with documents and features that should be labelled. The method with which features and documents are queried is the same as DUALIST, and is explained below.

Features are presented to the user ranked according to highest Information Gain (IG); these are the features whose presence/absence most reduce classification uncertainty on average, according to the current model. Both the labelled and unlabelled (probabilistically labelled by the current model) are used to compute the IG.

This is essentially:

$$IG(f) = H(C) - H(C|f)$$

Which states that the information gain of a feature is equal to the classification entropy (uncertainty) reduced by the classification entropy *conditioned on* the presence of the given feature. I.e. how much is the classification uncertainty reduced when the presence/absence of the feature is known.

This is calculated as shown in Equation 4.4:

$$IG(f) = \sum_{I_f \in \{f, \neg f\}} \sum_{c \in C} P(I_f, c) \cdot \log \frac{P(I_f, c)}{P(I_f) \cdot P(c)} \quad (4.4)$$

I_f represents either the presence of the feature (f) or its absence ($\neg f$), and C is the set of all classifications in a classification task.

Once features have been ranked according to IG, they are organised into columns representing the classifications with which they seem to most correlate. This is achieved by placing the feature into the classification with which it occurs most often, as well as any classification with which it occurs at least 75% as often.

Unlabelled documents are presented to the user for labelling ranked according to posterior class entropy, according to the current model. Equation 4.5 demonstrates how this is calculated. This value is greater the more uncertain the classifier is about the document, since entropy is greater the more uniform the probabilities it is summing over.

$$H(C|d) = - \sum_{c \in C} P(c|d) \cdot \log P(c|d) \quad (4.5)$$

4.1.5 *Methodologies for Applying Classifiers to Twitter Corpora*

Previous sections have described the process of determining definitions for classification tasks, and how classifiers are trained for a given classification task, but this is only part of the battle, a single iteration of the explore-search cycle. For large datasets, often multiple classifiers are required, which work from each other's output.

METHOD52 has been applied to many TWITTER corpora, and during this time a number of methodologies have consistently proven to be effective strategies for analysis; Wibberley et al. (2014) identify three such strategies:

Twitcident Attitudes on TWITTER tend mostly to be expressed in response to some event in the world. These bursts of reactions are termed “Twitcidents” and tend to last hours to days. The principle of “Twitcident” analysis states that each event needs to be studied separately in order to be correctly interpreted. The example given by Wibberley et al. (2014) was a speech given by the UK Prime Minister expressing a sceptical view of the European Union (EU), which elicited many enthusiastic responses. These messages therefore express a negative sentiment toward the EU. This demonstrates that the positive tone of those tweets needs to be separately analysed in the context of the speech event.

Exploratory / Patterns of Use It is rarely the case that imposing a pre-conceived classification architecture on a new dataset produces reliable results. It is a purposeful choice to make METHOD52 a framework that allows rapid training of classifiers, so that the user can attempt a classification and ascertain quickly whether the patterns in the data support it. These patterns correspond to the notion of “class” in this thesis, classes which are explored and isolated in the explore-search cycle.

Russian Doll At each stage of analysis, the data is usually dominated by one pattern of usage, overshadowing less prominent patterns. Once a classifier is trained to pull away the documents representing the dominant pattern, new patterns are revealed in the remaining data. This process can generate a chain of classifiers revealing various patterns of usage.

Each individual stage of classifier learning and pattern analysis is made very rapid by METHOD52, but as the corpus size increases and

we identify many patterns across many topics, the load on user time increases. The number of classifiers being chained increases and each link in the chain requires the same exploratory labour.

Section 4.2 illustrates the process of applying the above methodologies to a large dataset; it demonstrates that the methodologies, while powerful, can still result in much user effort, which could be reduced by improved feature discovery strategies. Section 4.3 introduces a second dataset, which together with Section 4.2 motivates the remainder of the chapter, which introduces new methodologies to further exploit feature discovery for class exploration and isolation.

4.2 DATASET: BRUSSELS BOMBINGS

This section describes the collection and analysis of a TWITTER dataset undertaken specifically for this thesis, in order to demonstrate the iterative process of exploring a dataset, defining classes, isolating those classes, and beginning the process anew for the isolated subset. It shows how important to this process bespoke classifier training is. It demonstrates how this iterative procedure leads to engagement with the data.

At the same time, this kind of bespoke analysis takes thought and time, and while METHOD52 makes it possible, there are still improvements to be made to decrease the time and effort that is necessitated by the current methodologies, and increase the effectiveness of the approach. This section focuses on exposing those areas in need of improvement which primarily motivates the “feature exploration” approaches introduced at this chapter’s start.

Given that TWITTER is a very reactionary, event-driven medium (the “Twitcident” principle (Wibberley et al., 2014)), the largest datasets focused on coherent issues are often those in response to some important real-world event. At the time of this experiment, an obvious such event was the 2016 Brussels bombings. 8.4 million tweets were obtained containing the word “Brussels” using the TWITTER API from shortly after the last bomb detonation on the 22nd March, 2016 until the evening of the same day, with the aim to investigate the nature of TWITTER discussion of the bombings. The collection is first filtered such that all those tweets whose language is not detected as English by TWITTER are removed.

Once a large dataset of documents has been collected, it is difficult to know where to begin with the analysis. The user can often only

guess at the types of conversations that are present in a collection of millions of documents. It is unlikely to be useful to train a classifier expecting it to find a classification that occurs once in ten thousand, or to apply an existing classifier for classes such as positive/negative language, if we do not understand the context of the positivity/negativity. Instead, the data must be explored. Random samples of the Brussels dataset were manually read to discover discussion types. Once potential classes of interesting tweets have been discovered, classifiers can be trained to attempt to pick out these groupings of tweets. And by pulling out these classifications, reducing the noise in the set, we can begin to analyse the Brussels conversation. This is the “Patterns of use” principle (Wibberley et al., 2014). And it is also the “inspect” and “define” sub-steps of the “explore” half of the explore-search cycle.

Despite the fast classifier training process that METHOD52 offers, this analysis can be extremely time-consuming, it needs to be repeated every time a new subset of the data is extracted, as we slowly drill down through the noise to the different discussions present in the data (each iteration of the cycle). Furthermore, at each stage of training, it is likely that when the analyst encounters the actual text, the assumptions about which classifications can be made must be discarded or drastically altered.

In the first layer of the analysis, it became clear that although there were some tweets not discussing the bombings, the majority of tweets were either discussing the attack or expressing sympathy for its victims. The tweets discussing the attack were very diverse, and could obviously be sub-divided into distinct categories.

The seemingly logical first step is to attempt a classification task that encompasses all of the categories that can be seen in the data for the first classifier, because the categories are observable already, and only one classifier would be needed to split them. This could be a four/five-way classification that includes “sympathy/condolences”, “irrelevant/other”, and two or three categories for the sub-categories of “attack-related”. Wibberley et al. (2014) suggest though that this scenario is usually more amenable to the “Russian Doll” strategy, in which we attempt to pull away one or two patterns of use at a time with separate classifiers. It is otherwise often unreasonable to expect the fairly simple NBC algorithm to distinguish between many classifications, especially when there is a substantial vocabulary overlap between documents of several classifications, as in the sub-categories

of attack-related documents (Section 4.4.5 describes the vocabulary overlap problem and considers a technique for addressing it). The first classifier was therefore trained to distinguish three categories:

Condolence Offering sympathy for the victims and their families.

Attack Related to the attacks in any way except if offering condolences or sympathy.

Other Content is unrelated to the Brussels bombings.

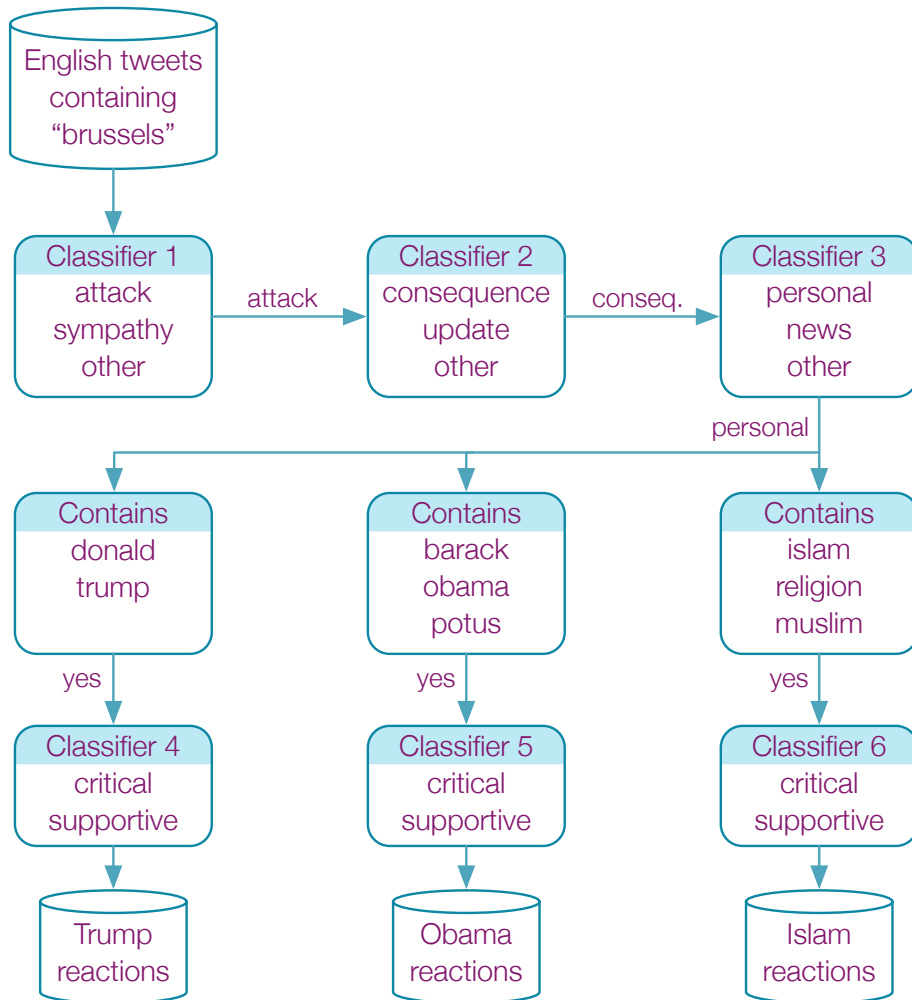


Figure 4.6: Shows how three datasets “Trump reactions”, “Obama reactions”, and “Islam reactions” were created. Each resulting dataset consists of personal reactions to the Brussels bombings on the topics of Donald Trump, President Obama, and Islam. The personal reactions are categorised into either critical or supportive of Trump, Obama, and Islam respectively.

When building a classifier in `METHOD52`, the user must first build a gold standard set of labelled documents, which will be used to

evaluate the classifier, not to train it. This is done in the “Sample” tab as shown in Figure 4.2. Then when labelling training documents in the “Training” tab, the user has access to the information shown in the screen in 4.7, allowing them to determine precision, recall, f-score, and accuracy of their classifier as it is trained. In the Brussels study, the classifiers were trained until it seemed no improvements could be made to the f-scores of at least the categories that were being passed to the next component of the METHOD52 job.

Actual	Predicted	attack	sympathy	other
attack		112	0	0
sympathy		12	49	0
other		23	1	3

Document Category Layout

Advanced EM Settings

Label		Precision	Recall	F-Score	Accuracy	Coded	Prior Multiplier
attack	<input type="button" value="Sample"/>	0.762	1.000	0.865		32	<input type="text" value="1"/>
sympathy	<input type="button" value="Sample"/>	0.980	0.803	0.883		10	<input type="text" value="1"/>
other	<input type="button" value="Sample"/>	1.000	0.111	0.200		11	<input type="text" value="1"/>
Unlabelled		5114	Features	18	0.820		sent out:10

Figure 4.7: Screenshot of the METHOD52 classifier evaluation screen, which appears above the training documents being labelled. The upper portion of the table shows the number of documents classified correctly or incorrectly for each classification, and the lower portion gives details of precision, recall, f-score, accuracy. From here, the user can also click the sample button to take a random sample of documents which the classifier determines belongs to the selected classification.

At this stage, having noticed that the “attack” tweets seemed to be quite diverse in their content and that, conversely, “condolence” tweets seemed to follow a very predictable pattern, it was determined that “attack” tweets were the most interesting category to pursue. Therefore, the analysis continued solely with those tweets.

The process of random sampling and reading began again to determine how to further divide the “attack” tweets. The most time-consuming issue is identification of categories that do not exist with enough frequency to be amenable to classification, since this fact is rarely known in advance of training. Many classifiers were attempted and subsequently discarded, wasting annotation effort. In particular,

some tweets were personal reactions to the bombings, people's opinions on blame and how society should react. These tweets seemed to be the most interesting, but random sampling of the current dataset did not produce enough positive examples to annotate testing and training data. However, a potential goal became determining whether enough layers of tweets can be filtered in order to expose more of these personal reactions tweets.

At this point, the dominant type of tweet seeming to form a coherent category was news and updates on the attacks themselves. The successful classifier (accuracy exceeding 65%, which is typically low, but used due to time constraints) in this step defined classifications as follows:

Update Specifically bearing news regarding the bombings, e. g. casualties, suspects, etc.

Consequence Related to the attack, but not an update on the bombings, e. g. transport closures as a result of the bombings, support/helplines, advise, commentary.

Other For the persistent presence of irrelevant data.

The most interesting and varied tweets were within the "consequence" category; "update" tweets were popular, but essentially repeated facts about the investigation surrounding the bombings (information like this comes from failed attempts at different types of update classifiers). Therefore, only the "consequence" tweets were input to the next iteration, a category which included any personal reaction tweets that were encountered in training.

The newly generated subset of data still contained a wide variety of discussion. But one category was beginning to be more well-defined: an "informational" or "news" category, wherein the tweets were mostly characterised by the sharing of facts or information. Tweets not in this category were of much more interest, since they resembled most often the personal reactions that were beginning to surface in the dataset.

Therefore, the next classifier performed the following split:

News Brussels-related but purely informational, objective, or news.

Personal All other Brussels-related including personal reactions to the bombings, whether that be opinion, advice, or commentary.

Other Again maintained for persistent irrelevant data.

The need for an “other” category and the problems that it brings are commonly encountered in this type of analysis. When a classification problem has an “other” class for completely irrelevant documents, this is the “none of the above” problem. Given that it is essentially the “not what we want” category, selecting features to label for this category is problematic: here are always infinite possibilities for the things the analyst *does not* want. It can be difficult to determine which features should be labelled as indicative of the “none of the above” category. Section 4.4.4 provides a strategy for improving results in this scenario.

The “other” category is a useful designation for tweets which do not easily conform to the main categories. But this is often an added complexity that the classifier would better be without, and can sometimes be a sign that the categories require different definitions. Up to this point, the analysis has already taken many days (see Table 4.4).

The classifiers were chained together as shown in Figures 4.6 & 4.8 in order to produce a dataset of personal reactions to the Brussels bombings. The exploration then began again: are there different types of personal reactions? Are there distinct topics aside from the bombings themselves that are discussed in the reactions? Indeed, experience with hand-labelling some personal reaction tweets, and more random sampling of the personal reactions revealed that there were many tweets about Barack Obama (US president at the time), Donald Trump, and Islam/Muslims. The data was split into these categories using the presence of keywords describing these topics:

- Tweets containing “obama”, “barack” or “potus” are part of the “*Obama reactions*” dataset.
- Tweets containing “donald”, or “trump” are part of the “*Trump reactions*” dataset.
- Tweets containing “islam”, “muslim”, or “religion” are part of the “*Islam reactions*” dataset.

Given the above rules, a single tweet can be part of multiple datasets if it contains the appropriate words. Tweets that contain none of the above words are discarded. In METHOD52 this is accomplished with a KEYWORD ANNOTATOR as shown in Figure 4.8.

This produced three data strands (summarised in Table 4.1), each containing personal reactions to the Brussels bombings in relation to one of three common topics. Each of the three datasets were then

subject to their own iteration of the analysis, which resulted in the determination that it was possible to split each dataset into two categories: whether the author is *critical* or *supportive* of the featured topic (e.g. Islam).

Dataset	Documents	Description
Islam reactions	255,000	Personal reactions to the Brussels bombings concerning Islam/Muslims
Obama reactions	56,000	Personal reactions to the Brussels bombings concerning the President of US, Barack Obama
Trump reactions	144,000	Personal reactions to the Brussels bombings concerning Donald Trump

Table 4.1: Personal reactions dataset summary.

This analysis clearly models the explore-search cycle: data is presented to the analyst, classes are identified such as news and incident updates, and classifiers are trained to isolate them so that they can be stripped away to reveal further interesting classes like the personal reactions in subsequent iterations of the search/explore cycle.

At this stage, each personal reaction topic can be analysed in isolation. The same tweet could be either critical or supportive depending on whether we're interested in Donald Trump or Barack Obama as our topic. Therefore, attempting to apply a general purpose sentiment classifier to the complete dataset would only be useful if the analyst was looking for positive or negative language in general, instead of where that sentiment is directed.

In order to expect good performance from a classifier, one must have a solid idea of the definition of each classification. The more inconsistency in the labelling, the worse a classifier will perform, because the model will learn flawed statistics about how often features should occur in each category. The following discussion describes the set of rules that were necessary to consistently annotate each of the personal reaction datasets as either supportive or critical of their target topic, in order to demonstrate how important this concept is. This in turn demonstrates the importance of the explore step, and bespoke encoding and isolation of classes.

Each of these rules lend support for their conclusion. Often, the classification is made even clearer by several of the rules being applicable.

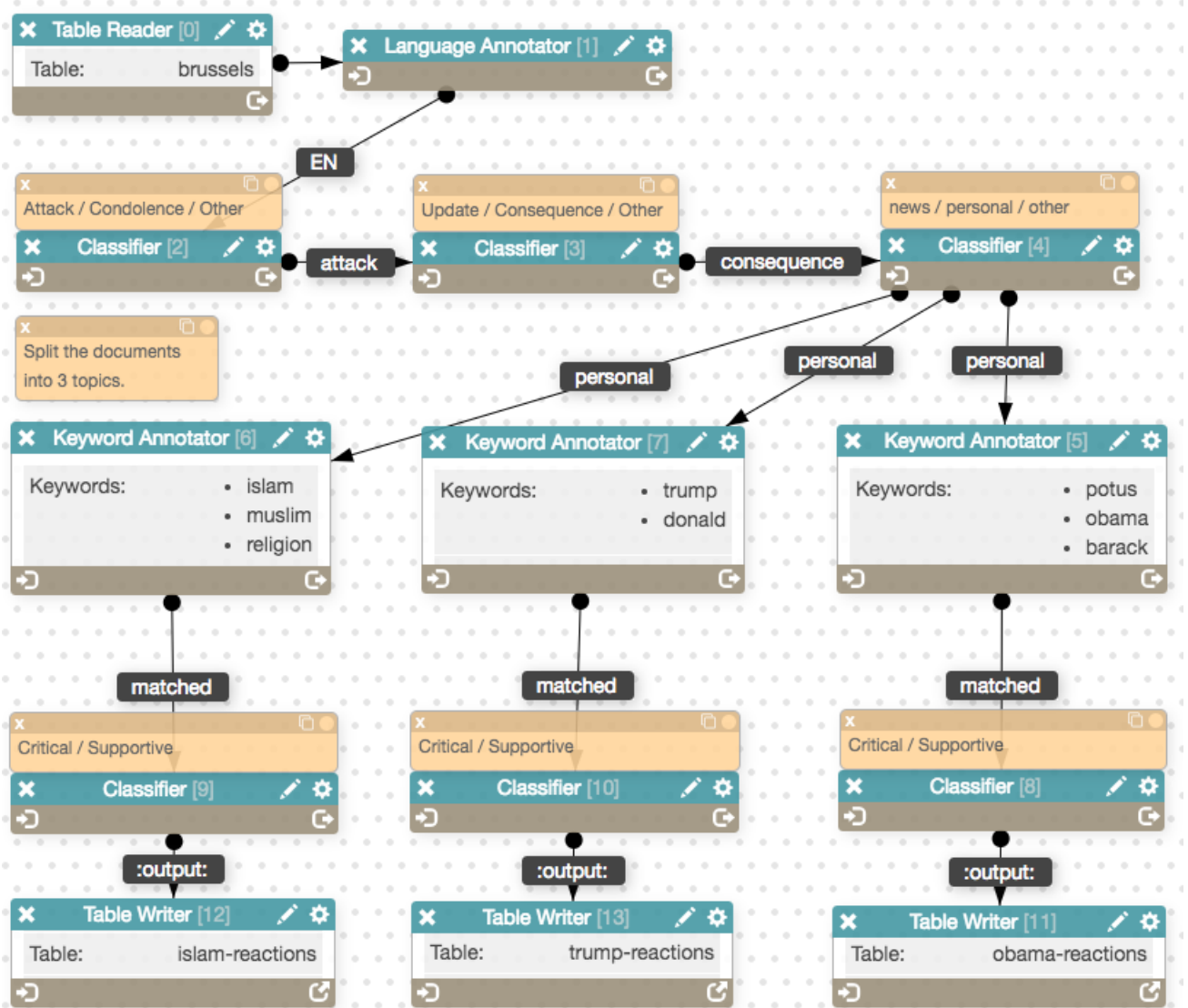


Figure 4.8: This is a screenshot of the instantiation of Figure 4.6 in a METHOD52 job.

Direct If the user makes a direct criticism of the topic, then the document is critical, and likewise, if the comment directly says something supportive about the topic, then the document is supportive.

Sharing If the user is sharing supportive/critical material (e.g. retweeting, or link sharing) without further comment, then assume the user agrees with the sentiment of the material and label supportive/critical respectively. Otherwise, use other rules to judge the tweet based on the user's further comment in the context of what they are sharing. If the user is sharing nothing but an emotive supportive/critical hashtag, then assume the user

agrees with the sentiment of the hashtag and label supportive/critical respectively.

Self-support If the user is actually the subject of the dataset topic, e.g. A Muslim in the *Islam reactions* dataset, and they are defending themselves, then these are documents supportive of the topic.

Self-exclusion When the user’s comment identifies the supporters of the topic as a separate group or entity, the comment is critical, because it implies the author is not themselves part of the supporters. E.g. users that make negative comments about Donald Trump supporters, are generally excluding themselves from the group of Trump supporters.

Concessions If a document makes small supportive concessions, but is ultimately critical, then the document should be labelled critical, and vice versa with supportive.

Opposition If there is a natural opponent to the dataset topic (e.g. other candidates for presidency), then criticism of the opponent is labelled as supportive for the target, except where the criticism is clearly directed at both opponent and target, or when the comment is clearly outside the scope of the opposition.

Agreement The supportive/critical status of a comment can depend on whether the comment is supportive/critical of others who are being supportive/critical.

The last rule “agreement” introduces a very necessary point for interpretation of tweets by the annotator. Not only must we apply the other rules to find out whether a given sentence is critical or supportive in nature, but in order to determine whether the message as a whole is critical or supportive of the current topic, we must follow the targets of the individual critical/supportive sentences. This so we remain consistent in our annotation effort.

Table 4.2 defines simple agreement rules, and Table 4.3 demonstrates the rules being applied to actual TWITTER examples. Below, the examples are explained along with why the conclusions drawn by the application of the rules are justified.

Rule	Meaning
S	Person supportive of Islam
$\neg S$	Person critical of Islam
$Supp(x) = x$	Person who is being supportive of x holds the same supportive/critical status as x
$Crit(x) = \neg x$	Person who is being critical of x holds the opposite supportive/critical status of x

Table 4.2: How the critical/supportive status of a comment can depend on whether the comment is critical/supportive of others being supportive/critical, using the Islam reactions dataset as an example.

Example	Target	Rule Application
1 @USERNAME YOU are a hate preacher. #Islamophobia is hate preaching. #Bruxelles #brussels	Islam/Muslims	$Crit(\neg S)$ $= S$
2 The #Brussels tag is loaded with morons crying about racism & Islamophobia	Islam/Muslims	$Crit(Crit(\neg S))$ $= Crit(S)$ $= \neg S$
3 @USERNAME American stands with Brussels and against hatred and only those blinded by fear support Trump	Donald Trump	$Crit(S)$ $= \neg S$
4 Cruz tryna blame Trump for the Brussels attack lmao. Anyone who supports Cruz is a brainwashed monkey puppet. #TheWorldIsBurning	Donald Trump	1. $Crit(Supp(\neg S))$ $= Crit(\neg S)$ $= S$ 2. $Crit(\neg S)$ $= S$

Table 4.3: Example usage of the rules defined in Table 4.2. Usernames have been anonymised.

Example 1 The author is critical of someone who is being critical of Islam; the author accuses him/her of Islamophobia. The rule is justified in identifying the author as *supportive*, since being concerned with calling out hatespeech against a group is itself being supportive of that group.

Example 2 The author is critical of users on the Brussels hashtag who are criticising those speaking negatively (critically) of Islam. Therefore, the author is critical of supporters of Islam, and thus is *critical* of Islam. This conclusion is justified because the author is being critical of Islam by implying that potentially Islamophobic comments are in fact justified.

Example 3 The author criticises the supporters of Donald Trump by suggesting that they are only supporters because they are blinded by fear. Being critical of their support shows disagreement for that support, which implies that the author is critical of Trump, as the rule suggests.

Example 4 The last example has two lines of argument, both of which agree the author is a Trump supporter.

1. Ted Cruz is critical of Trump, and by identifying that Cruz is trying to blame Trump, the author is being critical of Cruz's criticism of Trump, thus the author is being supportive of Trump.
2. The author is critical of those supporting Ted Cruz, who is being critical of Trump. This implies the author is a supporter of Trump.

The complexity of these rules, and the layers of filtering done to produce these datasets in the first place exposes the potential problems with just applying an out-of-the-box pre-baked positive/negative classifier to the 8 million tweets. In other words, it highlights the importance of discovering and defining the class definitions in order to produce an informative study.

The process of arriving at the critical/supportive classifications of the topic-based datasets took weeks of effort, but this bespoke analysis is crucial to actually understanding the data. Only now that the data is broken down to smaller categories of meaning, can the researcher read samples from the categories productively, to build a coherent picture of *why* people were critical/supportive of Donald Trump in the wake of the Brussels attacks, or whether people came out in support or criticism of Islam, and what the nature of that support/criticism was.

Section 4.4 uses the Surprisingly Frequent Phrase Detection (SFPD) technique from Chapter 3 to speed analysis and engagement with data during classifier training, and provide support for a more effective exploration than simple random sampling, or other means of a grouping documents which still involve reading many of them. The sections that follow afterwards detail how to utilise features discovered in the dataset during classifier training to help isolate the classes of interest.

Classifier	Days Building
Attack / Sympathy / Other	2
Update / Consequence / Other	5
News / Personal / Other	2
Trump: Critical / Supportive	4
Obama: Critical / Supportive	3
Islam: Critical / Supportive	3

Table 4.4: This table shows time taken in working days to complete each classifier model. The second classifier was particularly time-consuming because many classifiers were abandoned while trying to determine a viable split of the data, and where the analysis was ultimately heading (i.e. personal reactions). The Critical/Supportive of Trump classifier took longer than the other similar classifiers since it was the first criticism classifier, and much of the work to codify what criticism meant in this dataset was accomplished here.

4.3 DATASET: FATHER'S DAY

The dataset discussed in this section will primarily serve as the case study for the problems addressed by the “feature extraction” and “feature incorporation” approaches outlined in the introduction. User difficulties with understanding and labelling individual features are highlighted in subsequent sections using data from this dataset as examples, but these difficulties represent issues that arise in general using the `METHOD52` classifier approach.

In a study by [Wibberley et al. \(2014\)](#), the goal was to inspect tweets mentioning Father's day, in order to find users to whom it would be appropriate to send Father's Day marketing messages.

The Father's day dataset required the same strategies used in the Brussels example. The initial expectation was that there would be three types of tweet:

- The target subset: users asking for gift ideas.
- Marketing messages.
- General Father's day discussion.

In the initial analysis, the tweets were completely dominated by marketing tweets; two layers of classifiers were needed to learn classes of marketing tweet and filter away this data. During exploration of the resulting data and attempts to create a classifier for identifying the target tweets, an unexpected category of tweets emerged, to whose authors the sending of Father's Day marketing messages

would be wholly inappropriate. This category was named “sad/distressed”. The following is a summary of the final categories for the data with examples:

Sad / Distressed The author of this tweet is *not* a suitable candidate to receive a marketing tweet about Father's Day, due to their negative views on the subject. E.g. *Every year fathers day rolls around & its such a joke. Thanks for being there you f***ing piece of sh*t a**hole "father" #f***you*²

Suitable / Target The author of this tweet is a suitable candidate to receive a marketing tweet about Father's Day. E.g. *I should probably get something for my dad for Father's Day considering he does everything for me*

Marketing the author of this tweet is marketing some product or service. E.g. *Not sure what to get Dad for Father's Day? We make custom baskets! Whether he likes single malts or fine cigars,...* [HTTPLINK](#)

Without applying this layered analysis through iterations of the class explore-search cycle, this category may not have been discovered, which could be much to the detriment of the marketing campaign.

As discovered in the Brussels attacks analysis, more could be done to bring information to the user's attention faster and more effectively. Strategies for achieving this follow in Section 4.4.

Training well-performing classifiers is a difficult problem, especially without deep knowledge about the capabilities of the classifier algorithms, which many social scientists and humanities scholars do not have. Section 4.4.5 attempts to remedy this situation in particular, by introducing an **SFPD** classifier training methodology that allows direct engagement with the underlying properties of the data that help and hinder classifier performance.

Classifier training is also complicated by the ambiguous nature of language, and the assumptions about it that users bring to the table. A particularly ambiguous word in the Father's Day dataset was **LOVE**. Our initial assumption may be that it would be used in *suitable* tweets mentioning love of one's father. The reality is quite different, since it is used in different manners across both *suitable* and *marketing* messages. Furthermore, the simple surface feature labelling is weak at then providing a facility to label a feature that would distinguish between these different uses of **LOVE** in a generalisable way. Sections

² Asterisks not present in the original tweet.

4.5 & 4.6 attempt to alleviate this problem by involving the user directly with the model features, and adding more options for the feature extraction process.

4.4 EXPLORATION USING SURPRISINGLY FREQUENT FEATURES

It took weeks to form the pipeline of analysis for the Brussels dataset (see Table 4.4), because it involved repetition of the process of randomly sampling data, reading many tweets, hypothesising possible classifications and training classifiers to test those theories. Then each time a classification worked, it produced a new sub-dataset needing similar analysis. Any sufficiently large dataset presents similar challenges. The aim of this section is to provide better strategies using feature discovery for data exploration, with the goal of defining classes of documents in the data.

While in this chapter [SFPD](#) is used mostly in conjunction with classifiers, and data that has been isolated based on the classifier's predicted classes, given its origin in explaining topics, it can easily be applied to aid feature discovery in the output of topic modelling, or clustering techniques.

Table 4.5 shows the result of applying Surprisingly Frequent Phrase Detection from Chapter 3 to the raw Brussels dataset before any classifiers have been applied. The reference corpus was the default large WIKIPEDIA corpus. The phrases shown are a selection from the top 50. The phrases turn out to reveal the main categories of the first classifier in the pipeline (they are organised below into the categories that they are indicative of). This knowledge would have greatly sped analysis because it tells us which classifications to try first, and gives us features that we might label as indicative of the classifications.

Condolences
condolences to the victims of the brussels
heartbreaking
thoughts and prayers are with the people of brussels
bourse square
pray for brussels
so sad to hear about brussels
solidarity with
Attack information and fallout (Update & Consequence)
paris attacks suspect salah abdeslam
zaventem airport
explosions
maelbeek metro station
maalbeek metro station
key suspect in paris attacks captured in brussels raid
eurostar
jihadists
daesh
kalashnikov rifle
counterterrorism raid in brussels
airport several injured
salah abdeslam
unexploded suicide vest
eu foreign policy chief federica mogherini
tihange nuclear power plant
Related Topics
brexit
trump
nigel farage

Table 4.5: Selection of [SFPD](#) phrases from the top 50 generated from the full raw Brussels dataset. The top two sections correspond with classifications that were applied to the data, and the last hints at other potential topics in the data.

Note that many of the phrases, especially in the “attack information and fallout” list would be excellent choices to form a query to TWITTER to get more information on each of the events hinted at by the terms. It is too late to get a real-time stream of such data without paying a large sum to a tweet archiving service. This problem will frequently arise because it is difficult to predict relevant phrases

without analysing the data first. A project begins by collecting data using some initial terms (perhaps even before project start dates), and once the project gains momentum, and there is enough data for analysis, researchers find terms that would have produced more relevant data had they been included in the query terms. And by this point, in order to complete the analysis, effort will have already been invested in training many bespoke classifiers. Therefore, it becomes impractical to restart collection with new terms. Chapter 5 is concerned with producing a strategy to solve this problem by automatically adapting the TWITTER query, to avoid missing this data.

Also interesting to note, is that we typically use the kinds of phrases shown in Table 4.5 to discuss the classification definitions, or as evidence for the existence of the classifications in the data. But without some kind of feature discovery, the analyst must hunt these features down in documents manually, or rely on those proposed in active learning.

On TWITTER, news headlines are often repeatedly shared with slight alterations. These shares can skew frequency statistics by biasing towards phrases in these headlines. This is avoided by filtering tweets using a near-duplicate detection algorithm. This is covered in more detail in Section 5.5. Experiments with SFPD in this section are all undertaken after the target and reference corpora are stripped of near-duplicate documents.

Once a classifier is applied to the data, extracting a subset of documents, the SFPD can be reapplied to the resulting data to continue the process again, gleaning more specific information and hints for classification once other patterns of use have been stripped away. This strategy of application of SFPD will be referred below as “SFPD for Corpus Insights”. The following subsections discuss this and other strategies for incorporating SFPD into the classification analysis loop.

4.4.1 SFPD for Corpus Insights

This section describes the simplest use case of SFPD to aid classification and corpus exploration.

Previously, without SFPD, in order to search for classes in the data, to define a classification problem, the METHOD52 user could read random samples of the dataset, attempt to train a classifier on their hypothesis, and then discard it and begin again if the data did not support the classification task. Here, the user is at the mercy of chance,

the chance that the random sampling draws enough documents to represent themes that are not only of interest, but that occur commonly enough in the data to be amenable to a classifier. The positive is that very common themes in the data are more likely to occur in a uniformly random sample. Otherwise the analyst could also resort to `METHOD52`'s k-means clustering component and attempt to interpret the clusters of documents; `SFPD` is compared to clustering in Section 4.4.1.1.

Instead of relying on random sampling, `SFPD` can be applied. Using the corpus of interest as a target set, and some large reference corpus, the phrases returned can hint at the kinds of classifications that it could be possible to apply. Table 4.5 demonstrated this effect. The phrases in the table would have directed the analyst sooner to the categories that were otherwise discovered only through random sampling and experimentation over a long period of time.

A keyword extraction approach like `SFPD` is appropriate here, because determining the top most interesting features is a rapid method of gaining an overview of a corpus. It complements the random document sampling approach, because it gives it potential focus. Once a feature has been found, the user may already have an idea about potential classification problems. But if not, they can constrain a random sampling session to only those documents containing the feature of interest using `METHOD52` (job shown in Figure 4.9).

By varying `SFPD`'s λ (likelihood-lift ratio) parameter, we can favour high-frequency terms (λ closer to 1 than 0), in order to determine categories that must be identified in earlier classification steps. Consider the *condolences* category. "Condolences" was a category of tweets that dominated the data, so it was necessary to strip it away before we could analyse the nuances of the "attack" category in subsequent classifiers. Therefore, by introducing a bias toward frequent surprising terms, we find terms from categories that will more likely need analysis first. This retains the ability to discover common themes first, which was an advantage of the approach relying solely on random sampling.

By favouring more the *lift* of terms (λ closer to 0), we can find surprisingly frequent terms that aren't necessarily very frequently occurring. Performing this analysis early may give an idea as to where the overall analysis could go after stripping away the more frequent categories. For example, the "related topics" list reveals that the Brussels analysis in Section 4.2 may have missed discussions on Brexit

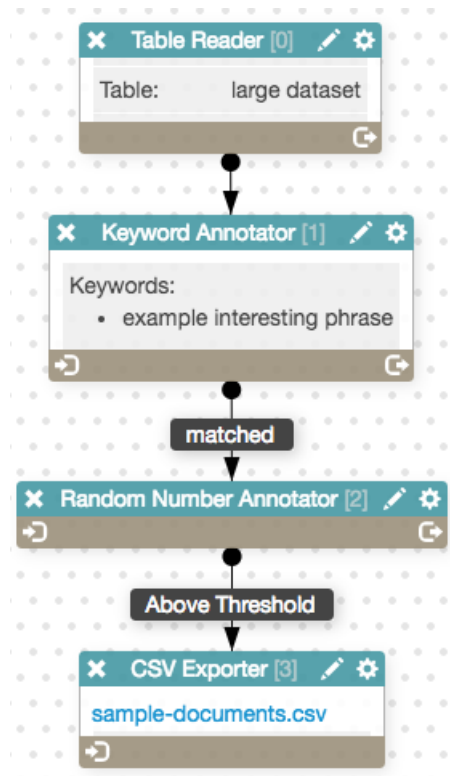


Figure 4.9: Screenshot of job pipeline which filters a dataset for only those that contain a specific term. A random sample is drawn from the filtered documents and exported to a csv file.

and Nigel Farage in light of the Brussels attacks. Knowing this early allows for the planning of the overall pipeline/processing strategy, including which classes of document should be stripped away, and which terms should be considered for labelling in certain classifiers.

The size of the dataset under consideration may also affect choice of the λ value. With a small target dataset, consider biasing toward the lift (λ closer to 0), since in a small dataset, there is less difference between the frequencies of features, and those that occur with enough frequency to be surprising compared to the reference are usually interesting overall in the small dataset. But when the target set is large, consider biasing more toward the likelihood (λ closer to 1), since in a large dataset there are many more chances that each word does not occur in the reference set, raising the chances that terms in general will occur surprisingly often. Therefore, focus on those features which occur more frequently, as they may be more likely to be interesting, or at least constitute a class that can be identified for either analysis or filtering out.

A client of CASM CONSULTING is currently interested how industrial strategy is discussed online. They employed the corpus insights

approach, wherein the target set was a large selection of tweets collected on the query “industrial strategy”. They experimented with different values for λ when trying to identify phrases of interest. They found utility in first using SFPD with $\lambda = 0$ to find rare words and acronyms, since while these phrases were generally very noisy, they were occasionally useful information that would be missed when biasing toward more frequent terms. Unsurprisingly they rarely found use for the other extreme $\lambda = 1$, which ignores the reference corpus completely.

This section has described strategies for finding insights in a corpus using SFPD, which can aid classification tasks generally by facilitating the discovery of interesting classes of documents. But using SFPD on a classification task in progress has not yet been explored. Section 4.4.2 continues with SFPD strategy for exploring documents under a specific classification during a classification task.

4.4.1.1 *Clustering Comparison*

This section compares SFPD to the k-means clustering method. A random sample of 500,000 tweets were taken from the Brussels data, then near-duplicate and non-English tweets were filtered out, leaving 202,467. These tweets were then clustered using k-means with 10 clusters. From each resulting cluster, 50 tweets were randomly sampled and manually inspected for an overall theme. Table 4.6 lists the discovered themes by cluster, and Table 4.7 shows selected example tweets for each theme.

Cluster	# Tweets	Theme
1	113759	–
2	14706	–
3	13992	Arresting/Capturing suspects
4	13238	Explosions
5	17917	Thoughts and prayers
6	3383	Moment of silence
7	9725	Safety
8	13216	–
9	2392	Islam
10	13377	Latest information

Table 4.6: 50 random documents were sampled from each cluster and manually inspected to find common themes among them. Clusters 1, 2, & 8 did not seem to have any coherent theme, possibly because we are forcing k-means to work with ten clusters when it does not naturally find ten distinct ones. The large cluster could be indicative that there are many much smaller clusters if we were to repeat the process on this subset.

After performing clustering, we must still interpret how the data has been divided into clusters, which entails reading many documents anyway. Therefore, it may not save the time that identifying themes by key phases could. The methods can be complementary because [SFPD](#) could be used on each cluster separately to gain different views on the data, and aid in the interpretation of the clusters (as the original word relevance measure was used to interpret the output of topic modelling ([Sievert and Shirley, 2014](#))).

Both K-means and [SFPD](#) discover the highly frequent themes of condolences and updates about the attacks. But some of the larger clusters would seem to need more processing in order to be divided into more coherent topics/themes. And clustering methods must group together *whole documents*, but when discovering topics at the phrase-level, we can quickly arrive at specifics, e.g. “unexploded suicide vest” or “tihan nuclear power plant” from Table 4.5.

A key difference in how clustering compares to [SFPD](#) is the fact that the clustering algorithm only has access to the target data, the data that needs to be categorised. [SFPD](#) has the potential advantage that a reference corpus can be selected in such a way as to tailor what [SFPD](#) should determine as surprising (and therefore of interest).

Cluster 3: Arresting/capturing suspects
<p>This man was arrested in the UK for "inciting racial hatred" due to a single tweet. No 1st amendment I guess. HTTPLINK</p> <p>Abdeslam was shot in police raid in the district of Molenbeek at 4:30pm HTTPLINK</p> <p>BREAKING: Paris attacks #ISIS fugitive Salah Abdeslam has been captured in #Brussels in a police operation. HTTPLINK</p> <p>BRUSSELS (Reuters) - Belgian police arrested seven people in overnight raids in their investigation into Islamic State suicide bombing...</p> <p>#BrusselsAttacks: Third Brussels suspect arrested, also wanted for Paris attacks: Reports >>HTTPLINK</p>
Cluster 4: Explosions
<p>Explosions at Airport and Subway Station Bring Brussels to a Halt: The attacks, which killed at least 13, put ... HTTPLINK</p> <p>2 Explosions Rip Through Brussels Airport; Deaths & Injuries Reported Two explosions ripped through the Brussels a... HTTPLINK</p> <p>Live blog: After Brussels airport blasts, explosion at metro station</p> <p>3rd explosion heard in Brussels, this time at a Metro station.. Entire metro system shut down in Brussels</p> <p>RT_com: BREAKING: Several people injured in #BrusselsAirport departure hall following 2 explosions - Belgian media HTTPLINK</p>
Cluster 5: Thoughts and prayers
<p>Praying for the world. #Brussels #PrayersForBrussels #prayforpeace HTTPLINK</p> <p>Shocking news from #Brussels today. Our thoughts and prayers are with our team members and partners in Belgium.</p> <p>We're sad to hear about the attacks in #Brussels. HTTPLINK</p> <p>Sending thoughts and prayers to #Brussels for those killed, for those injured, their families and all affected by today's attack.</p> <p>I have all the people & families of those affected in Brussels in my heart today. Pray for Belgium. #PrayForBelgium #PrayForPeace</p>
Cluster 6: Moment of silence
<p>Where was the moment of silence for the many dead in #Gaza #Brussels RIP #Gaza RIP HTTPLINK</p> <p>RT tomwfootball: Minute's silence in memory of Brussels victims ahead of Wales v Northern Ireland. Belgium played ... HTTPLINK</p> <p>Don't give victims of tragedy a "moment of silence". Theres already too much silence. Don't give them silence, give them voices. #Brussels</p> <p>Moment of silence at the United Center to honor those who died in the Brussels attacks. HTTPLINK</p> <p>Very nice, but was there a moment of silence for Ankara or Istanbul? #JustCurious HTTPLINK</p>
Cluster 7: Safety
<p>hope everyone in brussels is safe</p> <p>To my friend Luloann who flew into Brussels, Belgium last night. Praying you have a fun, safe trip and safe return home to Denmark!</p> <p>I'm sad. And suddenly I don't feel safe anymore. #Brussels</p> <p>stay safe HTTPLINK</p> <p>Thinking of everyone involved, please be careful & stay safe !! #Brussels</p>
Cluster 9: Islam
<p>Brussels: The BBC hasn't used the word Muslim for a whole day but mentions Islamophobia at regular intervals.</p> <p>As usual, not sure what's more depressing/predictable: bigots blaming Islam or conspiracy theorists shouting 'false flag'. Ugh. #brussels</p> <p>America is not #Brussels - US Muslims are "better assimilated" Our welcome is our strength, Freedom our best defense HTTPLINK</p> <p>All those saying #StopIslam should know in reality true Islam is the cure to violence. True Muslims want peace #Brussels #MuslimsForPeace</p> <p>HUGE difference between TRUMP & Lyin' Ted: Trump called for a halt to Muslim refugees b/f Brussels was hit-Cruz copied Trump AFTER Terror.</p>
Cluster 10: Latest information
<p>BELGIUM LATEST: - 28 killed after explosions at Brussels airport, ... HTTPLINK</p> <p>RT ShannonBream: Leading SpecialReport FoxNews - GregPalkot LIVE from Brussels with breaking details on the latest terror raids</p> <p>The Latest: Kasich says he's 'sickened' by Brussels attacks HTTPLINK #UE4 #STEAM #GIVEAWAY #unity3d #gamedev #indiedev</p> <p>Follow our Live Blog for latest updates as Brussels airport blast drama unfolds. HTTPLINK</p> <p>Latest photos from Brussels shootout show victim being carried out on stretcher HTTPLINK HTTPLINK</p>

Table 4.7: Examples of cluster themes.

4.4.2 SFPD for Classification Insights

This section considers the utility of [SFPD](#) used in conjunction with the outputs of classifiers.

During classifier active learning, the main goal is to find those features and documents which were they to be labelled, they'd improve the classifier's ability to learn the distinction between classifications,

and therefore its ability to isolate the classes of interest. Whenever the user submits a new feature or document labels, the user is presented with the model's current performance on the test data, which is a good guide for whether the classifier is better able to predict the test data. However, this is not the primary goal of the analyst; the analyst is testing a hypothesis about the data: does the text fit the class definitions?

Performance on the test data is certainly an indicator. A classifier that has excellent performance on the evaluation data is likely managing to draw the required distinction in the data. And a classifier that despite all efforts fails to achieve satisfactory performance on the evaluation data is potentially attempting a classification that the data does not support. However, this is less clear than with the well-performing classifier, since the classifier may be performing badly because the analyst has poor knowledge of the data, and/or has produced training labels that are ineffective. It is also even possible that the NBC is too simple a technique to capture the class definitions.

Furthermore, the evaluation data is usually small, 100-300 documents. Analysts generally do not have the time to create evaluation sets of thousands of tweets, and then create training data as well. Additionally, it is the exploration of the dataset that is the focus of this thesis, the process that absorbs most of the analyst's time. It is the analyst's *last task* to annotate a sufficiently large evaluation set to provide a convincing account of the accuracy of their study's conclusions. The analyst ideally should have an indicator of the success of their strategy long before having to annotate 300+ documents per classifier. So we should expect that users label a very small amount of data (e.g. 100) before proceeding to training for each classifier, in order to scaffold a viable strategy. The evaluation set then represents a very limited view on the corpus. And therefore, the indicator of how well the text supports a given classification definition is weakened.

Random sampling, together with the SFPD corpus insight approach in Section 4.4.1, as shown, helps to indicate what types of classification a corpus *might* support before classifier training has begun, which alleviates this problem. However, it is also possible to approach the issue from the other side: given a classifier already in progress, what distinction is it finding in the text? The answer may seem obvious: if the classifier is performing well, then it is the distinction we defined it to make. But as discussed, the very small evaluation set can be a weak indicator. Therefore, a well-performing classifier may not

be doing what is intended, and an ill-performing classifier may be making a distinction that the training data supports, but the testing does not. And as mentioned above, even with a very good evaluation set, if the classifier is ill-performing, this is not a reliable indicator that the data does not support the intended class definitions.

METHOD52 already provides functionality that can help. The user can randomly sample from a given classification. That is, the model is trained on the provided data, and applied to the unlabelled data, then a uniformly random sample is drawn from those documents which the classifier has labelled with the category of interest. These documents provide a view on the actual distinction that the classifier is making on unseen data, again engaging the user with more data.

In the same way that SFPD supported random sampling across the entire corpus for corpus insights, it can be applied to a particular classification for class-specific insights here (recall Figure 4.9).

In order to assess whether a word occurs surprisingly frequently, there must be a notion of expected frequency. The expected frequency is defined by a reference corpus, which therefore influences which words will be considered surprising. If the WIKIPEDIA reference corpus is used here, it would be like selecting a classification for downstream analysis and taking the SFPD corpus insights approach, which is helpful for the reasons stated in the previous section. But different insights can be obtained by changing the reference corpus. The target data for SFPD remains the documents that the current model classifies with a specified category, but the reference corpus is built from all documents classified with *not* the target category. This alters the notion of surprisingness such that those features which occur surprisingly often in the target category compared to the other categories are the most surprising. This exposes features that are emerging as typical of a particular categorisation, that occur less frequently in other classifications.

Table 4.8 below shows a selection from the top 50 phrases generated using this method on the Brussels attack “consequence” category as the target, and the attack “update” category as the reference corpus:

Target “consequence”, Reference “update”
donald trump
ted cruz
racism
hateful
the scariest thing about brussels is our reaction to it ³
mosques
bigotry
racial hatred
islamophobic
flow in ⁴
lyin ted
is a liar
temporary ban on muslims
of radicalism in belgium

Table 4.8: Selection from top 50 phrases from output of SFPD on the “consequence” data with “update” as the reference.

At this point, the stream had not been divided into the three topics (Obama, Trump, & Islam), and yet these terms show not only the topics arising, but that criticism will play a key role in the consequences dataset in subsequent analysis. Therefore, before the downstream analysis has begun, the significance of Trump and Islam tweets is known. This technique is suggesting to annotate those features as indicative of “consequence” if we wish to get the most data possible on these issues. Even if the inclusion of such features does not necessarily improve performance on the small test set, this knowledge allows the analyst to ensure higher recall of relevant details for the downstream analysis.

This insight is all gained without the help of the evaluation set, and provides additional evidence that the classifier is making a useful distinction.

As is true for all feature discovery methods in this thesis, the features proposed are intended to be complementary to the high-IG features already proposed in the classifier interface. The features already proposed are very much tied to the current classifier model’s under-

³ This was a prevalent theme. Initial reactions were very anti-Islam, including a hashtag “stopislam”. This prompted a massive backlash of users criticising such discussions as Islamophobia.

⁴ The authors using this phrase tend to be making a point about Muslims entering their country in numbers larger than they deem desirable.

standing of the data. For example [SFPD](#) shows early on during the “Consequence” versus “Update” classifier that “Donald Trump” is a prominent feature, but since that bigram has too little bearing on the consequence/update distinction, it does not arise as a high-[IG](#) feature. Yet if we’d like to ensure Donald Trump tweets get classified as “Consequence” for further analysis, it would even be useful to label that feature even if it had no bearing on performance on the current gold standard.

“Barack Obama” did not appear in the top 50 ($\lambda = 0.4$) which is not especially surprising, since this key term produced the smallest dataset of the three Brussels topics, and the Obama supportive/critical classifier performed the worst.

If we were to use the default `WIKIPEDIA` data as the reference corpus, then from the above list only “trump” appears. The terms are instead less focused on those which are specific to the chosen classification, since each classification will typically share a certain amount with others, since they are drawn from the same dataset. It is impossible that the vocabulary of different classifications will be completely separable in any non-artificial dataset.

This method can also reveal phrases that should *not* be occurring under the given classification. This could be for one of two reasons:

1. *The phrase is actually indicative of documents that should be in one of the other categories.* This is especially common in the early stages of training, since the training data has not yet produced a well-performing classifier. This is useful information, however, because it gives the user a phrase to mark as indicative of that other category, since in order for the phrase to be surprisingly frequent, the classifier must be making similar mistakes repeatedly over that phrase.
2. *The phrase is indicative of documents that should have been filtered out already by classifiers earlier in the analysis.* This is indicative of errors made by previous classifiers. It also suggests that the error occurs more often in the selected classification, and could suggest either the need to include another “other” category, or to go back and attempt to improve performance of the erroneous classifier using this new feature knowledge. The user may be more likely to find such phrases when using a more general reference corpus, since the previous classifier’s mistakes could be spread more uniformly across the current classes. Either way,

this helps to *fail-fast*; the analyst will know early on that the preceding pipeline components are failing in a key area, and must address the problem before committing all effort to completing the current classifier.

This is very different knowledge compared to the high-*IG* feature discovery, which will focus on those terms which reduce classifier uncertainty most across the current classifications, instead of features which happen to occur surprisingly often in data under a given classification. This is another reason for the approaches being complementary.

But most critically important for the goal of better enabling researchers to better explore and isolate classes of data, this strategy provides another simple, but useful view on the data, since it provides more information on how the data fits their class definitions at a strategic point in the analysis. It also provides further opportunity to engage with the data using more than just small samples of individual documents, collected in whatever manner. Section 4.5 introduces functionality which allows the user to inspect a random sample of contexts of actual features used by the classifier, directly in the classifier interface. Therefore, upon the discovery of interesting terms, the user can rapidly direct random sampling efforts to specific areas of the data defined by the features that occur in them.

The approach begins to report less useful results when the corpora representing each category are much smaller; see Section 4.4.5 for more discussion of this problem.

4.4.3 Domain Adaptation

Section 3.7 discussed the general strategy of “establishing the expectation” for *SFPD*: the reference corpus is changed to alter which words in a target set seem most surprising. And Section 4.4.2 described a strategy that selects target and reference data according to specific classification definitions in order to discover features that are classification-specific. Altering the reference corpus is essentially a form of *domain adaptation*. The reference corpus defines our expectation of word frequencies, and therefore influences how surprising words are in a given target dataset. Therefore, the reference corpus can be tailored with prior knowledge of the target domain to down-weight terms that are known to be irrelevant. This section expands

on Section 3.7 and demonstrates the importance of the strategy for classifier training.

The simplest example concerns different languages. If the reference corpus is English, then most words in Arabic target data would seem to occur surprisingly frequently, since the expectation is defined over English words. These overestimates of surprise would obviously hinder analysis by including many frequent Arabic words that would not be surprising to the analyst expecting Arabic words. This reveals how important selection of the reference corpus is in general. Regardless of how good the measure of surprise or process of phrase extraction is, if the statistics upon which the expectation is based are inappropriate for the domain, then the technique can be useless.

The `SFPD` tool is built with the ability to incorporate custom reference corpora. Therefore, a reference corpus can be selected which defines an expectation which diminishes the surprisingness of terms which the analyst knows (from domain knowledge) are uninteresting in their study. Therefore, in order to down-weight commonly occurring Arabic words that should not be surprising, an Arabic reference corpus should be used. This strategy was adopted by (Conway et al., 2017) and Italian collaborators of `CASM CONSULTING`.

A different example that `METHOD52` is built for is `TWITTER` chat language. When analysing the content of `TWITTER` discussions we are rarely interested in interjections and informal variants of words. However, these terms can occur in tweets much more frequently than one would expect in `WIKIPEDIA`. Therefore, using a `WIKIPEDIA` reference corpus can raise the surprisingness of the chatty terms. To alleviate this problem, a `TWITTER`-based reference corpus should be chosen. A simple strategy for producing a more `TWITTER`-aware reference corpus is to use `TWITTER`'s sample `API` to generate a reference corpus. Streaming from the sample `API` acquires a 1% sample of all tweets in real-time. Therefore, it constitutes a broad sample of `TWITTER` language.

The difficulty with this strategy is avoiding too much bias in the sample. A sample collected over a single day, will doubtless be dominated by the day's trending topics. If the reference corpus is predominantly documents discussing the trending topics, the expectation generated for the surprisingness measure is not suitable as a general model of `TWITTER` language. Instead, given that the expectation is informed by the trending topics, `SFPD` would be directed to consider surprising any phrases that are less represented in trending tweets,

which could be quite narrow representations of overall TWITTER language. This implies that phrases (simply by virtue of not being part of the more popular trending topics) will have their surprisingness exaggerated. The influence of trending topics can be minimised by taking a sample over a longer time period, since over a period of months there will have been many trending topics accompanied by an accumulating stock of more general documents.

In other contexts, restricting the reference and target corpora to particular time periods is actually desirable. In the weeks leading up to the 2017 UK general election, starting from when the election was announced, METHOD52 users at DEMOS⁵ collected tweets from TWITTER users who were identified as supporting a UK political party (using the user profile description and their tweets). The analysts wished to explore discussion at the beginning and end of the election period. SFPD was applied ($\lambda = 0.4$) to lead the analysts to the important classes of discussion. The users were especially interested in how the discussion changed from the beginning of the election campaign compared to the end.

It was possible to emphasise this content by using a two part experiment: in the first part, the target set was constructed from the first week of tweets (from the election announcement), and the reference corpus was constructed from the remaining data. In the second part, the target set was created from the last week of the data, and the reference was constructed from the remaining data. These experiments were also split by political party, considering in the target data only those tweets from users aligned with a single political party. This method permits the separate analysis of content by party. The phrase list generated in the first part emphasises those phrases that occurred surprisingly often in the initial phase of the election period compared to the rest of the period. And conversely, in the second part, the list emphasises those phrases that occurred surprisingly frequently at the end of the period.

The analysts found that while the lists were fairly noisy, they managed to find at least 120 phrases of interest per party per time period. They used the phrases to index back into the data to analyse discussions surrounding the key phrases. Observations were made such as there being a focus by Conservative supporters on terrorism at the start of the period and Brexit at the end.

⁵ <https://www.demos.co.uk/research-area/centre-for-analysis-of-social-media/>

This type of strategy is appropriate anywhere that differences in phrase usage over time is of interest.

4.4.4 *Irrelevance Filtering*

[SFPD](#) analysis can also help to identify noise that may need to be filtered out by a classifier or simply keyword detection. Surprisingly frequent terms that aren't relevant to the topic of interest may be indicators of classes of documents that can easily be set apart by their common vocabulary, helping to reduce the need for the "other" category in our classifier-based analysis. Section [3.8](#) covered the use of [SFPD](#) to identify phrases whose presence can be used to filter away irrelevant documents, and shows how the approach can aid classifiers in a `METHOD52` pipeline.

The strategy can be directly incorporated into the classifier training procedure, in any classification task where there is an "irrelevant" or "other" category. This scenario is introduced in the Brussels dataset analysis (Section [4.2](#)) as the "none of the above" problem, where because there are limitless possibilities for features indicative of irrelevant documents, it is very difficult for the analyst to know which features to label to help the classifier learn this category.

This is a serious problem, because feature labels are what the active learning process requires in order to utilise unlabelled documents (see Section [4.1.2](#)). Therefore, having no features labelled under the irrelevant class would lead to a large bias in training data, which would diminish performance of the model.

Using either the corpus insights (Section [4.4.1](#)) or classification insights (Section [4.4.2](#)) [SFPD](#) method with either irrelevant documents as the target data or all documents being classified, the user can inspect the generated phrase lists for clusters of terms that indicate topics that are not relevant to their analysis.

This is useful when portions of the irrelevant data form at least loose topics, i.e. the irrelevant documents exhibit some amount of vocabulary overlap. If the irrelevant documents do not share enough vocabulary, then their phrases will *not* be noticed as surprising, and they cannot be filtered merely on the presence of set of phrases.

Application of this method is simple for static datasets (or short-term collections). But if it is used in the context of an ongoing collection of documents, the classifier may need additional training in order to be useful for future unseen data. This is because the longer a real-time

stream of documents is collected, the more new topics (and therefore new vocabulary) will arise. Therefore, labelled features focusing on irrelevant topics that were present in the original data will be less effective at identifying new irrelevant data. The method would not be appropriate at all if we were constructing `METHOD52` pipelines that would repeatedly apply to different generalised problems, but recall that what is being considered here are largely bespoke studies, where the aim is to best explore a given dataset. A fixed domain/dataset will contain finite amounts of irrelevant data, which may fall into irrelevant categories of data which can be filtered away.

Section 4.5 introduces new functionality to the classifier user interface: a tool that permits users to take a random sample of the original context documents of a specified feature. Analysts are encouraged to utilise the tool here for potential irrelevant phrases, since seemingly obvious terms could in actuality be false friends. Below is an example from the Brussels dataset found after examining the original contexts of the `SFPD`-proposed irrelevant term “brussels sprouts”:

I have been more correct than anyone on terrorism. Terrorism in Brussels sprouts without a ban on Muslims.

The ambiguity between the city and the vegetable was frequently exploited in wordplay forming actual criticism of the topics of interest, showing that even seemingly totally irrelevant terms could be misleadingly relevant.

4.4.5 *Vocabulary Overlap*

The `NBC` performs classification by recording the frequency with which features occur in documents with each classification. Assume for this argument that there is an equal number of documents in each category of training data, since the classifier will also take this balance into account. In the extreme hypothetical scenario that exactly the same features occur in documents of each classification in the training data, the classifier would then have no decision power; any document at classification time would appear to be equally indicative of each classification. In the opposite case, where the vocabulary of features for each category is mutually exclusive, the classifier has most discriminative power. Therefore, when inventing classification tasks and expecting the `NBCs` to take to them well, not only must the user determine that there is enough data to support this, but also that

the training data that they create produces classification vocabularies that are separable enough to provide the classifier with sufficient decision power.

This issue is typically not well understood by those who are not familiar with the algorithm, a problem that is likely made worse by lack of explicit importance of surface features. `METHOD52` users should not be expected to understand the detailed mechanics of the algorithm. But a good grasp of this issue enables much more effective classifier training. The challenge is to give a practical understanding of how to manage this problem to analysts, and how to use it as an opportunity to aid data exploration.

The section will use the Brussels data subset defined by personal reactions concerning Donald Trump, since the supportive/critical classifier for this topic performed best of the three. Therefore, we should expect a fair amount of discriminative features.

One approach is to directly compare how discriminative each feature is in the classifier's vocabulary, by ranking features according to the ratio between their most and least likely probabilities conditioned on the classes. Therefore, if a feature is most likely in classification c_1 and least in c_4 , then its ranking score would be:

$$\text{score}(\text{feature}) = \frac{P(\text{feature}|c_4)}{P(\text{feature}|c_1)}$$

Probabilities should either be smoothed to avoid zeros, or classifications for which the probability is zero could be excluded.

For unigrams, this approach is equivalent to ranking the combined results of applying `SFPD` (without phrase expansion) to all pairs of classifications where in each pair one is the target and the other the reference; $\lambda = 0$ so that only the ratio between target and background likelihoods is in effect.

While this simple approach technically produces the features that are most discriminative for a given pair of classifications, it does not always produce practically informative features, because it is overly biased toward infrequent features. A feature that occurs 5 times in total, and just happens to occur in 4 supportive documents and only 1 critical, will seem incredibly discriminative; the more infrequent a feature the more likely it is that our sample does not adequately represent the feature's true distribution over the categories, and that classification errors could easily lead to misrepresentation with so few examples. It is the same problem that affects `SFPD` relying solely on lift ($\lambda = 0$), which is unsurprising given their equivalence. Furthermore,

practically speaking, if a feature is very infrequent, then its degree of overlap over the classifications is less important, since it will not be frequently encountered when analysing documents.

For example, the most discriminative unigram according to this score in the Trump personal reactions data was “suggest”; it occurred only 20 times in total. And manual inspection revealed that the classifier failed to label 7 of them as supportive. This problem would be exacerbated with the use of more sparsely occurring bigrams and trigrams, which suggests using SFPD’s phrase expansion method.

In order to put less emphasis on low frequency features, the λ interpolation can be re-introduced, an approach which is equivalent to the “SFPD for classification insights” approach (Section 4.4.2) for two-class problems. The reason the approaches are not equivalent for problems with more than two categories, is that this approach makes all pairwise comparisons between classifications, whereas Section 4.4.2 suggests splitting the data one-versus-the-rest; the target set is the classification of interest, and the reference is constructed from all other classifications.

After applying this approach with various higher values of λ (0.3, 0.4, 0.5, 0.6, 0.7), some useful discriminative features arose in the top 25 frequently, mostly supportive terms:

vote trump (supportive) users urging people to vote for Donald Trump in the next election.

back in january (supportive) users recall that Donald Trump gave warnings about Brussels

elite (supportive) users supportive of Donald Trump make derogatory comments about “The Elite”

Despite the increased λ , very infrequent features are still featured heavily. This could be due to the fact that once near-duplicate tweets are removed, the dataset is reduced to 30791 documents. Once split between supportive and critical, the remaining documents constitute very small reference corpora for the other classification. The small reference corpora could define expectations that are easily surprised. The classification insights approach is also vulnerable to this problem as the classifier filters produce smaller subsets of the data.

While with sufficient data this approach could be useful for vocabulary comparison (Section 4.4.2 demonstrated how the output describes the vocabulary of a given classification), the small dataset pro-

duces disappointingly few interesting discriminative features for a classifier that performs fairly well.

There is an alternative approach, which shares similarity with both corpus and classification insight approaches. As before, the documents in each classification each constitute separate target sets. But the reference corpus is kept the same: the entire Brussels dataset before any filtering. This produces a phrase list for each classification, where the reference corpus is much larger and more stable, but still with enough information to discount many features that are simply frequent across all classifications. There may be datasets where a WIKIPEDIA reference corpus may be sufficient, but the sample from which the classification documents were drawn will be more similar.

The phrases in the lists are now no longer necessarily just the discriminative phrases, because they are not directly compared across classifications. Instead, phrases are more likely to be discriminative if they *do not appear on more than one of the per-classification lists*. The phrases below were produced using this method (selected from the top 25, $\lambda = 0.3$):

Critical of Donald Trump

a hell hole Donald Trump referred to Brussels as a hell-hole, which provoked much criticism, but also much support (see below).

a hellhole

trumpster Often used as a derogatory term for supporters of Donald Trump.

nato Trump made comments about NATO that were heavily criticised.

campaign Trump's critics worried that the terrorist attack would bolster his campaign.

incites hatred violence banning all muslims Many accused Trump of inciting hatred against Muslims.

Supportive of Donald Trump

vote trump

warned about brussels back in january Trump made a warning about Brussels that his supporters felt important.

mr trump The honorific is most often included by Donald Trump's supporters.

trump predicted

leader Often used when suggesting that Trump is the type of leader the US needs.

mocked him for it Used when criticising Trump's critics for mocking him.

cruz Ted Cruz was often criticised in support of Trump.

president Similar to usage of "leader"

trump2016

donald trump brussels it's like living in a hell hole

is the only candidate who

a hellhole A term shared heavily between supportive and critical tweets.

References to Brussels as a "hellhole" were shared among both critical and supportive tweets. Donald Trump had first used the description, and it was taken up by both supporters in encouragement, and others in criticism. Below follows a typical example of each:

- Trump was mocked for calling #Brussels a hellhole, he was right.
- Wtf you know bout Brussels Donald Trump, we'll see which place will turn into a hellhole or a disaster once you get elected...

With this approach, when phrases occur surprisingly often in more than one category list, it is likely that those phrases represent vocabulary overlap across the categories, since they occur surprisingly frequently in several categories. Otherwise, if a phrase occurs in only a single category list, then it could be a phrase that is particularly indicative of that single category. Therefore, this strategy provides a useful alternative for determining vocabulary overlap, especially when the classification corpora are smaller.

4.4.6 *Using Discovered Phrases in the Feature Model*

All Section 4.4 SFPD techniques generate insight about the dataset and its categorisation. But they can also be used to influence the actual classifier model.

For unigrams, bigrams, and trigrams, this is trivial; the user can use the feature labelling interface by inputting the relevant ngram. The feature would receive pseudo-counts, and the NBC would treat it as that much more indicative of the labelled class. This is possible because the feature extraction process extracts unigrams, bigrams, and trigrams during training and classification. However, if the user were to label a 4-gram, since the feature extraction process would never produce 4-grams, the classifier would never be able to use the information. It is impractical to extract all 4+ ngrams just to accommodate the small list of phrases that the user labels.

Instead, there is an alternative method for longer phrases. When the user specifies a phrase longer than the currently extracted ngrams, it is added to a list of custom-size ngrams for the system to look for. When the specified sequence occurs in the data, it is extracted also as a complete ngram with the custom size. These rules are fulfilled by applying an efficient, well-established multi-pattern string matching algorithm (Aho and Corasick, 1975). Using the algorithm, given a list of phrases to search for, we can efficiently determine where each phrase occurs in documents, so we can treat them as single tokens and model them just as any other feature in the classifier model.

Unfortunately, in practice, it is rarely necessary to enter phrases of more than three tokens in size. Frequently, when the analyst would need a 4-gram, an inner trigram is distinctive enough to not require the full 4-gram. If the trigram is sufficiently distinctive, it is unnecessary to risk the loss of generalisability with the added sparsity of the 4-gram feature.

4.5 ORIGINAL FEATURE CONTEXTS

This section turns back now to the existing high-IG feature querying and labelling aspect of the NBC active learning system, and improving it in order to better isolate classes of data using this feature discovery mechanism.

Wibberley et al. (2014) find that an NBC can often reach a good level of performance on tasks such as determining relevancy, topic, and

subjectivity with a suitable amount of training data, using unigrams and bigrams as features.

However, there are classification tasks that require a greater level of care in terms of the labelling of features and documents, such as sentiment directed at a particular target, as in the critical/supportive task in the Brussels dataset (Section 4.2).

In general, a classifier's performance is often enhanced with the inclusion of prior knowledge, and when there are only a few training documents (as is often the case in the active learning scenario) prior knowledge can greatly boost performance (Melville et al., 2008, 2009; Settles, 2011). This was one of the original motivations for including the feature labelling functionality in DUALIST, because it allows for the rapid development of classifiers. This highlights the importance of including prior knowledge.

Prior knowledge need not be immediately applicable to the training data, it is possible to add to the classifier's ability to generalise by including it. Even though, as stated previously, we're most concerned with bespoke analysis, so generalising to completely different domains is not necessarily required, recall that in this scenario the evaluation set is often small and potentially unreliable, or subject to change throughout the early to mid stages of analysis. So generalising beyond the current evaluation set is still useful. Furthermore, it is useful to be *minimally bespoke*, to make the analysis only as bespoke as it must be to acquire the necessary insights, so that, for example, the study might be more easily adapted to a different domain, with minimal additional work to adapt.

Despite the usefulness of feature labelling, we found that for many tasks annotators did not make much use of the feature labelling facility. Two main categories of problem were determined:

Annotator awareness It is difficult to determine whether a feature is indicative of any particular classification because alone it does not contain enough information about the ways it is used in the dataset. And worse, the annotator's idea of how a feature is used is often at odds with how it is actually used in the dataset. So labelling the feature would damage the model's performance.

Conditional indicativeness A feature can be indicative of multiple categories, depending on how that feature happens to be used. Therefore, it must go unlabelled, or labelled as always indicative

of all of the concerned categories (this may be a better alternative if there are other additional categories to distinguish from).

Returning to the Father’s Day dataset introduced in Section 4.3, a unigram feature that may be proposed to the user is the word LOVE. Our intuition might be that LOVE probably occurs mostly in tweets where the author is expressing love toward their father, and is therefore an appropriate target for marketing. However, sampling some of the original contexts, the user can see the following:

1. Fathers Day is Coming Up, lord knows I love my Dad.. I Gotta Get Something Special..
2. Trust us...Your Dad would love a bottle of American Revolution Vodka! #fathersday #vodka #giftidea
3. Father’s Day is coming fast! Get dad some Doux South pickles! Seriously, dads LOVE pickles! (Almost as much as they love their kids)
4. Does Dad love golf? Here’s a nicely themed Father’s Day gift.
<http://t.co/eZGMcxXgeG> #Delivery

From this sample, marketing tweets seem to dominate usage of LOVE, contrary to the initial intuition. This is a scenario that frequently occurs. In the Brussels dataset, the hashtag “#stopislam” would seem to be *obviously* critical of Islam. However, a large sample of its original contexts reveal that it is used more frequently in tweets expressing abhorrence for the hashtag. This demonstrates that a facility for viewing the contexts in which a feature occurs is vital.

Functionality was added which allows the user to inspect a random sample of the original contexts of any feature in order to raise annotator awareness of this issue. Furthermore, the original context documents can be labelled for training as well. This is particularly useful when the candidate feature is not revealed to have the indicative power that the user was hoping for. Instead, the user labels some of the original context documents in order to encode in the model some ability to deal with the contexts in which it usually occurs.

The original contexts of LOVE also reveal a problem of *conditional indicativeness*; the feature LOVE seems to be able to be indicative of both “suitable” and “marketing” depending on its context. At least in this case, the classifier has a chance at using LOVE to distinguish between “sad” and the other two categories if the user labels it as

both indicative of “marketing” and “suitable”. The situation would be worse still if LOVE was found to be indicative of “sad” tweets also, because the user would be unable to do any labelling. But more context-rich features may enable us to distinguish between the uses of LOVE. This problem is addressed in Section 4.6.

As one of the earliest new features introduced to METHOD52, the original contexts display has been in regular use since August 2014. Users report two major improvements in their interaction with feature labelling:

- Features that they previously would have labelled were instead left unlabelled upon discovering that the original contexts presented usage that they had not expected. Therefore, without the display of original contexts, they would have labelled features that would have likely been a detriment to performance.
- The list of high-IG features presented by the system usually contains many features that they could not imagine being indicative of either classification until the original contexts were inspected. Many useful features were discovered simply by inspecting the original contexts of each feature in the list.

The drawback with this new method is that it increases the time it takes for users to decide whether to annotate features. This means that there is potential to increase the time it takes to complete classifier training. The intuition, however, is that the above improvements lead to a viable, well-performing classifier sooner than one would otherwise expect without it, because it is the feature labelling that is responsible for leveraging the unlabelled documents, and for rapid classifier building (labelling features is faster than labelling whole documents). Furthermore, since uninformed features are more likely to negatively impact performance, the user with more awareness is less likely to be required to backtrack and re-examine their annotation decisions, or discard training labels and begin again. The problem of conditional indicativeness, however, remains. Section 4.6 examines an example case and introduces a methodology to tackle the issue.

4.6 ENCODING CONTEXT WITH SYNTACTIC NGRAMS

Section 4.5 introduced the problem of *conditional indicativeness*, where the same feature can be indicative of multiple different classifications. The section also introduced the problem of *annotator awareness*, in

which the user’s understanding or interpretation of a feature’s indicativeness is flawed. While Section 4.5 sought to raise annotator awareness by the inclusion of a service which reveals a sample of the original contexts of a feature to the analyst, it mostly left the problem of conditional indicativeness untouched. This section reasons why incorporating a feature extraction method for the classifier based on dependency parsing trees may alleviate the problem.

LOVE in the Father’s Day dataset was shown through four example sentences to be used in two different manners:

1. In “suitable” tweets, where LOVE is used to express positive sentiment toward the author’s father.
2. In “marketing” tweets, where the author describes some promotional product that they are suggesting fathers will love.

If we were to also extract bigram features, then the last two context sentences (3 & 4) would happen to produce the features LOVE_PICKLES and LOVE_GOLF, which are more likely to indicate marketing tweets. But if there are any words in between (as in “...love a bottle...”), the bigrams fail to produce the desired features. We’d have to resort to trigrams to connect “love” to “dad” in the first sentence (LOVE_MY_DAD), which introduces needless feature sparsity, as the feature LOVE_DAD would be sufficient. Any time that the word “love” is applied to “dad”, a feature LOVE_DAD would be ideal, regardless of any words like “my” sitting between them. We’d also have to resort to trigrams for LOVE_A_BOTTLE, which has the same problems, and does not generalise to sentences with different or more determiners/modifiers between the two words (e.g. “...love this bottle...”, “...love a nice bottle...”).

A technique that would produce these features is the extraction of syntactic bigrams using dependency relations. In order to create a normal bigram, two tokens are joined which are adjacent to one another in terms of their order of occurrence within the utterance. However, in order to create a *syntactic* bigram, specifically a syntactic bigram constructed from a dependency relation, two tokens are joined which are directly connected via a dependency relation.

Given the dependency tree in Figure 4.10, the following are all the possible syntactic bigrams: DADS_ADORE, ABSOLUTELY_ADORE, ADORE_PICKLES, OUR_PICKLES. This produces bigrams that are more likely to represent meaningful pairings, since one element of the bigram will have been syntactically dependent on the other.

Two standard bigrams do not make it into the syntactic bigram set: DADS_ABSOLUTELY and ADORE_OUR, which isolated would seem ambiguous to the annotator. This shows that these features can also alleviate the *annotator awareness* problem from Section 4.5, by constructing features that are more interpretable.

The dependency trees of the four example Father’s Day tweets from Section 4.5 would produce the syntactic bigrams listed below, which are useful, generalisable features, that allow disambiguation between the “marketing” and “suitable” uses of the word LOVE.

1. LOVE_DAD
2. LOVE_BOTTLE
3. LOVE_PICKLES
4. LOVE_GOLF

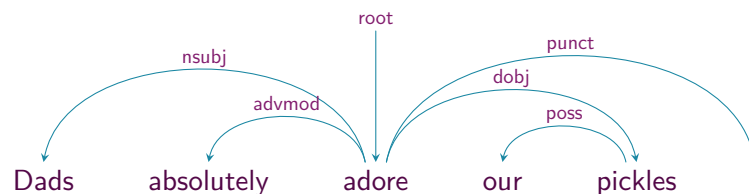


Figure 4.10: Dependency tree of the sentence: Dads absolutely adore our pickles. The dependency scheme is the Stanford dependencies, which is introduced, along with the reason for its use in this thesis, in Section 4.6.2.

Section 4.6.1 explores related work in using dependency features to improve performance in classification tasks, in order to inform the strategy adopted (described in Section 4.6.2) for utilising dependency features to alleviate the conditional indicativeness problem. Section 4.6.3 describes efforts to adapt dependency parsing techniques to the chatty language of TWITTER. Section 4.6.4 considers the technique’s impact on the METHOD52 classifier framework.

4.6.1 Related Work: Dependency Features for Classification

Having established above that dependency analysis would seem to provide useful information about the contexts of features, this section first discusses previous approaches adopted for incorporating dependency features into the feature model of a classifier, in order to

establish a sensible method for their inclusion in the classifier training. The focus is on subjectivity and sentiment classification, since these are fundamental use cases, and very well studied.

There is much research attempting to incorporate syntactic features into sentiment analysis (and other types of opinion mining) classifiers, mostly in the domains of film and product reviews (Agarwal et al., 2015; Joshi and Penstein-Rosé, 2009; Matsumoto et al., 2005; Dave et al., 2003; Wilson et al., 2004). However, there is less work that tries to utilise such features for the classification of tweets, with the exception of Jiang et al. (2011), who extracted particular relations between words (such as the relation between a transitive verb and its direct object) in the tasks of identifying subjectivity and, subsequently, the polarity of sentiment of tweets with respect to a particular query. Improved performance was obtained on both tasks (though more so on the first). Tweets are significantly different from reviews, such that even parsing them can be an issue, and therefore techniques which performed well or poorly elsewhere may produce very different results in the TWITTER domain.

Classification is studied at different levels of utterance. Some research focuses on identifying categories for entire documents (Gamon, 2004; Dave et al., 2003; Ng et al., 2006), some make classifications at the sentence level (Hu and Liu, 2004; Arora et al., 2010; Joshi and Penstein-Rosé, 2009), and others on a per phrase basis (Wilson et al., 2004, 2005). While technically in this scenario we are always producing a single label for the entire tweet document, our tasks are most similar to sentence-level classification. Tweets are not like movie or product reviews, because they do not consist of very many sentences, they are typically only a single sentence. This is due to their 140 character limit.⁶

METHOD52 uses the semi-supervised NBC described in Section 4.1. Three machine learning classifiers are commonly used in this area: the NBC, Maximum Entropy classifier and Support Vector Machine (SVM). In the movie review domain with these three classifiers, Pang et al. (2002) performed sentiment analysis, experimenting with unigrams, bigrams, adjectives only, other part-of-speech information, and sentence position information. The authors found that the unigram only model using the SVM performed best, and the NBC worst, but only by a small margin. However, many papers have since adopted the SVM approach, usually with a linear kernel function (Jiang et al., 2011;

⁶ Again, when this work was carried out, the 140 limit was in place. The new 240 limit, still permits only fairly short documents however.

Joshi and Penstein-Rosé, 2009; Gamon, 2004), though there are approaches that make use of tree kernels (Agarwal et al., 2011; Wu et al., 2009).

Interestingly, Xia and Zong (2010) reasoned and showed experimental evidence that NBCs could be better able to handle syntactic features than SVMs, since the NBC's independence assumption holds better with word relation features. They achieved even better performance with an ensemble model, in which an SVM handled unigram features and the NBC handled word relation features which were extracted from the dependency parse of the sentence. Go et al. (2009) showed that all three classifiers are capable of high accuracies when classifying sentiment of tweets.

Incorporating sensible syntactic features is not a simple task. Dave et al. (2003) extracted syntactic relations between adjectives and nouns as features, but did not manage to improve performance on sentiment analysis of product reviews. Ng et al. (2006) extended the approach of Dave et al. (2003) by including subject-verb and verb-object relation features; they managed to improve the performance of unigram classifiers, but note that they can achieve the same performance using unigrams, bigrams, and trigrams together. While in general this is a good case for abandoning the approach in favour of the simpler ngrams, this thesis is not only concerned with classifier performance; if the resulting dependency features improve the ability of the METHOD52 users to incorporate prior knowledge, either through being more understandable, or useful in the conditional indicativeness problem, then dataset exploration would be improved.

There is a general trend that those dependency extraction approaches which have been more successful tended not to have hand-picked specific relations (like subject-verb, or those between nouns and adjectives), but instead considered the entire set of relations and allowed the learner to generalise (Joshi and Penstein-Rosé, 2009). Below follows examples of work which managed to successfully apply dependency features without discriminating on the dependency label.

Gamon (2004) used phrase structure and dependency relation information as features in predicting customer satisfaction on very noisy customer feedback documents. A significant improvement over just using surface features (unigrams, bigrams and trigrams) was found.

Matsumoto et al. (2005) extracted frequently occurring dependency sub-trees as features, obtaining better performance than just using

unigrams, bigrams, and commonly occurring subsequences of tokens when classifying movie reviews as positive or negative.

Wilson et al. (2004) extracted certain features regardless of the dependency relation type, such as “leaf nodes” (words which have no dependants), features made up of a tree node and its children, and the standard binary dependency relations between words. They managed to find improvements on the task of predicting the opinion strength of phrases.

Joshi and Penstein-Rosé (2009) themselves found a small performance improvement on the task of identifying whether a sentence in a product review expresses an opinion, by deriving features from the standard binary dependency relations between words, again not hand-selecting any particular relation type. Xia and Zong (2010) with a similar method improved performance on sentiment analysis of movie and product reviews. Paramesha and Ravishankar (2015) improve upon bag-of-words sentence-level sentiment analysis (again on product reviews) by including dependency features.

Satapathy and Karnick (2011) were slightly more selective, in that they selected only those dependency relations in which an adjective or verb were present, but they still managed to find an improvement upon using unigrams and bigrams on the task of sentiment analysis of movie reviews.

Sentiment analysis methods often utilise word lists of sentiment bearing words. Dependency relations can show how the sentiment is propagated through a sentence, showing whether a term’s sentiment is negated, augmented, diminished, or inverted (Di Caro and Grella, 2013; Paramesha and Ravishankar, 2015). This approach is, however, too specific for our scenario, since classification into positive or negative categories is only one of the many classification problems METHOD52 could be applied to. And in these problems, it is rarely possible to find, or even construct a word list describing how indicative each individual word is of each category.

Buddhitha and Inkpen (2015) develop a system to solve a task similar to what METHOD52 users are often attempting: given a topic and TWITTER messages, determine the sentiment toward the given topic in the supplied messages. They argue that in order to determine sentiment toward a given topic, instead of just overall sentiment, we need to know how words depend on each other and therefore where their meaning applies. They extract features from tokens that are related to the topic words via dependencies. This task was defined in the work-

shop SemEval 2015, in task 10 subtask C. Other researchers identified the same reasoning and also made use of dependency features (Townsend et al., 2015).

Bigrams and trigrams are much more sparse features than unigrams, as are most types of syntactic features discussed in this thesis. Therefore, an aspect of research attempting to incorporate syntactic features is the procedure for dealing with feature sparsity.

One tactic is to generalise (“wildcard” or “back-off”) parts of the features themselves. In a relation such as `amod(fun, movie)`⁷, one of the words could be “backed-off” to its part-of-speech, e.g. `amod(fun, noun)` (Joshi and Penstein-Rosé, 2009; Xia and Zong, 2010), or even both words could be wildcarded (Gamon, 2004; Wilson et al., 2004). Also, the type of dependency relation could be omitted (Nastase et al., 2007). Depending on the task, other back-off options may also be sensible, such as replacing certain sentiment-bearing words with “positive” or “negative”, or identifying words which intensify, diminish, or negate the meaning of other words (Balahur, 2013). Furthermore, labels for how strongly or weakly subjective a term is can be useful (Wilson et al., 2005).

Another approach is to prune the extracted features via some type of feature selection procedure. After features are extracted, only the top N features according to some notion of predictiveness are retained. A few popular measures are χ^2 (Joshi and Penstein-Rosé, 2009), mutual information (Xia and Zong, 2010), information gain (Nastase et al., 2007; Xia and Zong, 2010), and log-likelihood ratio (Gamon, 2004; Ng et al., 2006; Satapathy and Karnick, 2011).

It is also possible to have the feature pruning process be part of the actual feature extraction process. Matsumoto et al. (2005) extract dependency sub-trees as features for sentiment analysis. They do not pre-specify sub-trees of interest, neither do they generate all sub-trees and then use a measure of predictivity to rank them. Instead, they use an algorithm to efficiently extract only frequently occurring sub-trees.

Given that METHOD52 leverages large amounts of unlabelled data as part of the learning process, a certain amount of data sparsity is combated there. However, the automatically classified data is less reliable than the human-annotated data, which means that additional procedures may still need to be employed.

⁷ “amod” means “adjectival modification”, and so `amod(fun, movie)` is a relation specifying that “fun” is being used to modify “movie”.

4.6.2 Approach

This section details the approach adopted to incorporate dependency analysis into `METHOD52`'s classifier model and training environment.

The previous work discussed above in Section 4.6.1 shows that an approach which does not heavily discriminate on the *types* of dependency relations may be more likely to succeed. Therefore, a general syntactic ngram extraction approach is taken, which does not try to extract specific relations, instead allowing all relation types.

Previous work also suggests that feature sparsity is a concern. Sparseness of features was reduced by not including the dependency relation types in the extracted features, so that there is only one feature `LOVE_DAD`, rather than all possible relation type variants `LOVE_DAD(NSUBJ)`, `LOVE_DAD(DOBJ)`, etc. Typed dependencies are also more difficult for the analysts to interpret, given that the social scientists and humanities scholars are generally inexperienced with dependency relations and their meaning.

Given that users are usually incorporating large collections of unlabelled data into the classifier model, this also provides more assurance that enough occurrences of features will be encountered.

With such a user-facing feature extraction process, where the user can see the features that are extracted, applying a feature selection measure and filtering low-ranking features may present a confusing scenario to the user, where features they expect to be able to use are not being extracted. Therefore, no feature selection method is currently applied to further reduce sparsity.

It may be the case that the `NBC` will perform better simply for using syntactic ngrams instead of the usual proximity-based ngrams. [Xia and Zong \(2010\)](#) reason and show experimentally that `NBCs` may cope better with word relation features, because they violate the Naïve Bayes (`NB`) independence assumption to a lesser degree than unigrams. However, unigrams are still kept in the feature model since without them performance drops significantly.

A dependency relation scheme was chosen which emphasises the dependencies between content words over function words: Stanford dependencies⁸. This is because analysts tend to be most interested in the content of documents and the meaning it signifies. This means that for most use cases of `METHOD52` classifiers, the user is inter-

⁸ https://nlp.stanford.edu/software/dependencies_manual.pdf

ested in content words. In the Father’s Day data, the existence of a relation between “father” and “is” is uninteresting, but the extraction of FATHER_ABSENT from “father is absent” is important. Figure 4.11 demonstrates this with the copula relation, where “father” is directly connected to “absent” despite being related through the copula.

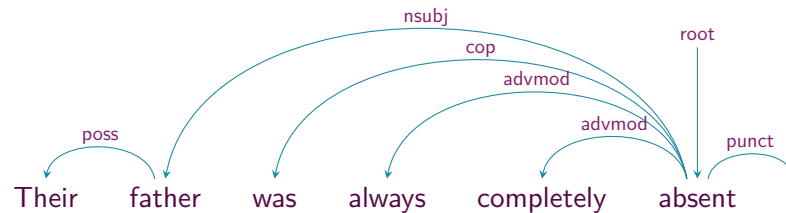


Figure 4.11: Dependency tree showing how Stanford dependencies attach the copula dependency. Note that “father” and “absent” are directly connected, instead of both being siblings with “was” as the head.

Furthermore, prepositional relations are collapsed to emphasise the content words to which the prepositions relate. Figure 4.12 shows a prepositional phrase in a dependency tree. Without collapsing, given that “gift” is not directly connected to “father”, a syntactic bigram would not pair the two, and GIFT_FOR is not that useful. Instead, when a preposition is encountered, the ngram extraction is allowed to recurse an additional time, so that GIFT_FOR_FATHER is extracted. If syntactic *trigrams* were extracted then NEED_GIFT_FOR_FATHER would also now be extracted.

It should also be noted that when ngrams are generated from the dependency tree, it is the original sentence order that determines the order of the tokens in the ngram. This is a purposeful choice, made to decrease the chance of the user misunderstanding the meaning of the ngram, which would exacerbate the *annotator awareness* problem.

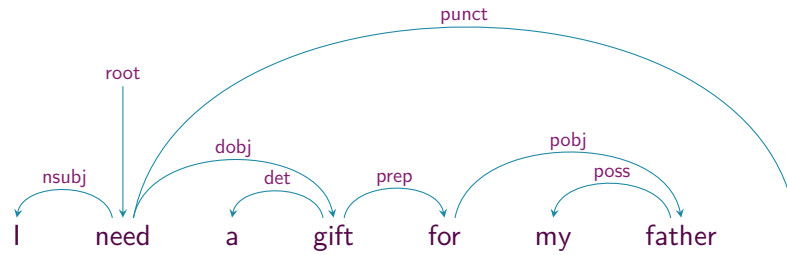


Figure 4.12: Dependency tree showing prepositional relationship. Without collapsing prepositional relations, “gift” would only appear in A_GIFT and GIFT_FOR. But if prepositional relations are collapsed, then GIFT_FOR_FATHER will be extracted; trigrams would also produce NEED_GIFT_FOR_FATHER.

4.6.3 Adapting Dependency Parsing to Twitter

In order to utilise dependency parsing in the active learning model and feature discovery, where the most prominent type of data is collected from TWITTER, the effectiveness of dependency parsing on tweets must be a consideration.

When creating a dependency parser for English language text, the usual source of training data is the Wall Street Journal (WSJ) section of Penn Treebank (PTB). When a parser trained on such data is applied to TWITTER language, it encounters words and sentence structure that are often very different from the training examples, which hinders parser performance.

Most dependency parsers rely heavily on Part of Speech (PoS) information about each word in order to make parsing decisions. And in order to match the style of the PoS tags and text in the PTB, the PoS taggers are usually trained on similar text. Therefore, before the parser even has a chance to encounter difficulties, it is often relying on poorly assigned PoS information for its decision-making, because the PoS tagger itself had poor knowledge of the language used in tweets.

In an ideal scenario, a large corpus of tweets would be marked up with both the correct PoS tags and dependency relations, so that the parser can be trained on in-domain data and therefore know how to deal with the language. However, marking up full dependency trees in addition to PoS tags for sufficient sentences, by several annotators in order to obtain high-annotator agreement would take thousands of man-hours.

In order to get the most out of the parser-produced features in classifier training, Sections 4.6.3.1 & 4.6.3.2 provide techniques which

improved the performance of an arc-eager transition-based dependency parser (Nivre, 2003) on TWITTER data. The parser was chosen for its computationally efficient processing time of $\theta(n)$ (where n is the number of tokens in a sentence), since it must be able to process large streams of real-time data. Additionally, transition-based parsers tend to be more effective at short range dependencies, due to their ability to deal with rich feature representations (Kübler et al., 2009, p.88). Given the shortness of tweets, this is ideal. An evaluation of these techniques follows in Section 4.6.3.3.

4.6.3.1 *Twitter-specific PoS tagging*

The TWEETNLP package (Gimpel et al., 2011; Owoputi et al., 2012) contains a PoS tagger specifically designed for TWITTER language; it can provide a more accurate PoS analysis of tweets than a system not trained on tweets.

Given that the dependency parser relies very heavily on PoS features, having the parser use the TWEETNLP PoS tagger is ideal. However, in order to adapt to TWITTER, Gimpel et al. (2011) devised a new PoS set, distinct from the PTB one. This presents a problem, because if the parser is trained on PTB tags, and at parse time is provided with TWEETNLP tags, these will mean nothing to the model.

Some tags are simply hierarchically related to PTB tags. For example, all types of PTB adjectives map to the single tag A. Every tag in PTB has an equivalent in TWEETNLP, many mapping to the same tag. Some definitions of tags are expanded; the authors chose to treat turns of phrase like “smh” (“shaking my head”) as interjections for example. However, some TWEETNLP tags do not have direct equivalents in the PTB. The tag L stands for a “nominal+verbal” such as “book’ll”. This arises from the philosophy that, since users on TWITTER aren’t writing highly edited, well-structured documents, they are likely to also simply write “bookll”, and that as such cases increase in complexity it makes less sense to attempt to split these terms.

Unfortunately, in order to utilise these PoS tags for parsing here, they need to be present at both the training and parsing stages. So a conversion tool was created that both adapts the PTB at train time to use the TWEETNLP equivalents, and during parse time adapts the TWEETNLP output to be closer to the PTB system, which involves splitting tokens tagged with any of s, z, y, L, M to more closely match how tokens appear in the PTB:

- L (nominal + verbal), M (proper noun + verbal), and Y (existential “there” & predeterminers + verbal) are split by examining the ending of the word for known verbal endings (like “ve”, “ll”, etc.) and examining the start for pronouns. Special cases are made for terms such as “imma” which can translate to “I’m a...” or “I’m gonna / going to” depending on the PoS that follow.
- s (nominal + possessive) and z (proper noun + possessive) are split by attempting to find an “s” (whether preceded by an apostrophe or not).

The converter also needed to be aware of common misspellings.

The converter maintains a mapping from the pre-converted tokens to the converted, so that after parsing the tokens can be collapsed back to their original forms if desired.

A few years after creation of this tool, TWEETNLP now includes a TWITTER dependency parser trained on a small sample of hand-annotated tweets (Kong et al., 2014), which *could* perform the tasks required of this adapted parser, but it is a graph-based parser, and does not provide dependency types other than multi-word expressions and coordination, which are required for preposition collapsing and future work on filtering and selection of dependencies by type. Furthermore, given that we’re using a transition-based model that can make great use of previous parsing decisions as features for making decisions later in the sentence, typed dependencies provide more informative features for this purpose.

4.6.3.2 Stripping non-syntactic elements

Tweets contain many tokens that are not part of a traditional sentence’s syntactic structure:

URLs employed by users to link to other tweets, or external sites.

These links can appear anywhere in the text, and do not affect sentence structure.

Emoticons symbols, or characters which together signify a symbol are used to add mood to a tweet, but they are not part of the syntax of a sentence. For example “:-D” represents a laughing expression or one with a large smile. There are some exceptions. For example, heart symbols can be used instead of the word “love”: “I ♥ Esperanto”. In this case, the symbol is being used as a verb, therefore it is part of the sentence’s syntax.

Retweets Before TWITTER changed its API, when a user would retweet⁹ someone’s post, TWITTER would insert its own special syntax to demonstrate that the tweet is a retweet. These characters bore no meaning in the sentence’s syntax.

Hashtags primarily used to mark the topic of a tweet, e.g. *“still shocked bout Amelia. #xfactor”*. Here X Factor is marked as the topic so this message can be grouped with other X Factor messages, but the hashtag is not part of the sentence. However, sometimes the topic is marked succinctly, by making the hashtag part of the sentence: *“Does anyone else think that Amelia that was on the #XFactor looked like Charlotte Church?”* In this case, the “#XFactor” is obviously being used as a noun, and thus is part of the sentence’s syntax.

Given that the goal for the dependency parser in this thesis is to enable the extraction of features that encode information about how words related to each other, these non-syntactic features that do not relate structurally to other words are not needed. Furthermore, they are noise that hinders the performance of the parser, because they will not have been present during training on the [WSJ](#).

To avoid this problem, non-syntactic tokens are stripped out of sentences before text is input to the parser (including retweet characters, for backwards compatibility to older datasets). The [PoS](#) tagger in Section [4.6.3.1](#) is itself capable of identifying emoticons, URLs, and hashtags. Additionally, it attempts to distinguish between the syntactic/non-syntactic use of hashtags and emoticons by tagging them just like any other word if they are used syntactically, or with a tag set aside for non-syntactic usages. Therefore, the output of the tagger is used to strip non-syntactic tokens.

When using syntactic bigrams & trigrams, we still continue to use unigrams in the classifier’s feature model; these features contain a lot of information. In this way, we do not lose the ability to use tokens such as emoticons during classification, which for example are good indicators of sentiment ([Hogenboom et al., 2013](#)).

⁹ To “retweet” is to re-publish someone else’s tweet with optional additional text, maintaining a reference to the original tweet.

4.6.3.3 Evaluation

In order to gain an indicator as to whether the above enhancements to the dependency parsing strategy improve TWITTER parsing capability an evaluation was conducted.

It is prohibitively difficult and time-consuming to create a large set of full hand-annotated dependency trees. Instead, the precision and recall were evaluated for a subset of commonly used dependency relations (see Table 4.9) over a collection of 254 sentences randomly selected from tweets collected on the hashtag #XFACTOR during the 2011 airing of the eighth series of The X Factor.

The TV singing contest is incredibly popular (it was a trending hashtag) so it provided a large sample of data, but data that, due to the influence of the topic, tended to contain some useful discussion, which perhaps a random sample of all tweets might not provide.

Dependency	Description
NN	noun compound modifier
ADVMOD	adverb modifier
DET	determiner
PREP	prepositional modifier
AMOD	adjectival modifier
NEG	negation modifier
PRT	phrasal verb particle
TMOD	temporal modifier
POSS	possession modifier

Table 4.9: The dependency relations used when evaluating the dependency parser on tweets.

Table 4.10 shows that on average much precision is gained by including the the techniques from Sections 4.6.3.1 & 4.6.3.2, at the expense of very little recall, leading to overall gains in f-score. Performance is first shown without either of the improvements (the PTB TAGS method), followed by performance having stripped non-syntactic elements (STRIPPED & PTB TAGS), followed by performance of the full system using the TWITTER-trained tagger and stripping non-syntactic elements (STRIPPED & TWEETNLP TAGS).

Parser	Precision	Recall	F ₁ Score
PTB TAGS	55.9	71	62.6
STRIPPED & PTB TAGS	63.6	71.9	67.5
STRIPPED & TWEETNLP TAGS	68.4	70	69.2

Table 4.10: Average precision and recall across the relations in Table 4.9. Tagging using PTB tags was performed using the Stanford PoS Tagger (Toutanova et al., 2003).

With this increased precision (and f-score), higher quality syntactic ngrams will be proposed to the user during active learning than would otherwise be possible. More specifically, with the baseline precision of 55.9% the rate of identifying false dependency relations (i.e. false discovery rate) is 44.1%. This means that, for the baseline, we would have expected to see syntactic ngrams proposed based on incorrect dependencies 44.1% of the time. However, for the proposed method, this is 31.6% of the time, a reduction of 12.5 percentage points. This means that 28.3% of the times that the baseline method would produce ngrams based on faulty relations, the proposed method would not. This still leaves room for future improvement, since 31.6% of the time, proposed ngrams will still be based on incorrect relations. However, we would generally expect noticeable errors to happen far less frequently than this, since the dependency *type* is not revealed to the user; the proposed ngram only suggests that there exists *some type* of relation between the words, so the parser need only be correct about the existence of some dependency, not necessarily which dependency.

4.6.4 Impact on classifier training process

The user is now able to elect to extract syntactic bigrams and trigrams instead of normal proximity bigrams and trigrams in the classifier training interface. This section details the how these changes impact the classifier training process.

The main drawback of syntactic ngram features is the time it takes to extract them during startup of the classifier component. With an unlabelled dataset of around 100,000 tweets, the latency before the user can begin training as the system extracts features from all the data was previously a couple of seconds. However, when 100,000 documents require PoS-tagging and dependency parsing, the process is slowed. The delay can be nearly two minutes.

The delay is prevented from slowing every submission of labels, by ensuring that the feature extraction is only ever done once. The system then maintains a feature-extracted version of each document, to avoid having to re-process.

In addition to the delay caused by parsing text, the parser takes approximately 15-20 seconds to start up, as it deserialises its large model. This delay is avoided for the user by running the dependency parser as an always-on parser service, which the classifier component contacts over HTTP. The parser service can parse sentences in parallel. By running a single parser service, instead of loading a parser for every classification job, the system saves greatly on memory, since the parser model is over a gigabyte in size.

It is unfortunately impossible to avoid some slowdown, since the extraction of these features is a more complex process than simply extracting proximity-based ngrams. This does cost the user some patience, which could lead to less ready take up of the feature.

Visually the syntactic ngrams tend to be more understandable than proximity ngrams. The syntactic ngrams are more likely to present combinations of words that function together, and make sense when read in isolation. Presenting more understandable features should lead to more use of the feature labelling interface, which would lead to more utilisation of the unlabelled data, and more rapid classifier creation.

The screenshots in Figure 4.13 compare proximity and syntactic features proposed by METHOD52 for the “News” classification of the “News/Personal” classifier in the Brussels study (the classifier which tries to distinguish between breaking news and personal opinion). The classifier was first built using proximity ngrams, then in order to get comparison syntactic ngrams the same training data was used to train a classifier using syntactic ngrams (including unigrams in both cases).

Annotations on Figure 4.13 show some of the obvious differences between the two lists. Ngrams marked yellow contain a URL, which (like emoticons and TWITTER markup) are not syntactically related to words in the sentence. If the analyst determines that these non-syntactic elements are not modifying words in useful ways (i.e. they’re only useful as unigrams), then the syntactic ngrams will be more useful.

Ngrams marked orange as those which have dangling prepositions on either side of the ngram. These are not permitted in the syntactic

ngrams. E.g. instead of splitting training evidence across “to paris” and “linked to”, only the full ngram “linked to paris” is permitted in the syntactic ngrams. This should reduce noise in the model’s feature space, and leaves room for more interpretable ngrams. This reduces the number of features with overlapping indicativeness, which should reduce the extent to which the feature space violates the independence assumption made by the Naïve Bayes classifier (a potential advantage of dependency relation approaches as raised by [Xia and Zong \(2010\)](#)).

The ngram “shots fired” appears much higher in the syntactic ngram list. This is because the syntactic method can find this functional relationship despite tokens appearing between (which proximity ngrams will miss). The following are tweets from the unlabelled data:

- #Brussels Update: Reports shots fired, ‘arabic shouted’ before two explosions at #zaventemairport - injuries.
- DEVELOPING: The Belgian prosecutor’s office says shots have been fired during an anti-terror raid in Brussels [HTTPLINK](#)
- Belga news agency: ‘1 person dead’ ‘shots were fired’ and ‘shouts in Arabic shortly before the explosions’ #brussels [HTTPLINK](#)

Under the Stanford dependencies scheme used by `METHOD52`’s parser, there is a direct dependency between “shots” and “fired” in all three of those phrases: “shots fired”, “shots have been fired”, and “shots were fired”. So the syntactic ngram approach is able to resolve similar meanings to the same unit for the training process, and therefore allow these high-[IG](#) features to climb higher in the list if their frequency of usage justifies it.

Figure [4.14](#) shows a screenshot of proposed syntactic ngrams from a classifier that was instead built from ground up using syntactic ngrams as features (instead of the previous classifier which used the same labelled data as the proximity ngram classifier). The classifier is attempting to distinguish tweets which are either supportive or critical of Donald Trump in the Brussels data. The ngrams presented are those proposed for the supportive classification, and they demonstrate some more useful properties.

Some syntactic ngrams represent token ranges that are not possible for proximity bi/trigrams. For example, the ngram “message-support” arises in tweets like the following:

- Message from #Brussels. I Support @realDonaldTrump!
- Message from leader in Brussels, support Donald Trump [HTTPLINK](#)

Terms like “message” and “support” are heads of their independent clauses and are therefore joined when representing tweets as a single parse tree. Conjunction also permits longer range ngrams similar to these. Early in the training process, one of the proposed ngrams was “wake-vote”, which arose because of many tweet variations of “wake up [...] and vote for Trump”. Under the dependency scheme, the heads of phrases that “and” joins are directly dependent. It would be impossible for the classifier to reason over these features with the proximity ngrams, and they would obviously never be proposed to the user.

The proposed syntactic ngram “need leader” is an example of how this method can help to alleviate the *conditional indicativeness* problem. The token “leader” is used frequently in both supportive and critical tweets, so it is not sufficiently discriminative for labelling. For example:

- So @realDonaldTrump, this attack is #Brussels fault? Let’s blame the victim and divide the world? How could you ever be a leader?!
- And sadly 9 of their missionaries were killed in Brussels attack. Real leader is @realDonaldTrump wipe out ISIS [HTTPLINK](#)

There are, however, very many tweet variants like the following which contain “need” and “leader” that are supportive of Donald Trump:

- Need world leader who will stand up to face of terror,not hide behind it!Elect @realDonaldTrump and end this madness #Brussels #Trump2016
- Seeing the death and destruction that happened in #Brussels is sickening. Enough is enough, we need a tough leader. #WeNeedTrump #RETWEET
- IN WAY OVER HIS HEAD. GIVE UP CRUZ. YOUR A BOY WE NEED A MAN.YOUR A FOLLOWER WE NEED A LEADER. #OnlyTrump [HTTPLINK](#)

- #Brussels #Belgium Remember this #AZPrimary when you go to the polls. We need a STRONG LEADER more than ever now, #Trump is that man!#MAGA

Proximity bigrams would not find “need leader” in any of these. Proximity trigrams would only find “need world leader” and “need a leader”, which are therefore separate features. However, in all examples, “need” and “leader” are directly connected with a dependency relation, so the syntactic bigram is found frequently and is proposed by the system.

Finally, the list in 4.14 shows another example of terms covering logical variants so that we can be more targeted with the pseudo-counts we assign: “Vote trump” covers “vote trump”, “vote donald trump”, “vote for trump”, and “vote for donald trump” which all appear in the data and can only be counted as the same feature with the syntactic ngram approach.

News	News
HTTPLINK	HTTPLINK
police	raid
after	police
raid	after
paris	paris
paris_attacks	#brussels
#brussels	attacks
HTTPLINK_HTTPLINK	paris-attacks
attacks	brussels-raid
brussels	brussels
brussels_HTTPLINK	deal
to_paris_attacks	linked
linked	belgian
to_paris	reuters
linked_to	fired
reuters	#news
belgian	shots
#news	shots-fired
the	european
fugitive	fugitive
linked_to_paris	shootout
abdeslam	migrant
via	latest
shootout	anti-terror
in_brussels_HTTPLINK	found
terror	linked-to-paris
shots	eu
brussels_raid	during
fired	new
attacks_HTTPLINK	brussels-reuters
european	the
raid	raid-linked
raid_HTTPLINK	anti-terror-raid
anti_terror	gunman
brussels_reuters	ap
during	airport
suspect	following
salah	suspect
HTTPLINK_via	flights
brussels_police	leaders
anti	turkey
salah_abdeslam	brussels-ap
raid_linked	linked-to-attacks
terror_raid	eu-leaders
raid_linked_to	operation
gunman	arrested
alert	union
arrested	officers
new	abdeslam
shots_fired	the-latest
paris_attacks_HTTPLINK	european-union

(a) Proximity ngrams

(b) Syntactic ngrams

Figure 4.13: Annotated screenshot comparison of ngrams proposed by “News/Personal” classifiers in the Brussels analysis for the “News” category, where one classifier uses proximity ngrams (a) and the other uses syntactic ngrams (b). Yellow ngrams do not appear in (b), because URLs are never syntactically related to words, and orange do not because they have dangling prepositions. “shots-fired” appears far higher in (b), because all phrases “shots fired”, “shots were fired” and “shots have been fired” appearing in the dataset now resolve to the same syntactic bigram.

Supportive
trump
donald-trump
donald
#trump
vote
#trump2016
ago
vote-trump
months
warned
months-ago
HTTPLINK
brussels
we-need
muslims
wake-up
leader
video
message-from-brussels
i-support
america
need-trump
message
trump-warned
wrong
trump-is
is-right
trump-was
support-donald-trump
media
message-support
muslim
vote-for-trump
trump-said
b4-look
plz
#trump-plz
vote-plz
vote-#trump-plz
trump-brussels
need-leader

Figure 4.14: Example list of *syntactic* ngrams generated by the classifier which distinguishes between tweets that are critical or supportive of Donald Trump.

Notice that often the ngrams produced would also have been produced by the proximity-based ngram extraction. While it would be gratifying to generate features that are fundamentally different and superior, the safest approach is to subsume what already works, and attempt incremental improvements. Therefore, it is sensible to aim for the dependency extraction process to produce most of the ngrams which would have been extracted by the proximity-based approach anyway, but to avoid extracting those few ngrams composed of words

that are not closely syntactically related to one another. The promixity-based approach would blindly join such words.

Whether the syntactic ngrams actually improve performance over proximity-based for the same feature and document labels is likely to be highly task-dependent. Furthermore, such a test would be a poor evaluation, because different documents and features would be queried to the user in active learning under the different feature extraction schemes. Therefore, fixing the labelled document and feature lists would introduce bias toward one technique.

Therefore, it is difficult to get a straight answer in terms of raw classifier performance. It is expected that the syntactic ngrams are less necessary in simple topic relevancy classification problems; the proximity ngrams tended to be sufficient to achieve good performance in the early Brussels classifiers for example. However, when the classification problem may be best solved with some knowledge of how words apply to each other, or even just more complex class definitions, syntactic features may be more appropriate, such as for the supportive/critical problem in the Brussels project. When training classifiers for this problem, the *target* of the sentiment was important, since the supportive/critical language being directed at one target or another affects the categorisation of the entire document.

4.7 WEIGHTING PRIOR KNOWLEDGE

METHOD52 (like DUALIST) already provides the facility for the user to label an individual feature (usually unigram/bigram) with the classification that it is indicative of. The classifier then pretends to have seen an extra $\alpha = 50$ occurrences of that feature in documents with that classification.

The system then uses that information to leverage the unlabelled data to learn more about that classification. However, when labelling a feature as indicative of a particular class under the model described above, even if the feature is very indicative of that class, provided that the feature barely occurs in the data, the classifier performance will often deteriorate. Given that this problem can occur with features that barely exist in the data, what is the effect on more common features? There is a theoretical

This is a problem with what was described earlier as “feature incorporation”: a feature has been extracted, it has been discovered, and

the analyst has labelled it, so now this knowledge must be incorporated into the classifier model.

Assigning the $\alpha = 50$ pseudo-counts can be thought of as attributing a portion of the probability mass to that feature under that class. Given that at any time, probabilities must sum to 1: when introducing a new outcome or adding evidence to an existing outcome, probability mass must be reassigned. Figure 4.15 demonstrates this concept with a simple example.

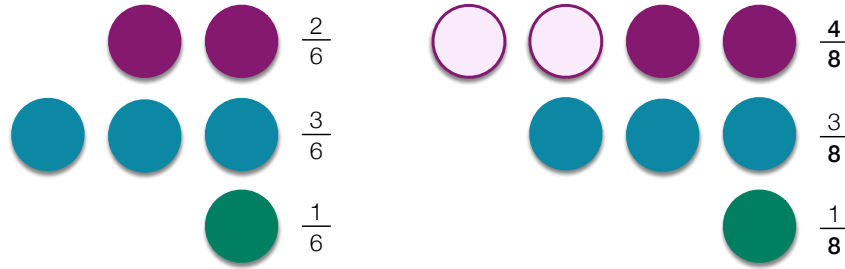


Figure 4.15: Simple example of re-allocating probability mass. Each colour is an outcome, and each circle is evidence of the outcome. On the left, with six pieces of evidence in total, and two purple outcomes, the probability of purple is $\frac{1}{3}$ ($= \frac{2}{6}$). When two counts of evidence of purple are added (the hollow circles shown on the right), the other outcomes must become less likely. E. g. green becomes $\frac{1}{8}$ instead of $\frac{1}{6}$.

When an **NBC** is classifying a new document D (a collection of features $f_1 \dots f_n \in V$), the predicted classification is the one with the highest probability given the document. For each classification label $c \in C$, the probability given the document $P(c|D)$ is proportional to the product of the prior probability of the classification $P(c)$ and the probability of the document given the classification $P(D|c)$:

$$P(c|D) \propto P(c) \cdot P(D|c)$$

Where $P(D|c)$ is assumed (the **NB** independence assumption) to be equal to:

$$\prod_{f \in D} P(f|c)$$

The following statement holds true:

$$\sum_{f \in V} P(f|c) = 1$$

This states that, for a particular classification c , the sum of feature probabilities given the classification for all features in the vocabulary is equal to 1. This must hold true for $P(f|c)$ to be a probability.

Imagine that for some user-labelled f , we add pseudo-count evidence in favour of c so that we increase $P(f|c)$. In order to maintain the sum-to-one equality above, all probabilities $P(\bar{f}|c)$ for $\bar{f} \in V$ must have just been reduced. Therefore, upon re-classifying the data, a probability $P(\bar{f}|c)$ that was once greater than $P(\bar{f}|\bar{c})$, may no longer be greater, which may be enough to change the classification of the document containing \bar{f} . Therefore, if many pseudo-counts have been assigned to a feature which barely occurs, enough probability mass could be diverted to undo what has been learnt about other features in exchange for no decision power over the feature to which pseudo-counts were added (according to the current evaluation data). The only possible benefit could be potential generalisation capability if the feature is likely to be useful in other datasets.

The problem is that we do not know what amount of pseudo-counts is appropriate for a given feature. Generalisation capability on unseen data is useful, but ideally it should be possible to add generalisability without sacrificing much performance on the current evaluation set. Sections 4.7.1 & 4.7.2 introduce methods for obtaining more informed pseudo-counts.

4.7.1 Frequency-based Weighting

One way of alleviating the pseudo-count problem is to take inspiration from how the [NBC](#) itself weights its probabilities: by frequency of occurrence. The more often something occurs, the more likely it is.

Assume that the sets of unlabelled U and labelled L documents are representative samples of our target domain. This is an assumption that [DUALIST](#) and [METHOD52](#) already make, because the entire motivation behind this type of system is that it allows the rapid learning of a bespoke classifier for a particular problem and domain. Therefore, a sensible sample of data in the target domain would have been collected and annotated in order to create the classifier.

Let $count_U(f)$ be the number of occurrences of f in U . Given a feature f labelled by the user as indicative of class c , we assume that there is a high correlation between the occurrence of f and documents labelled c ([METHOD52](#) currently assumes the correlation is equivalent to a fixed value of 50 for each labelled feature). Therefore,

if humans were to annotate all documents in U , we would expect that the *actual count* of f in documents labelled c , $count_U(f, c)$, would be close to, and at most equal to $count_U(f)$, since most, if not all, documents in U containing f would have been labelled c due to this high correlation.

We could imagine that we acquired our prior knowledge from having annotated a comparable sample to U , and so use $count_U(f)$ as a sensible proxy for how often we'd expect f to be indicative of c in general.

Given that probability mass is taken from other features under the same classification when a feature is assigned pseudo-counts after labelling, it would be easy to lose decision power over other features if most of the probability mass is given to a new feature because we devoted too large a number of pseudo-counts to it. Given that the unlabelled data can often consist of hundreds of thousands of documents, $count_U(f)$ could easily be such an over-large figure, since the labelled documents will at most number in the hundreds. It is also usually dangerous to assume that every mention of a labelled feature in that set should count as a pseudo-count, because the pseudo-counts would dwarf the counts derived from the hand-annotated data.

Instead, a maximum α can be specified. An α_{max} of 50 minimises disruption from the previous approach of assigning a fixed 50 for all features. Furthermore, in order to maintain the generalisability of feature labelling, a minimum α_{min} should be specified. Otherwise, finding no examples of the feature in the unlabelled data would result in adding zero pseudo-counts.

The simplest method of selecting values between α_{max} and α_{min} , is to take the raw $count_U(f)$. With the unlabelled dataset being often large, the maximum pseudo-counts will usually be hit easily. However, given that the aim is user exploration, and giving the user flexible tools for said exploration, an approach was taken that focuses on the *proportion* of features in the text accounted for by the labelled feature, instead of the raw frequency of the feature. In this way, the size of the unlabelled dataset is irrelevant. The proportion of features in the unlabelled documents that are equal to f , our labelled feature, is referred to by $prop_U(f)$, and is calculated as shown in Equation 4.6:

$$prop_U(f) = \frac{count_U(f)}{\sum_{\bar{f} \in V} count_U(\bar{f})} \quad (4.6)$$

In order to use this proportion to select a value between α_{max} and α_{min} , Equation 4.7 is a possibility. This would result in a scenario whereby if the feature did not occur in the unlabelled data, then α_{min} would be chosen, and if the unlabelled data was completely made up of only the labelled feature, only then would α_{max} be chosen.

$$\alpha = (\alpha_{max} - \alpha_{min}) \cdot prop_U(f) + \alpha_{min} \quad (4.7)$$

It would never be expected that a single feature would constitute the entire unlabelled dataset, or even most of it. Therefore, it is not useful to only scale to α_{max} with such a high proportion. For completeness, it may be useful to assign only the α_{min} until the feature achieves a certain proportion of occurrence deemed useful. But the particular proportions at which these two things should occur will be data specific. Very homogeneous data should scale to α_{max} later, since it is less interesting for the same features to occur more often.

The $prop_U(f)$ at and above which α_{max} is always assigned will be referred to as the *saturation* point, α_{sat} . And the $prop_U(f)$ below which α_{min} is always assigned will be referred to as the *scaling threshold*, α_{scal} . For all $prop_U(f)$ in between, the selected alpha scales linearly from the scaling threshold to the saturation point. This concept is illustrated in Figure 4.16.

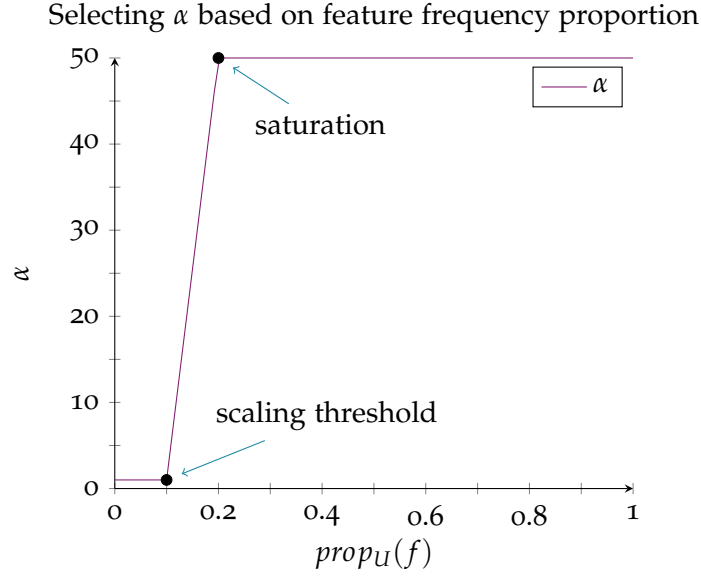


Figure 4.16: Graph demonstrating the selection of α value given a minimum 1 and maximum 50. The α is scaled based on the proportion of occurrence of a feature in the unlabelled data out of all features that occur in the unlabelled data. Proportions below the *scaling threshold* of 0.1 receive the minimum $\alpha = 1$, and proportions greater than the *saturation* parameter 0.2 receive the maximum $\alpha = 50$. All proportions between sit on a simple linear interpolation between the minimum and maximum α values.

In order to implement this idea, the equation of the scaling line is required, and the use of MAX and MIN functions to ensure the scaling threshold and saturation points are obeyed. This produces an equation for the entire purple line in Figure 4.16. Equation 4.8 is the equation for the scaling line, and Equation 4.9 wraps it in the MAX and MIN functions.

$$\alpha_{interpolation} = \overbrace{\frac{\alpha_{max} - \alpha_{min}}{\alpha_{sat} - \alpha_{scal}}}^{gradient} \cdot (prop_U(f) - \alpha_{scal}) + \alpha_{min} \quad (4.8)$$

$$\alpha = \max[\alpha_{min}, \min[\alpha_{max}, \alpha_{interpolation}]] \quad (4.9)$$

Each dataset is unique, and these parameters permit flexibility. However, they also allow the user to mostly follow the DUALIST work that found $\alpha = 50$ to be a fairly safe parameter setting, but with added ability to reduce probability mass lost to infrequent features.

Settings used for the many classifiers in the Brussels case study were $\alpha_{max} = 50$, but with an $\alpha_{min} = 1$ and $\alpha_{sat} = 0.1$ ($\alpha_{scal} = 0$). This

means that very rare features do not impose greatly on the probability mass, but that features occurring in 10% or more of the unlabelled data already return to the DUALIST default of $\alpha = 50$. With frequency weighting in place, the labelling of an infrequent feature rarely negatively impacts performance, but lends power to nudge the classifier in the right direction when without the prior knowledge the classification probabilities would be near uniform.

4.7.2 Indicativeness Weighting

When using $\text{count}_U(f)$ as a guide for the number of pseudo-counts to assign (as with the technique in Section 4.7.1), we're assuming *very high* correlation between $\text{count}_U(f)$ and the theoretical $\text{count}_U(f, c)$. But this may not reflect reality. Instead, the user's prior knowledge can be exploited to determine a better level of correlation.

In other words, the user should be able to annotate features as possessing *varying levels of indicativeness* for a given classification. A feature known (through prior knowledge) to be only slightly indicative of a classification should not receive as many pseudo-counts as a feature known to be very indicative of that classification. This section proposes a method for how this can be accomplished.

The most simplistic method is to prompt the user to supply a level of indicativeness manually. For example, we could ask the user to assign a number between 1 and 10 on how indicative of the classification the user thinks the feature is. But the user will struggle to make distinctions because a human is unlikely to be able to distinguish between a feature that is 8 or 9 indicative, and if the levels are reduced to "quite" and "very" then they are of limited use. Furthermore, the values would be based on user assumptions that may not apply to the current data.

The alternative proposed here, describes how an estimate for the degree of correlation can be obtained in a way that also familiarises the user with more data, and does not require manual indicativeness selection. Furthermore, the estimate is based on actual observations in the data.

When the user labels a new feature f as indicative of classification c , a random sample S of the documents in U that contain f is presented to the user for labelling. The fraction of the sample that is actually classified c is used to weight the pseudo-counts assigned to the feature:

$$\alpha = \frac{\text{count}_S(f, c)}{|S|} \cdot \alpha_{freq}$$

Where α_{freq} is the result of the dynamic frequency proportion alpha from Section 4.7.1. The alpha is now weighted by an actual observation of how correlated f is with c , without asking the user to make the weighting manually, but still using the user's domain knowledge.

This method takes advantage of how users are already encouraged to assess potential new feature labels since the inclusion of the original contexts feature from Section 4.5. When assessing a new feature, users can read a sample of the original contexts of the feature in order to decide whether the feature is indicative enough of the classification to warrant labelling, and they may label these documents to encode that information. This new functionality allows those less indicative features to still have a small contribution, by taking advantage of this empirically discovered indicativeness proportion. This method is, therefore, minimally disruptive to the classifier training workflow. Furthermore, given that it is evident that different features should have different levels of indicativeness, and that this method is similar to how the Naïve Bayes classifier would encode this type of knowledge, this method is likely to be useful. However, it is left for future work to have users experiment with this feature.

4.8 CHAPTER SUMMARY

This chapter encompassed the contributions toward the following research question laid out in the introduction:

How can feature discovery support the identification, definition, and characterisation of classes of text?

The chapter began by fully describing the existing classifier and active learning frameworks built into `METHOD52`, which (like `DUALIST` (Settles, 2011)) places more explicit emphasis on exploiting the individual surface features extracted for the classifier model. Then several key areas of weakness in the classifier training strategy were identified, and solutions produced for alleviating the weaknesses. Strategies were identified for further exploiting feature discovery in the wider context of the iterative process of discovering, defining, and isolating classes of documents.

Feature exploration The goal of this thesis was to exploit feature discovery strategies to give more options for exploring the data and finding classes of documents. Additionally, the original feature discovery mechanism in `METHOD52`'s active learning environment was locked into the current model of the classification problem.

`SFPD` was used in several feature discovery strategies to permit more efficient exploration of corpora and identification and evaluation of possible classification schemes:

- A strategy agnostic of classification for discovering more about a given corpus (Section 4.4.1).
- Strategies for analysing corpora defined by specific classifications, and the classifier's vocabulary (Sections 4.4.2 & 4.4.5).
- A method for alleviating the "none of the above" classification problem and filtering irrelevant data (Section 4.4.4).

Section 4.4.3 describes how to adapt these techniques to different domains of data.

Feature extraction It is difficult to interpret the meaning of isolated features that are presented in the active learning loop, and mislabelling occurs after misunderstandings of the features' role in the data. Therefore, samples of the original contexts of features are now presented to the analyst (Section 4.5).

More importantly, even with an understanding of the contexts of features, they can be insufficiently expressive or logically coherent for useful labelling. Syntactic ngrams were extracted instead to alleviate this issue (Section 4.6). The extraction of these ngrams was achieved using dependency parsing, which required adaptation to the `TWITTER` domain (Section 4.6.3).

Feature incorporation The existing method for incorporating user knowledge concerning features was very simple and fixed for every feature. This led to problems and was obviously naïve. The frequency of features and user knowledge of their indicativeness was used to incorporate the feature knowledge in a more logical way (Section 4.7).

Section 4.4 demonstrated the Surprisingly Frequent Phrase Detection (SFPD) capability to aid exploration of corpora, using the key terms that describe it, and Chapter 4 more generally presented contributions to address the following research question laid out in the introduction:

How can feature discovery support the identification, definition, and characterisation of classes of text?

This chapter now turns to final research question:

How can feature discovery support the collection of data relevant to classes of text?

Social media study often involves streaming data over time from an API using a boolean query over words and phrases in order to study an ongoing phenomenon. The phenomenon may be specific, such as “Brexit”, or general, such as “crime”. The query may be concerning a phenomenon already occurring, or in anticipation of upcoming planned events like elections, or in hopes of capturing arising events such as riots or disasters. The goal is to construct a query which best models the class of relevant data.

In METHOD52, there are components for querying the TWITTER, FACEBOOK, and REDDIT APIs. Users can place these components and configure them with the boolean query in order to begin streaming data from these APIs. Usually, analysts find that the boolean query is not sufficient to model the class of relevant data, and must build a bespoke *relevance pipeline* in order to raise the precision of the relevance decision. Additionally, the analyst may be covering several classes of relevance with a single API query.

The relevance pipeline may constitute any number of METHOD52 components which attempt to filter the data, effectively increasing the precision of the social media query. The pipeline could be as simple as a single keyword filter component which has additional matching options beyond those provided by the APIs. Or, for example, the pipeline could constitute chains of bespoke classifiers to refine the notion of relevance that the pipeline models.

For example, in a `METHOD52` project on monitoring `TWITTER` user crime concerns, analysts were interested in finding discussions about several different categories of crime, including violent knife crime. The analysts first collected data using several fairly general crime-related terms, such as “stabbing”, “street brawl”, “gunpoint”, and “murder” in order to cover all of the categories of crime with which they were concerned. The crime categories were split from the data by searching for more specific terms within the documents. To identify knife crime related content, the analysts chose the terms “stab”, “knife”, “machete”, “hack”, and “hacking”. Yet even in the resulting documents there were many irrelevant tweets. A classifier was trained to distinguish actual mentions of knife crime from discussions of books, films, and games, or jokes. Each of these steps increases the relevancy precision of the knife crime dataset being collected. However, the recall of knife crime content from `TWITTER` is upper-bounded by the original generic query terms.

The relevance pipeline cannot increase recall beyond this upper bound; this is only possible by adjusting the initial `API` query, by adding more generic crime terms in order to collect data that was not covered by the previous terms. This is important because the classes of document that can be discovered or defined in a corpus depend on which documents are actually present in the corpus. If the analyst has failed to collect sufficient relevant documents, the effectiveness of any analysis of these documents is reduced.

If the analyst fails to collect (recall) a sufficient variety of the knife crime tweets, their analysis of knife crime discussion could miss key elements of the online discussion. This is a common problem in sociological studies of social media data. If our objective is to analyse online behaviour and interaction, yet we fail to find elements of the online discussion, then our study is undermined. For example, consider a study in which the objective is to report on the attitudes to vaccination on `TWITTER`. We’re interested in what individuals and organisations are exerting influence, what information is being shared, and how people are talking about vaccines. The study is undermined if we fail to discover major influencers, or discussions concerning specific vaccines or diseases because we failed to account for these in our query terms. The same would be true if our study aimed to determine online attitudes to Brexit. The more comprehensive coverage our query has of arising topics relevant to Brexit, the better equipped we are to study online Brexit attitudes.

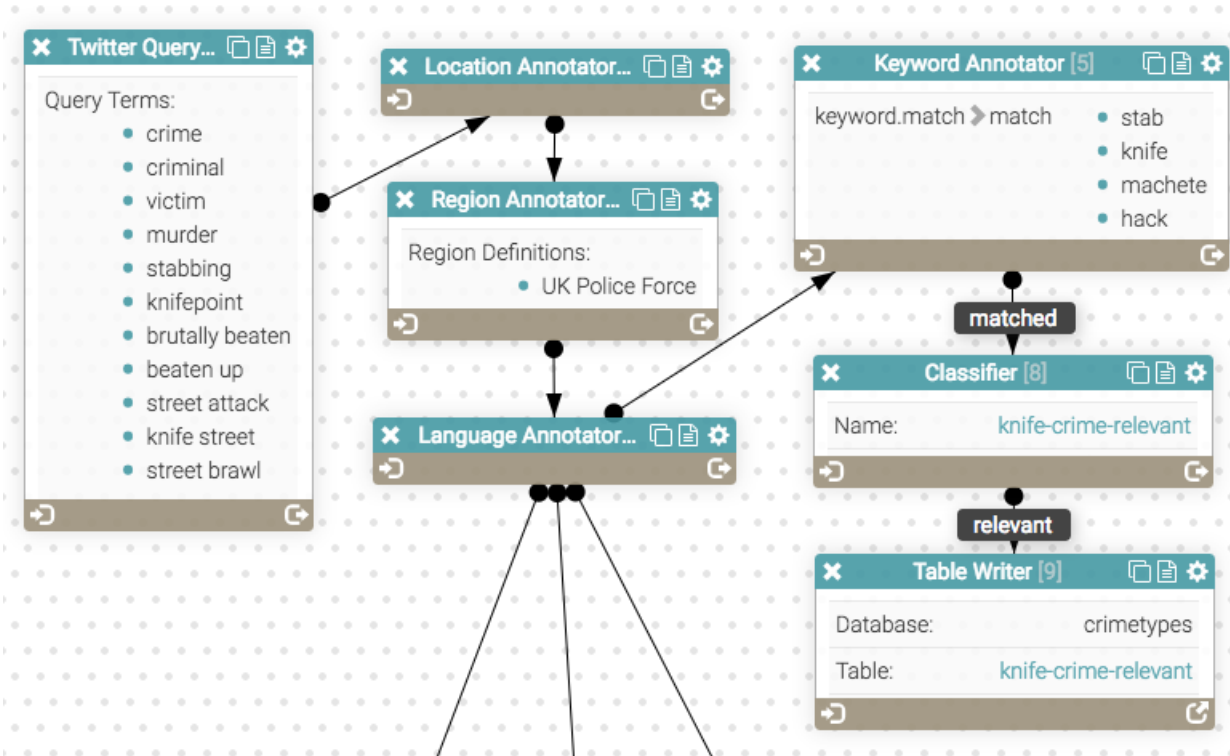


Figure 5.1: A partial METHOD52 pipeline showing a general crime query to TWITTER (on the left), followed by a series of generic location and language annotations (middle). The tweets are then passed down various relevancy pipelines. One of them is shown to the right; the keyword annotator and classifier both assess the relevance to knife crime of incoming documents. Note that none of these downstream components can increase the recall of knife crime relevant documents beyond that achieved by the original TWITTER query.

Building an appropriately high recall query is a difficult task, which can often only be done iteratively. Manual searches of TWITTER (or other platforms) or a priori assumptions are used to build the initial version of the query, and only analysis of the results may reveal other terms that would potentially increase the recall of the query.

There are a number of reasons that this is the case. Real-world events affect the discussions on social media platforms, especially on TWITTER, which has been shown to be a highly reactionary medium, where discussions are frequently sparked and shaped by real-world events (Wibberley et al., 2014). The nature of how the event and its discussion unfolds over time determines which key terms are the most effective in collecting all relevant data. The vocabulary used to discuss it can change over time, and related or sub-topics could arise that are of interest to the analyst. For example, new terms describing those individuals who support or oppose Brexit may be used in dis-

cussions about Brexit as they are invented (e.g. Brexiteer, Remoaner), or real-world events may lead to increased discussion of previously under-represented crimes in a query for crime-related discussion.

The problem of newly arising issues is most salient when topic keywords are highly generalised. For example, “general election” and “ge2017” could have been used as key terms to follow discussion of the UK general election in 2017. These terms would find all posts that specifically go to the trouble of mentioning the high-level topic, but would fail to capture documents concerning relevant arising issues in the lead up to the election such as the “dementia tax” concern, if the general key terms are not present in those documents.

In the general election example, we would expect that the generic election query terms would discover tweets mentioning both the “dementia tax” issue and one of our election key terms. However, there are plenty of tweets discussing “dementia tax” that do not use our high-level terms to contextualise them. Therefore, by including “dementia tax” as a key term, we will be able to collect these additional tweets. This is essentially a severe case of vocabulary drift. The difference is a matter of perspective: what is a new related topic? What is just a sub-topic? What is just new terminology for exactly the same topic?

Furthermore, even without dynamically arising topics and vocabulary, it is likely that there are topics and vocabulary that are present from the start of collection, but which the analyst simply did not anticipate to occur, which they will therefore miss without updating the query. It is not uncommon that the analyst’s notion of relevance evolves as they observe the data.

These reasons suggest that it is possible to acquire more relevant data than afforded by a static query consisting only of unchanging query terms derived from the analyst’s initial hypothesis of relevant terms. The additional relevant data could allow a more complete picture of the class that the analyst is interested in, or even make available classes of data that would not be present in the data acquired by the static query. All of this based on the simple surface features that define the query. When building classifiers in `METHOD52` (especially for relevance), terms often arise which could form additional query terms, but would require manually adjusting the query and re-processing the data. This practice can also be found in other work. For example, [Waseem and Hovy \(2016\)](#), as part of the data collection strategy, use frequent terms derived from the data produced by an ini-

tial TWITTER query in order to manually bootstrap their hatespeech dataset.

However, there is a limit to how useful this manual intervention can be. The data collection phase of a project can be finished before analysis begins and new potential query terms are discovered. But even if the collection is ongoing, it would be an impossible task to monitor and update the query every 15 minutes for 24 hours every day. Therefore, some automated method of adapting the query is desirable. Furthermore, it is useful to tighten the loop of data collection, analysis, and adapting the data collection using the same framework used to analyse the dataset.

The aim of the work in this chapter was to create an automated query adaptation system as a METHOD52 framework that is fundamentally exploratory, mirroring the corpus analysis and comparison approaches discussed in Section 3.1, but extending them to streamed corpora.

A key motivation for this is the frequent observation that the terms produced by applying SFPD to a social media corpus would be useful additional query terms to be passed to the social media API. This chapter therefore explores the applications of SFPD further by using it as a core component of the query adaptation framework.

The analyst does not necessarily have a specific research question, but an area of interest. The system should be able to respond to user input, since it is unlikely for a system to be able to perfectly model the researcher's internal notion of the class of interest. And with the availability of METHOD52's semi-automated, trainable systems for filtering data for relevance and maintaining precision, it is possible for the framework to focus on very high recall. Furthermore, the analyst actually invested in studying the data is more interested in data that is as complete as possible, rather than data that contains only those documents of whose relevance the machine is certain, as would be the case in a system favouring precision over recall.

Again, TWITTER is the focus, but it is anticipated that the framework strategies would be applicable to similar APIs. The overall approach will be referred to as Adaptive Twitter Streaming (ATS). ATS is the task of providing an initial query to the TWITTER Streaming API, then either automatically or semi-automatically adapting that query over time in order to collect more relevant data that would have been missed by the original query.

The adaptive streaming hypothesis is that the documents produced by a boolean query to a social media [API](#) like TWITTER's, will contain references to related topics (or sub-topics, similar topics, or related vocabulary), but that those are potentially only iceberg tips, and that augmenting the query with terms specifically identifying those related topics would lead to the collection of additional relevant documents for those topics (Section 5.4 discusses this in more detail).

In many cases, the notion of "topic" here may correspond directly to our notion of a document "class", in particular the class or classes that the analyst is attempting to explore. However, the overarching aim of [ATS](#) is to increase the pool of potentially relevant data to the classes that the analyst is attempting to define and explore. This is slightly different than attempting to automate the entire explore-search cycle, which would demand that [ATS](#) define and isolate classes on its own. Therefore, in this thesis "topic" is used as a looser term than class to avoid confusion, which could refer to the real-world event that a tweet refers to, or conversations, but also just a collection of documents unified by the vocabulary they use.

Constraints emerge from just the usage of the public TWITTER [API](#). Firstly, the [API](#) limits the number of keywords that can be used in a query, which implies that there is a relatively small cap on the number of topics that can be followed at any given time, especially since topics may only adequately be described by more than a single keyword. Therefore, only the most relevant and discussed topics are of interest. Twitter not only limits the number of query terms, but also the size of individual query terms: a query must be 60 bytes or less.

Given that for any of the TWITTER [API](#) endpoints, the upper limit of tweets that will be returned is $\approx 1\%$ of TWITTER traffic, the query cannot be permitted to become too generalised, otherwise the query will encompass too much noise, potentially causing the loss of data as the 1% becomes focused on only the frequent noisy terms. In the general election collection the term "EU" (for European Union) would frequently appear. Including this term in the query would result in incredibly noisy data, since it is used in too many other contexts.

Like Chapter 4, this chapter integrates its work into `METHOD52` in order to be used in problems tackled by `METHOD52`. Using this approach, the system's weaknesses are identified, and its design iterated on. The first prototype of [ATS](#) was applied in a study to be introduced in Section 5.2. Throughout the study the various weaknesses of [ATS](#) were exposed. After describing in more detail the core of the adapt-

ive querying technique in Section 5.3 and its assumptions about topic structure in Section 5.4, subsequent sections discuss its weaknesses, the implementation and testing of solutions to those weaknesses, and furthermore the framework’s ability to generalise to new studies and data. Section 5.5 deals with the common retweet problem. Section 5.6 discusses defining relevance and how *ATS* uses relevancy labels to manage the query. Section 5.7 explores other sources of relevancy information and expands the generality of *ATS*. Section 5.8 deals with ensuring that the most relevant but also *productive* query terms are maintained. Section 5.9 considers domain adaptation in the context of streaming data. Section 5.10 applies the refinements discussed in previous sections and tests the framework on the problem of following Brexit-related discussion on *TWITTER*.

Firstly, Section 5.1 describes existing work on automatically adapting queries and the tracking of topics.

5.1 RELATED WORK:

PSEUDO-RELEVANCE FEEDBACK & TOPIC TRACKING

The goal for the adaptive *TWITTER* querying framework in this thesis bears similarities to the fields of pseudo-relevance feedback and topic tracking, which are therefore described in this section.

Explicit relevance feedback (or simply, *relevance feedback*) is an information retrieval idea whereby user feedback on retrieval results is used to expand an initial query. This can be a repeated process, in which the user selects a subset of the results as relevant, the system adapts the query using information from this subset, and then restarts the process with the new query.

Pseudo-relevance feedback (*PRF*), also called *blind* or *implicit* relevance feedback, eliminates the manual user interaction component of relevance feedback by instead assuming that the top ranked documents are relevant. Then, as before, the system uses these documents to perform query expansion and begin the procedure again.

This technique tackles a fundamental problem of Information Retrieval (*IR*): word mismatch. This is the problem that the formulation of a query by some user often contains different terms for describing concepts compared to those used by the authors of the documents undergoing inspection (Xu and Croft, 1996).

It is easy to see the relevance of *PRF* to an adaptive *TWITTER* querying approach in which we begin with a simple query, and use it to

acquire documents, which are then inspected for additional relevant terms so that the query can be improved to return more relevant data.

Lavrenko and Croft (2001) contrast classical probabilistic models of IR with newer language modelling approaches. The classical approaches try to model word occurrences in relevant and non-relevant classes of documents, so that documents can be ranked according to the odds of their being observed in the relevant class. The latter approaches initiated by Ponte and Croft (1998) model the query generation process: documents are ranked by the probability that the query would be observed as a random sample from a given document model.

With a ranking of retrieved documents, the top documents can be used to adapt the original query and repeat the process again for PRF.

There is a distinction to be made in terms of the type of query that a given IR framework is trying to support. One type of query is what one might submit to GOOGLE: a single topic or area of interest, possibly a question or statement (where query term ordering could matter). The second type, is more like a set of queries, a declaration of the topics and subtopics that are of interest to the searcher, for which the relevant documents may have markedly different distributions of terms.

For the first type, it is appropriate to consider the query as a sample from any given document model, given its more singular origin. But given a query with terms from multiple topics of interest, we would not expect it to be generated in full from any given document. And furthermore, given that more terms could be devoted to one topic than another, any ranking of documents based on how likely the query is would be biased toward those documents that are part of the more highly represented topic.

It is the second query type that is of most interest here, since it closely matches our data acquisition strategy, where our queries are a collection of terms concerning multiple topics that are of interest to the researcher. Therefore, this section predominantly concerns itself with topic tracking, where PRF methods often contribute. While in general, the second query type could be reformulated as several distinct cases of the first type, where the user could be forced to separate their queries into distinct topics, this is not desirable since related topics will usually not have clear divisions between their relevant terms.

One of the tasks in the DARPA-sponsored Translingual Information

Detection, Extraction, and Summarization (TIDES) program was a task called Topic Detection and Tracking (TDT). The task calls for detection and tracking of topics in news stories (Allan et al., 1998). TDT approaches often include some form of PRF, but the TDT problem itself assumes some set of topics that need to be followed, which is much more aligned with the idea of an adaptive TWITTER query that attempts to track several topics and find new ones.

In the TDT task, a number of example stories for each topic to be tracked are usually provided in addition to many more off-topic stories. Yamron et al. (1999) use a unigram language model over these example topic stories as a point of comparison together with thresholding, in order to decide whether unseen stories are part of any existing topics. The authors improve on their previous similar approaches by improving their smoothing of sparse counts.

Any notion of term frequency must be incrementally updated over time, since any system tracking topics in real-time must account for the changing status of topics and their associated terms since they represent real world events and conversations (Makkonen et al., 2009; Eichmann et al., 1999).

Jin et al. (1999) produced the best-performing system for TDT-2, which itself was a collection of probabilistic subsystems. One system encoded the probability of a news story belonging to any tracked topic using a distribution over unigrams, a two-state mixture model where one state is a distribution over unigrams in the given topic, and the other is over unigrams in the entire corpus.

In the TDT pilot project (Allan et al., 1998), a vector-space model was used to represent stories, weighting terms by a variant of Term Frequency–Inverse Document Frequency (TF-IDF), a measure of the importance of a term in a corpus of documents. The measure produces a value that is proportional to the frequency of the term in the document, but which is offset by the number of occurrences of the term in the entire corpus. Once stories are represented as vectors, vector similarity measures can be used, such as *cosine*, to compare stories to those in previous topics. Others also adopted the vector-space approach (Eichmann et al., 1999).

Lo and Gauvain (2001) use the normalised log-likelihood ratio between the unigram topic model and a background general English model as a similarity measure to compare documents to topic models. Those whose similarity is above a given threshold are considered part of that topic. Adaptation is accomplished when documents that

are found to be on-topic are incorporated into the topic model. The authors also use a common method for dealing with the sparseness of the on-topic training data: the probability of a given term in a topic model is linearly interpolated with its probability in the background model.

These show a general approach to tracking and detection, which treats topics as centroids in some way. Then, new stories, which by some similarity measure are close enough to these topic centroids can be considered part of that topic. Otherwise, the story should be part of a new topic. This thesis is not concerned with matching up topics with existing topics, instead modelling topics more directly as phrases, and relying on some fixed phrases to keep to a particular domain of interest. However, as shown above, techniques for actually detecting topics are often tied to their procedure for tracking them: a candidate topic may be considered a new topic if it cannot be matched to an existing tracked topic.

Song and Croft (1999) introduce a language modelling approach that smooths probabilities of unseen terms instead of backing off to corpus statistics (as with Ponte and Croft (1998)) in order to avoid the problem of potentially assigning higher probabilities to unseen terms than to those of the observed.

This thesis avoids techniques requiring language modelling of individual documents. Zhai and Lafferty (2004) show that language modelling approaches which estimate a model for each document can be very sensitive to its smoothing parameters. This problem is no doubt worsened for the short documents of TWITTER.

Kumaran and Allan (2005) found that by treating named entities separately from other terms in a story when comparing to previously seen topic stories, performance improved. Named entity recognisers are systems which identify sequences of tokens in text that correspond to real-world entities in categories such as: PERSON, ORGANISATION, EVENT, and LOCATION. Future work will consider separate named entity treatment as an optional feature in ATS; since the notion of a “topic” is so broad in this scenario, and tweets are such a noisy data, this was not explored in the current work.

Lv and Zhai (2010) find improvements in the use of PRF by more highly weighting those terms that appear in proximity to current query terms. However, in short documents like tweets, all terms are near to the current query terms.

Most early topic tracking algorithms were designed to track news stories over time in traditional media such as newspapers. However, microblogs (like `TWITTER`, which is the primary concern for this thesis), are a very different medium. Microblogs publish content from the general public, messages which could concern any topic, including the user's personal life, which may not fit easily into a simple set of topic categories (unlike the predictable categories of newspaper content). Furthermore, microblog posts are not subject to the editing for language, grammar, and spelling that sources like newspapers are (Kalyanam et al., 2016).

Angel et al. (2009) presented a system for real-time tracking of stories on microblogs and other social media. The framework utilises `FREEBASE`¹ and `WIKIPEDIA` for detection of named entities, which form the basis of story tracking, following the hypothesis that arising events and topics will usually concern at least one named entity. The popularity of entities over time is displayed for the user. New entities (and therefore stories) are presented to the user that have undergone a surge in frequency compared to their previous rate of occurrence.

The technique in this thesis tries to make as few assumptions as possible about the kinds of phrases that an analyst would want to track. Therefore it avoids assuming named entities are the phrases that would be tracked. This avoids the need for named entity recognition, which is a difficult task on `TWITTER`'s chat language. Although, permitting theoretically any phrase does introduce a lot of potential for noise.

Efron (2010) performed query expansion on `TWITTER` with hashtags alone, since hashtags often represent user-defined topics. The framework induces a language model per hashtag, and ranks hashtags in order of decreasing negative Kullback–Leibler (KL) divergence (a measure of similarity between probability distributions) from the query model. If a tag co-occurs with many other tags in the corpus, then it is also ranked higher. Hashtag analysis alone is not sufficient to build a query to capture as much relevant data as possible.

In 2011, interest in topic tracking on microblogging platforms like `TWITTER` grew when the Text Retrieval Conference (`TREC`) included a microblog track. The same dataset was used in both 2011 & 2012, which was composed of 16M tweets sampled over two weeks. 2013 & 2014 used a collection of 243 million tweets sampled over two months. In 2015, participants collected their own data. And from 2016 interest

¹ Freebase was a large collaborative knowledge base before its discontinuation.

turned more to summarisation. The main task from 2011-2015 was a version of ad hoc retrieval; given a collection of tweets and a topic query at a certain timestamp, find all the most recent and relevant tweets that occurred before the given timestamp.

The 2012 microblog track defined a topic filtering task in which 50 topics were to be tracked through the dataset of tweets from a particular timestamp. Some topics were shown to the systems for training purposes. Each system had to judge whether to present each tweet to a hypothetical user as relevant to any of the particular topics. If the choice was made to show the user, then the system was permitted to access the tweet's relevance judgement as immediate relevance feedback, as though some user had given the information. The main evaluation measure was $T11SU$ which is biased toward precision (Soboroff et al., 2012), as are subsequent years' measures, which is at odds with the objectives of the query adaptation aspect of *ATS*, since the idea is to increase recall for collection of additional relevant data, and rely on other parts of the framework and user intervention to raise the precision (though the importance of recall remains higher).

Han et al. (2012) produced a top scoring system in the *TREC* 2012 filtering and ad hoc retrieval task. The authors note that the challenging nature of extremely short documents had led research to focus on non-text features such as *URL*, hashtag, and timestamp. The authors instead attempt to model the query and documents using query and document expansion. They apply a language modelling *PRF* model (Lavrenko and Croft, 2001) for query expansion, by taking the 20 top-ranked documents according to the initial query, and expanding the query model with the document model term probabilities. A document model is smoothed with the contents of its k nearest neighbour documents according to cosine similarity.

Zhang et al. (2012) scored high in the *TREC* 2012 filtering task with their system based on their previous year's submission, which was also iterated on in subsequent years (Li et al., 2011; Zhu et al., 2013). The system expands queries with synonyms found in WordNet (Fellbaum, 1998), a large lexical database of English, which groups synonyms together. They use *INDRI* (Strohman et al., 2005) for indexing tweets, which is an *IR* tool combining language modelling and inference network retrieval frameworks, which permit the combination of multiple sources of evidence rather than simply the words of a document. This allowed the system to score tweets with a combination of features, such as the presence of named entities, the presence of

topic terms, the presence of topic terms in the expanded terms for a document, and the presence of topic terms in the linked text of a URL. The query is also expanded using terms found in INDRI's results that commonly co-occur with query terms.

Some submissions encoded temporal information in term relevance scores. [Rodriguez Perez et al. \(2012\)](#) track the historic rate of change of the frequency of each term, and the current rate of change. When the current rate of change is greater than the historic, the term is considered *bursty*. Documents containing more bursty terms score higher in relevance. The authors found improvements over a more traditional Inverse Document Frequency (IDF) baseline, which itself was selected for outperforming other traditional methods such as BM25.

[Kim et al. \(2012\)](#), however, observed that the results of applying temporal or recency features were mixed. They instead found improvements by expanding URLs to the title and metadata of the linked page. They also found that using TF-IDF and KL divergence measures for producing candidate query expansion terms yielded poor results. METHOD52 analysts are not necessarily interested in only bursty phenomena, so temporal features are left for future work.

Many of these submissions included language and spam detection among their preprocessing steps.

[Limsopatham et al. \(2012\)](#) provide an example of a well-performing submission based on Rocchio's relevance feedback approach, a vector-space method for adapting a query based on the relevance feedback of users on documents presented to them. Terms in tweets are weighted using language modelling with a Dirichlet prior.

The bias toward precision in the evaluation of retrieval methods is noted by [Li et al. \(2013\)](#). Their submission to TREC 2013 performs best, despite a methodology which emphasises recall. In order to maintain high recall without sacrificing much precision, a classifier trained in an active learning environment is used to make boolean relevance decisions on the results returned by the current query. The authors assert that a high-precision approach works well for simple and clear queries, but not for more complex information needs, such as when the analyst forms their internal definition of relevance as they query and analyse, or when their motivation is to study the most complete picture as possible. These are scenarios which match very well analysts' use of METHOD52.

In a more traditional retrieval scenario, only those documents that are determined to be highly relevant might be presented to the user

for relevance feedback. However, in a scenario where the user is invested in high recall this does not have to be the case. In active learning, the user is presented also with *edge cases* that the current classifier is less certain about. Acquiring user feedback for these cases allows the query expansion to increase recall, whilst training the classifier to keep up precision with its user-defined notion of relevance. The framework also allows the user to short-cut the process by making their own edits to the query. The general technique is shown to generalise well to other datasets, and to adapt well to various document selection, classification, and query expansion methods (Li et al., 2014).

This is the type of scenario the thesis aims to cater for. The users are invested in building a high-recall dataset, and are interested in the actual data, in its edge cases. This is a primary reason why the adaptive querying approach taken here involves high recall terms that are constrained by classifiers built in an active learning setting. Unlike Li et al. (2013), the technique used in this thesis does not directly tie in a single relevancy classifier; using METHOD52's pipeline structure, any methodology is permitted that the user builds for assigning relevance or extracting a target set for query expansion analysis.

Sabhnani and Carterette found improvements over most TREC 2015 systems, but focus on very high precision. Instead of indexing statistics over all incoming documents, the system maintains profiles for each topic, and only updates the statistics of the profile models with documents that were deemed relevant to the profile. The system also utilises an online clustering algorithm that permits new clusters in order to detect novelty and handle redundancy in documents.

The TREC topics are quite specific and focused topics. This is not always the type of topic tracking that a researcher could want. Instead of a focused topic such as "Egyptian protesters attack museum" we instead could be interested in "Egyptian politics" and any subtopics that arise. It may even be desirable to sort documents into extremely broad categories such as "politics" or "lifestyle".

Sriram et al. (2010) categorise tweets into pre-defined classifications such as "news", "events", "opinions", and "deals" using information gleaned from the tweet author's profile and past tweets as features in a supervised Naïve Bayes Classifier (NBC). The features are hand-selected and domain-specific instead of bag-of-words, such as the presence of slang, contractions, emotional terms, and dates/times. The approach reflected the theory that those broad categories represent *author intention*, so the most pertinent information is to be found

in the user profiles of tweet authors. Any data from the tweet or its author can be the basis of the relevancy classification in the approach taken by this thesis, but the classifier component keeps to using bag-of-words features for the purpose of generality.

Garcia Esparza et al. (2010) focus instead on the text of microblog messages to classify them as relevant to a fixed set of product types, including “music”, “movies”, “books”, and “games”. The authors use the IR software library LUCENE² to perform indexing of hand-labelled training documents for each category, where documents terms are weighted by TF-IDF (proportional to their frequency in a given category, and inversely proportional to their frequency over all categories). New documents can be looked up in the index and labelled with the category of the most similar document.

Duan et al. (2012) classify tweets into seven broad categories including “politics”, “lifestyle”, “science & technology” and “entertainment”. They show classification of isolated tweets to be challenging due to their short, noisy, and ambiguous nature, but find improvements by grouping and classifying together those tweets which share hashtags or URLs. In ATS users can create any problem-specific group of tweets before feeding them into the SFPD part of the pipeline, but automatically grouping by hashtag/URL is left for future investigation.

Magdy and Elsayed (2014) follow three broad topics: “Egyptian Politics”, “Syrian conflict”, and “international sports”. They adopt the classifier approach for increasing recall but maintaining precision. But instead of requesting training data from the user, it is acquired automatically. The system maintains a high precision boolean query, and any documents from the general TWITTER stream of Arabic tweets that match it are used as training data for the relevant class. A random selection of incoming documents that do not match the boolean query and do not contain the top k highly weighted terms (according to TF-IDF) are used as training data for the irrelevant class. The term exclusion technique reduces the likelihood that important topical terms will be included in the irrelevant class. The trained classifier can then make more inclusive relevancy decisions than the boolean query, therefore raising recall. This approach requires the topics of interest to have sufficient salience in the general 1% TWITTER sample, which cannot always be guaranteed.

² <https://lucene.apache.org/>

Magdy and Elsayed (2016) successfully apply the above technique to additional broad dynamic topics over different time periods. The authors found a large increase in the number of relevant tweets retrieved when appending the top k highly weighted terms (according to TF-IDF) to the boolean query for relevance.

Coletto et al. (2016) focus on *polarising* topics, which are topics in which users have taken a side, such as political parties. Using a seed hashtag for each topic's polarisation, in a repeated process the algorithm identifies the polarisation of each user based on their usage of the seed hashtags in their tweets. Once the polarisation is identified, the seed hashtags can be expanded to include new hashtags found in the polarised tweets, and the process can begin again.

Xie et al. (2016) argue that cluster based approaches that involve nearest neighbour comparisons over document vectors where topics are collections of documents, do not scale to the data volumes output by TWITTER. Other approaches treat topics as sets of key terms (Cataldi et al., 2013; Alvanaki et al., 2012).

The strategy in this thesis approaches each batch of documents as isolated events, finding interesting terms in each batch, without attempting expensive comparisons to previous models of documents or topics. Instead it relies on the way the documents were collected to keep topically on track.

Cataldi et al. (2013) weight document terms not only by the ratio of their frequency in-topic versus frequency overall, but also by the *reputation* of the users in whose tweets they occur, where reputation is proportional to a user's number of followers and *their* reputations. The authors also focus on *emerging* topics. These are topics which are popular in the currently considered time period, but not in the previous. They use the co-occurrence statistics of emerging keywords to group keywords into coherent topics. Alvanaki et al. (2012) note that small changes in the popularity of topics are much more easy to detect in more focused topics like "Obama scandal" when compared to the more broad "Obama". They are interested in sudden changes in popularity of topics, so focus on co-occurrence of terms in relation to their overall popularity instead of single terms. The time complexity of tracking these term comparisons leads the authors to limit their analysis to only hashtags and named entities.

A different approach still is to employ topic modelling techniques like Latent Dirichlet Allocation (LDA) (Blei et al., 2003), where the notion of "topic" means a probability distribution over words, and

a document is a mixture of topics. A fixed number of topics and two hyperparameters that affect the sparsity of the document-topic and topic-word distributions must be chosen manually. [Hong and Davison \(2010\)](#) show that while topic models have enjoyed success in other domains, the short texts of mediums like TWITTER diminish the effectiveness of the models. The short documents reveal very little term co-occurrence information. This problem leads to methods which attempt to aggregate tweets into larger pseudo-documents. Tweets can be aggregated which are published around the same time, or by the same author ([Diao et al., 2012](#)). [Mehrotra et al. \(2013\)](#) find improvements by pooling tweets that share the same hashtag. [Zuo et al. \(2016\)](#) introduce a more complex topic model which models pseudo-documents as latent variables themselves. [Schubert et al. \(2014\)](#) flag the nontriviality of applying traditional clustering and topic modelling to real time large data streams like TWITTER, and note the efficiency weakness of ([Alvanaki et al., 2012](#)). The authors resort to approximation via hashing to overcome that issue, and extend their analysis beyond hashtags and named entities.

5.2 DATASET: DISRUPTING DAESH

The DISRUPTING DAESH (DD) study ([Conway et al., 2017](#)) partially funded by the UK Home Office sought to contribute to public and policy debates on the value of disruption of the jihadist social media activity on TWITTER. It served as the study during which to prototype [ATS](#).

[Conway et al. \(2017\)](#) found that while Islamic State (IS) continues to distribute propaganda on TWITTER, they are being disrupted far more aggressively and successfully than found in earlier work; their accounts have shorter lifespans before suspension, and they acquire fewer followers. The authors also discovered that other jihadists are not subject to the same high levels of disruption as IS.

One of the aims of DD was to measure TWITTER's takedown strategy for accounts created by IS supporters.

Analysis was focused on those jihadist accounts that acquired at least one follower. Over the period 1st February, 2017 to 7th April, 2017, 722 pro-IS accounts were collected along with their 57,574 tweets, 7,216 (13%) of which contained links to sites external to TWITTER. Around 65% of the pro-IS accounts were suspended within 70 days of creation.

A system capable of monitoring account suspensions needed to be able to rapidly discover pro-IS accounts by the content that they produce and monitor when they are suspended by TWITTER.

A number of seed accounts and query terms were manually identified by the authors. Then METHOD52 was used to analyse the content produced by these accounts and terms on TWITTER. Networks of TWITTER followers and friends, and the links they shared were also examined.

In order to discover new query terms, SFPD was used. However, it became clear that a more ongoing key term tracking system would be required. After tweets have been collected for a short time from the seed accounts or seed query, the newly found terms should ideally be automatically adapted back into the TWITTER query, so that collection of more relevant data can begin immediately. DD therefore served as the first testing grounds for building the ATS approach. A second project has since acquired funding and begun in order to re-apply the strategies to other jihadist groups, and further refine the ATS approach.

5.3 ADAPTIVE STREAMING WITH SURPRISINGLY FREQUENT PHRASE DETECTION

This section provides an overview of the ATS framework, and details high-level concerns that needed to be addressed for the overall approach to produce a basic working system. It describes how an SFPD component sits at the centre of the framework, analysing collected data for new phrases to be incorporated into the TWITTER query.

There is a fundamental difference between this framework and most of the topic tracking systems presented in Section 5.1. The topic tracking systems are founded on the assumption that the user has a query which defines their interest, like a question that needs answering, which may be phrased poorly or incompletely, but ultimately represents what the user wants to know. With this assumption, documents can be ranked by how well they seem to answer the question, and they can be used to expand the question with additional or more appropriate vocabulary, so that answers can be more effectively sought in the next iteration.

Instead, for this thesis, the goal was to create a system that is fundamentally exploratory. With semi-automated, trainable systems for filtering the data for relevance, it is much more beneficial to ensure a

high recall for the given topic and potentially related topics. The user does not necessarily have a strict question in mind, but an area of interest. Instead of requiring an answer to a question, it is more akin to the analyst specifying a talking point. The user's query defines a dataset, and it is the data itself that is of interest. Maintaining high recall ensures that analysts have the most complete picture possible of their area of interest, as in the example given at the start of the chapter, where analysts wanted the most complete picture of specific crime type discussions online.

For this reason, [ATS](#) shares similarity with the strategies and goals of the corpus linguistics work described in Section 3.1, in that it is the application of a phrase discovery method in order to learn more about a corpus, but where the process is applied repeatedly to a growing corpus that is acquired using the discovered phrases themselves.

The core of [ATS](#) is the Surprisingly Frequent Phrase Detection ([SFPD](#)) from Chapter 3. The phrases discovered in the [SFPD](#) process are assumed to represent topics (/conversations/vocabulary) that should be streamed, because they are important to the dataset generated by the initial query, which is the analyst's initial statement of interest. This is essentially cutting out the middle man. If instead a topic modelling algorithm (such as [LDA](#)) were employed, the system would first extract topics from the data, which may build a more coherent picture of the document topics, but it would still then need to determine the most interesting phrases in those topics in order to decide how to modify the [API](#) query, which consists of words and phrases. Interpreting topics in this way is not trivial ([Sievert and Shirley, 2014](#)), and algorithms like [LDA](#) bring another set of parameters to tweak. Therefore, it was decided to directly find new query terms without modelling intermediate topics.

The implementation of the [ATS](#) framework is agnostic to the nature of the measure for ranking features. Therefore, a better measure of surprisingness or topical terms could easily be substituted if deemed to be useful. An adaptable measure does seem to be useful, however, since the different scenarios under which the framework can be used vary greatly, in terms of the frequency of features in the target/reference corpus, and in terms of the user's interest in features being focused on high frequency features or any feature that occurs more than expected. It also permits the flexibility described earlier, primarily in Section 4.4.1.

The framework, at its most basic, is a loop containing the `TWITTER` streamer component, which manages querying `TWITTER`'s `API`, and an `SFPD` component, which receives data from the streamer component, and periodically sends instructions (including new query terms) back to the streamer component, which in response modifies its query to `TWITTER`. Figure 5.2 illustrates this process. Section 5.4 discusses the assumptions that this type of system makes about the topical structure of incoming documents.

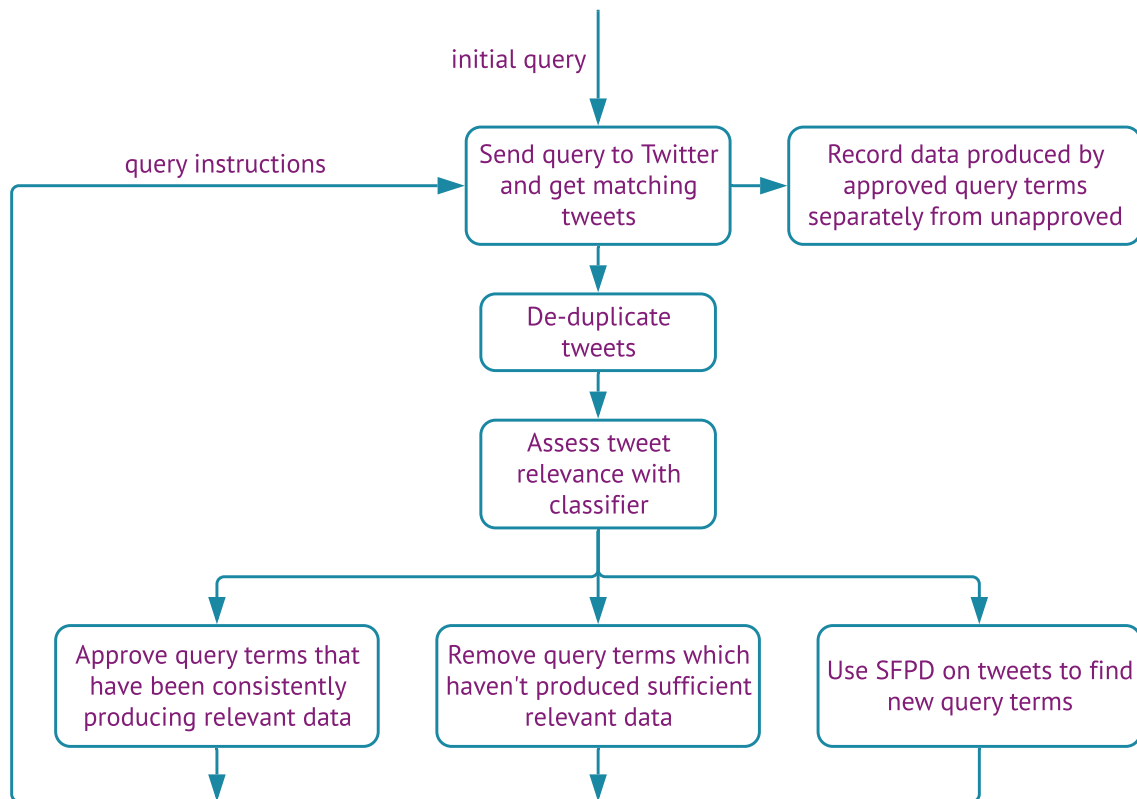


Figure 5.2: This diagram illustrates the general adaptive streaming process. An initial query to `TWITTER` is specified, the resulting tweets are filtered for near-duplicates, and annotated for relevance by a classifier. Using this collected data, we make changes to the `TWITTER` query. Query terms are approved which consistently produce relevant data, and data produced by approved query terms is stored separately from other data recorded from `TWITTER`. Query terms are removed if they do not produce sufficient relevant data. The `SFPD` method is applied to the data to find new query terms. A screenshot of a `METHOD52` job implementing this process is shown in Figure 5.3.

The `TWITTER STREAMER` component is first manually configured with one or more query terms, which serve to keep the query groun-

ded in a fairly relevant domain, helping the stream to not drift too far topically. These initial terms will be referred to as the *permanent terms*. Given that any data acquired with query terms to the streaming API will contain at least one of our query terms, the list of most surprisingly frequent phrases will almost always contain these terms to begin with, so these must be ignored. Although the TWITTER streaming component will never delete a permanent query term from its query, the SFPD component is permitted to *suggest* the removal of these terms whenever it would remove another query, because repeated removal requests may suggest to the analyst to re-evaluate the relevance of their chosen query term.

The new augmented SFPD component tracks which phrases have been proposed as query terms, so it can manage problems like TWITTER's query size limits. The component knows how to interpret METHOD52 annotations as relevance judgements, and uses this information to remove or completely blacklist terms. Relevance judgements can also be used to *qualify* terms. Data produced by permanent or qualified terms can be considered more reliable (Sections 5.6 & 5.7).

Relevant terms which despite being in the current TWITTER query are not acquiring sufficient data will be removed after some configurable expiry period in order to make space for more productive terms (Section 5.8).

Care is taken to ensure that the connection to the public TWITTER API is not being constantly interrupted and re-established. When the query is changed, the system notifies the TWITTER streaming component that it requires modification. The TWITTER STREAMER tests once per minute for whether such a request has been made, and if it has, it restarts with the new query.

In summary, ATS is built upon the keyword relevance score introduced by Sievert and Shirley (2014), using phrase expansion similar to Baroni and Bernardini's (2004), but instead of applying the measure to documents within modelled topics, it is applied directly to corpora generated by TWITTER query terms acting as the topic specification. A primarily classifier-based strategy is employed for maintaining relevancy precision as in (Li et al., 2013), anticipating broad topic structures with evolving sub-topics, and often utilising TWITTER's sample API for making distinctions from general TWITTER chatter like Magdy and Elsayed (2016), though not only as classifier training documents for the irrelevant category, but also as the reference corpus for the keyword discovery process.

When configuring [ATS](#) it helps to have an external measure of performance, if possible, to better be able to make appropriate choices for the various ways in which the [ATS](#) can be instantiated. For [DD](#), there was an obvious measure that could be drawn on. When the [ATS](#) is embedded in the [IS](#) supporter detection framework, the potential new pro-[IS](#) accounts undergo various checks to decrease the likelihood of false positives, before an analyst performs the last step of verifying the accounts as [IS](#) supporters. This gives us two figures:

1. The number of accounts raised as potential [IS](#) supporters by the framework.
2. The proportion of accounts from the list of potentials that the analyst actually verifies as pro-[IS](#).

Therefore, a method to evaluate the performance of the system, and thus the impact of changing the [ATS](#) setup, was to maximise [1](#) while holding an acceptable [2](#). In practice, for the early development of [ATS](#) it proved difficult to coordinate user labelling activity with iterations of the system given the time available. Therefore, performance evaluations were mostly carried out by direct inspection of the terms that were being added/removed/qualified, and the data that these terms produced. This does carry the benefit that performance can more easily be analysed with other tasks in mind: a phrase could be qualified that under different circumstances could be useful, but does not lead to identifying more jihadist accounts for the [DD](#) study.

It is important to be able describe the means by which a dataset was collected. This becomes a more complex task if a dataset was generated by an adaptive query. As part of the process of informing the [TWITTER](#) streaming component of new query term information, the [SFPD](#) component can write its instructions to a timestamped database table. Therefore, it is possible to reconstruct when query terms were added, qualified, or removed.

Figure [5.3](#) shows a basic instantiation of [ATS](#) in `METHOD52` for streaming over documents concerning Brexit.

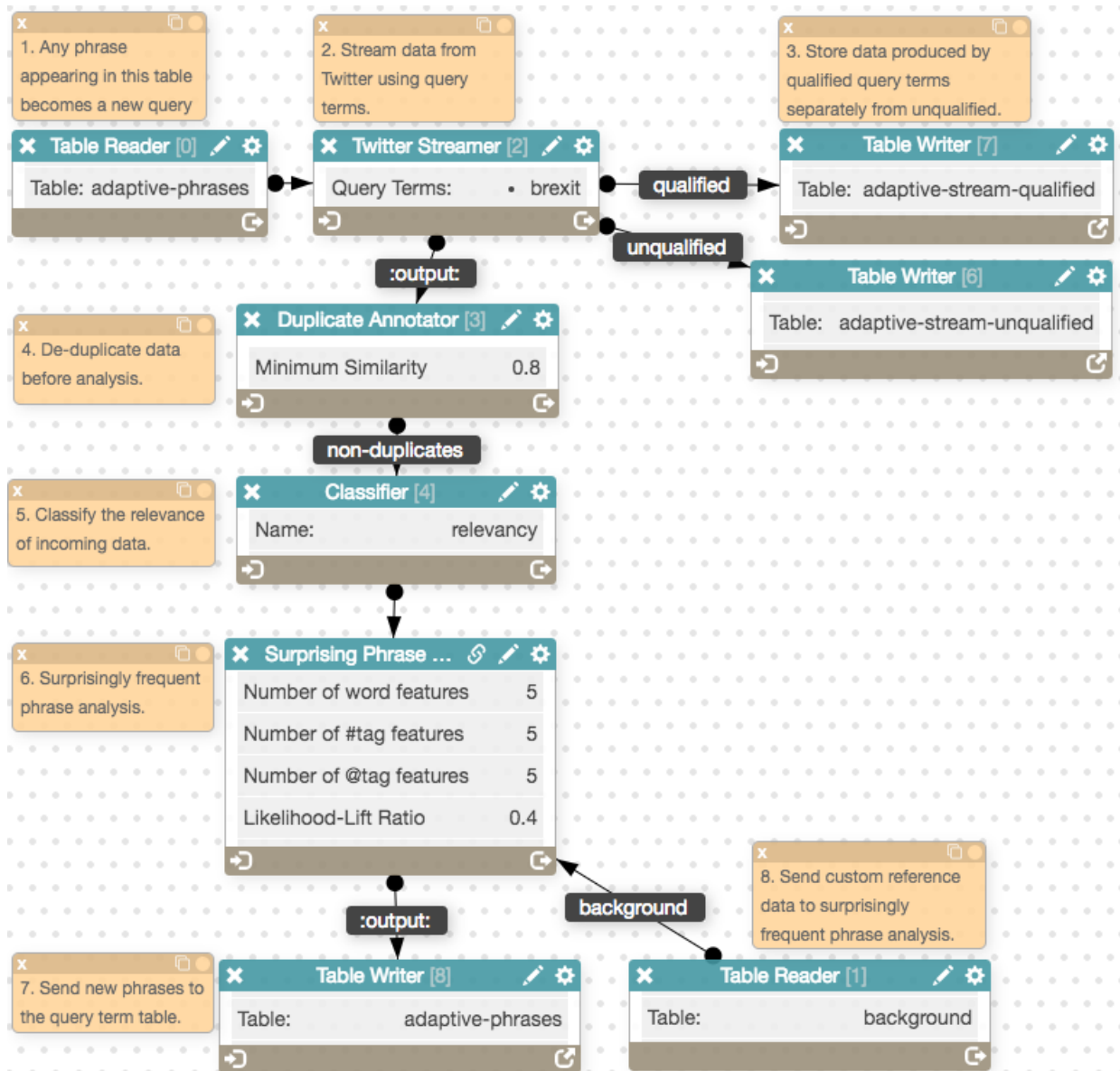
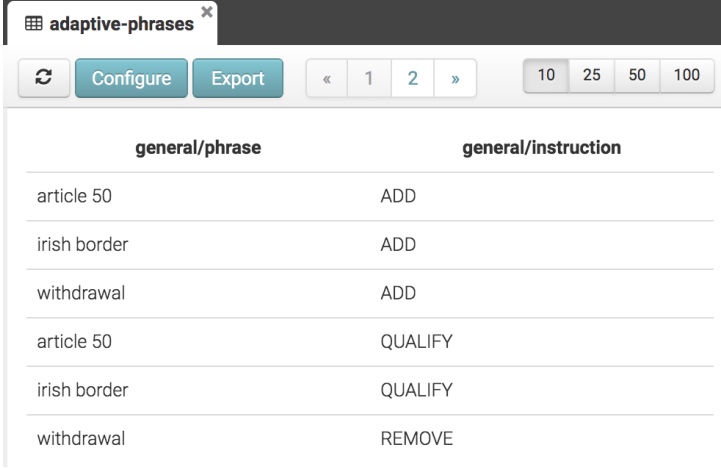


Figure 5.3: An adaptive streaming METHOD52 job. At step 1, a TABLE READER monitors a table for any new query terms. At 2, the TWITTER STREAMER collects real-time tweets matching our current query, and outputs (at 3) tweets to separate database tables depending on whether they were produced by a *qualified* query term (see Section 5.6). At 4, near-duplicate tweets are filtered out (Section 5.5). At 5, tweets are classified as relevant/irrelevant (Section 5.6). Surprisingly frequent phrase detection occurs at 6, which includes query term expiration (Section 5.8). At 7, new query terms and other instructions to the TWITTER STREAMER are written to the database table being monitored in 1. A database table is input to the SFPD as a reference corpus at 8 (Section 5.9).

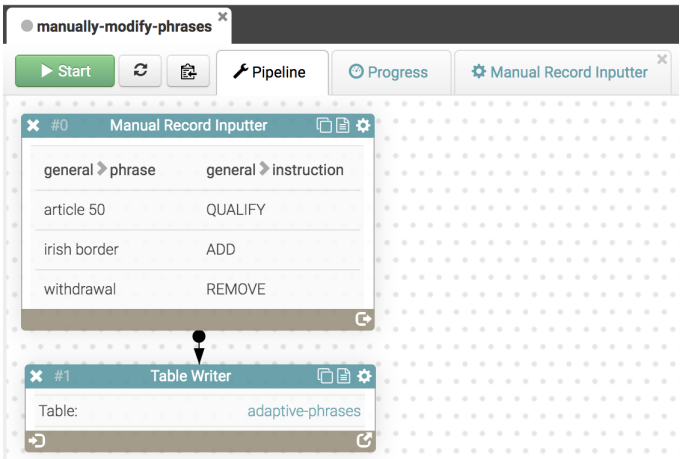
Users are also able to manually interact with the adaptive streaming process, by sending instructions to add, remove, qualify, and

blacklist query terms. Figure 5.3 shows that the `TWITTER STREAMER` component gets its instructions from the database table called “adaptive-phrases”. Instructions are automatically written to this table by the framework itself based on the data it receives. Therefore, the user can view these instructions (see Figure 5.4) and build a `METHOD52` job to write their own instructions to this table (see Figure 5.5).

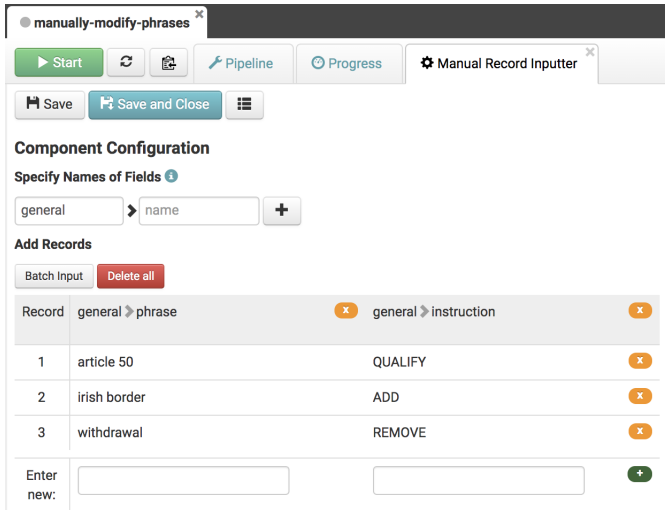


general/phrase	general/instruction
article 50	ADD
irish border	ADD
withdrawal	ADD
article 50	QUALIFY
irish border	QUALIFY
withdrawal	REMOVE

Figure 5.4: When clicking on the table name “adaptive-phrases” in the [ATS](#) job, the user is taken to the table viewing screen, allowing them to inspect the history of instructions that have been passed to the `TWITTER STREAMER`.



(a) A job which sends manually input data to a database table.



(b) By clicking the configure cog on the MANUAL RECORD INPUTTER component the user can type in new instructions to be sent to the “adaptive-phrases” table.

Figure 5.5: These screenshots show the procedure for manually inputting instructions to the [ATS](#) framework. (a) shows the job that must be created to send manual instructions to a database table. (b) shows how/where the text is input by the user.

The following sections detail the workings of [ATS](#), and address the issues that arose when building and using the framework.

5.4 TOPIC STRUCTURE

Any technique designed to stream topics from TWITTER makes assumptions about how topics arise on the platform. This section elucidates the assumptions made by the [ATS](#) framework.

Figure 5.6 illustrates two ways in which related topics might arise. Each image shows a *purple* topic being queried, and subsequently a *blue* topic arising in online discussion. The top diagram shows a scenario in which the *blue* topic begins to appear in the streamed discussion, increasingly occurring more often. However, the bottom diagram shows the *blue* topic never occurs in data produced by our *purple* query, but rises as a topic without mentioning our *purple* query terms.



Figure 5.6: *Purple* and *blue* represent documents on related topics. The query should start with collecting *purple*, then discover and incorporate *blue*. In each diagram, time proceeds from left to right. The top image shows that the proportion of *purple* discussion is slowly overtaken by that of *blue*. The bottom image shows a sudden stop in discussion of *purple* and an equally sudden beginning for *blue*, never seen in the *purple* documents.

The [ATS](#) framework is suited to the first case; it assumes that when querying terms for topic t_1 , terms for a related topic t_2 will also occur in documents mentioning t_1 (because they are related), and that these mentions are sufficient to register as surprisingly frequent.

We would expect that words typifying the topic of a text should appear surprisingly frequently. Terms specifying *related* topics are also expected to occur surprisingly frequently, since we assume that the mentions of related topics correlate with the mentions of our original topics.

The bottom image of Figure 5.6 describes a scenario that our method would not be suitable for: if there is no correlation between the mentions of our topic and a related topic, then we would not be able to discover it in documents obtained from querying for the original topic.

Having described generally how [ATS](#) assumes to discover topics/-vocabulary, this section now examines how the nature of the relation-

ship between original and related topics affects the likely correlation between the two.

Terms can establish a topical hierarchy. “Brexit” represents a topic, and a sub-topic that features within Brexit discussions is “immigration”. Brexit is the context and reason for discussion of immigration, and immigration is only part of the Brexit discussion. Therefore, Brexit is higher than immigration in this topical hierarchy. When our original topic is Brexit, immigration is likely to arise, but the reverse is less likely. Brexit is only one of the many contexts in which immigration can occur, so if the original query is immigration, there is no guarantee that Brexit out of the many contexts of immigration will occur frequently enough to be surprising. Therefore, immigration is higher in a *generality* hierarchy.

So when instead the query is the more general term, immigration, the more specific topic, Brexit, must be one of the main contexts of immigration to still occur frequently enough. This is most often probably a desirable trait, because if the analyst is interested in immigration, then they are probably interested in the main sub-topics of immigration.

While this may not be a problem for querying on immigration, it can be for Brexit. When finding that immigration is a relevant sub-topic of Brexit, it is incorporated into the TWITTER query. But if immigration is currently discussed in too many other contexts than Brexit, the data could be filled with noise from these topics. The hope in this situation is either that immigration is only proposed in the context of a longer phrase that tends to be used mostly in Brexit contexts, or that some relevancy analysis either prevents the overly general feature from remaining in the query, or filters out the noise generated from using it (see Section 5.6).

If the topics are related because they are *siblings* in a hierarchy, as “immigration” and “referendum” are both siblings under “Brexit” in a topic hierarchy, their common super-topic (here “Brexit”) must be a well-discussed context for our original topic in order for us to discover the upward hierarchical relationship from “immigration” to “Brexit”. Then in a subsequent iteration, after querying the more general “Brexit” topic, it will be easier to discover the downward hierarchical relationship to “referendum”.

When the topic relationship is not hierarchical, the likelihood of correlation is less clear. A related topic needs to be sufficiently related that it is mentioned in similar contexts to that of our original topic,

otherwise without this overlap, it cannot be discovered. However, the longer the *ATS* is running, the more chance it gets to discover another related topic that *is* mentioned in enough contexts with the previously unobtainable related topic, in order to capture it.

In order to visualise how *ATS* aims to exploit this assumed topic structure, Figure 5.7 shows how the discovery of topics in the documents streamed at each iteration of the *ATS* leads to adapting the query to include the discovered topics. And conversely it shows how the lack of discovery of documents for a topic, leads to the removal of the topic from the query.

In Figure 5.7, colours are used to represent topics, and five columns represent five spans of time, five iterations of *ATS*. For the first timespan (S1), we begin streaming for our main topic *red*, and after the timespan passes, we discover topics *yellow* and *blue* in the batch of data collected mentioning *red*. The figure illustrates this with a *red* rectangle containing *yellow* and *blue* circles. Therefore, at S2 we adapt our query to stream for all three topics *red*, *yellow*, and *blue*. For example, we could start our query with the term “Brexit” (*red*) and discover the topics “article 50” (*yellow*) and “irish border” (*blue*) within it, so in order to ensure we capture all relevant data, we adapt our query to include all three query terms for the next timespan.

After adapting the query, we discover topic *green* in the *yellow* data. So at S3 we include *green* in our query. This for example could mean discovering “petition to revoke” within the “Article 50” documents, so in the next iteration we add this query term also.

At S4, we’re still streaming on all four topics, but we no longer see any *blue* data in our *red* documents, and we barely see any *blue* returned from our query, so at S5 we are no longer streaming for *blue*. Or correspondingly, we no longer see sufficient “irish border” documents to justify retaining the term in our query, so it is removed.

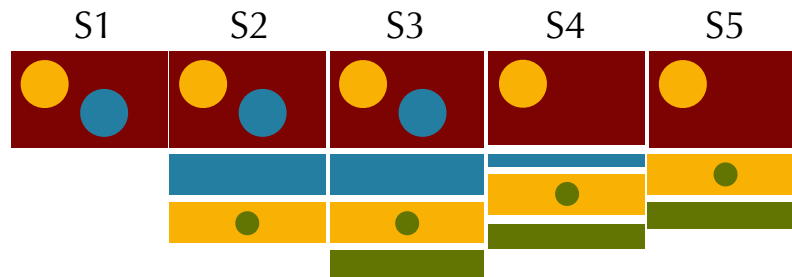


Figure 5.7: Each S is a timespan over which the surprisingness of features are calculated, thus the above shows the result of 5 iterations of [ATS](#). Each colour denotes a topic. Each rectangle is a topic we’ve streaming on during the timespan, the size of which is proportional to the data returned by the streamed topic. Each circle is a topic of interest that is mentioned within the dataset produced by its containing topic (rectangle).

5.5 NEAR-DUPLICATE TWEETS

As mentioned in Section [4.4](#), a dataset collected from `TWITTER` is likely to contain many near duplicate texts, which is due to “retweets”. Retweets are a part of the `TWITTER` platform; when a user sees a tweet that they’d like to share publicly to all of their followers, they may retweet it. This will publish a new tweet, containing the text of the original tweet, to which the user may or may not add their own comments. This section explains why it may be desirable to exclude them from the [ATS](#) analysis, and how this is accomplished.

One of the most commonly retweeted types of tweet are news headlines from media organisations. They are usually accompanied with a link to the full article. This leads to a spike in the frequency of the headline terms, which the [SFPD](#) will find surprising. The sharing of news headlines without further comment does not necessarily reveal anything about how users are discussing the topic, which is usually of interest when studying social media. Therefore, it can be desirable to detect and remove these tweets.

Due to recent changes to the `TWITTER API`, this process is now potentially easier. Originally (when this work began), there was no metadata distinction between retweets that contain additional comments, and those that are just straight-forward re-shares of existing tweets. Therefore, the metadata could not be trusted to make the necessary distinction. However, it now seems that retweets with additional comments are classed as “quotes” in the metadata. Nevertheless, there is nothing stopping a user from manually copying a tweet instead of using the `TWITTER` interface (which triggers the metadata

for retweets). Below, an approach is provided which does not rely on specific nuances of TWITTER's metadata, which is subject to change.

A possible strategy is to train a classifier to recognise news headlines. There are a number of words that occur particularly often in headline tweets, such as "live", and terms that occur in user names and descriptions (e.g. "news"). Their tweets usually contain URLs. But the way news is reported across different countries, different organisations, different topics, and times varies greatly. The classifier would need to be created for each application of ATS, and will often need retraining to keep up to date. This is a burden already present for analysing relevance of text (Section 5.6). It is also too difficult a task to maintain a current list of accounts known to be sources of news, in order to remove any retweets of their tweets.

Instead, the system can exploit the fact that re-shared news tweets are identical to the original news tweets in order to detect and remove them. However, there is nothing stopping users from manually copying text and altering it, or making small additions with emoticons or punctuation. Providing that the analyst still does not consider these minor additions to contribute information about the online discussion (which they usually do not), *near-duplicate* detection is necessary.

The solution employed is the addition of a near-duplicate detection algorithm (MINHASH Broder 1997³) into the streamer framework, where the document fingerprints are constructed from character n-grams (instead of words), since spelling is more fluid on TWITTER. User account names (@ tags) are filtered before documents are compared. Those tweets whose texts are very similar are filtered such that only one copy of them remains. The similarity comparison threshold is kept high (>80%) since tweets are very small documents, and therefore have a higher chance of being similar generally despite being distinct enough to be of interest to the analyst. This means that retweets which also carry a substantial comment from the retweeter should be allowed to pass the filter. Generally, this is desirable, since the extra comment is a contribution to the conversation.

The issue with this approach stems from those tweets which are commonly retweeted, but which are not headlines. It is problem-specific whether these tweets would be considered of interest to the researcher. They could be counted as agreement with a user discussion point, which could be interpreted as part of the discussion. Together the original and retweets would add frequency to terms on

³ A well-established method that has been used in successful search engines.

that topic, and raise them to the notice of [SFPD](#). However, this is a much less likely scenario than with headlines. News organisations tend to have many [TWITTER](#) followers, so their tweets reach wide audiences, leading to many retweets. The same would only be true of celebrity accounts and those that are similarly widely followed. If these tweets are too much of a problem for the analyst, then they must resort to training a classifier for the task. The user can optionally incorporate a classifier component or near-duplicate detection component in the [ATS](#) pipeline between the [TWITTER](#) streamer and the surprisingly frequent phrase detection to utilise either of these solutions.

5.6 MAINTAINING PRECISION THROUGH RELEVANCE ASSESSMENT

Section [5.4](#) introduced a problem that can arise when discovering a term that is greater in generality than the original term. The example used was querying [TWITTER](#) with the term “Brexit”. A sub-topic of the Brexit discourse that is likely to occur surprisingly frequently is “immigration”, but immigration as a topic is far more general than Brexit, since it occurs in many other contexts. Therefore, if the [TWITTER](#) query is adapted to find tweets containing immigration, it could produce too many irrelevant documents from the other contexts. For the [DD](#) project, similar examples were “soldiers”, “muslim”, and “allah”. This shows that in attempting to acquire higher recall, we may risk harming precision by including terms that produce too much noise.

A scenario with similar consequences is the discovery of irrelevant terms which are used unusually often across [TWITTER](#) because they represent a currently trending topic. The terms are used so frequently in such varying contexts, that they may appear in enough of the dataset as to warrant being considered surprisingly frequent terms by the algorithm. If such features were to be included, not only would the collected data become too noisy, but the streamer component would likely encounter [TWITTER API](#) rate limits, and therefore the signal from the other query terms would be drowned out. This problem is also tackled in a different manner in Section [5.9.1](#).

This section details how classifiers are used to maintain precision as [ATS](#) adapts the [TWITTER](#) query for higher recall.

A naïve solution to this class of problem is to monitor the rate at which each query term is producing documents⁴ and remove any term that produces a sudden burst of documents. While this would eliminate trending topic terms and overly general terms, it would also lead to loss of data if the relevant topics themselves became suddenly too well discussed, or the analyst was tracking bursty/trending topics. Furthermore, it is possible that a general term like immigration can be general enough to introduce too much noise, but infrequent enough or not sufficiently bursty to warrant removal.

In order to counter this class of problem, it was deemed necessary to assess the actual relevance of the data that a query term produces, and use that information to judge the inclusion of that particular query term. In other words, it is necessary to bring some precision back to the high-recall *ATS* approach.

What constitutes a relevant document is completely unique to the particular topic being studied. In *DD*, a relevant tweet is one whose text expresses pro-*IS* sentiment (or pro-violent extremism). This is very specialised, and an obvious candidate for this decision-making in *METHOD52* is a bespoke classifier.

Using documents obtained from an initial *TWITTER* search using the seed query terms, the user can construct and train an *NBC*, or even an entire relevancy pipeline of keyword-search and classifier components, to make the relevancy classification. The classifier can base its relevancy decision on any field of the tweet or author objects.

The relevancy decisions on new documents obtained using the current query terms can then be used to assess the relevance of those terms. When querying the *API* with a new term q , for every n documents obtained which contain q (including those documents used to find q as surprisingly frequent), if the fraction of those n documents classified as relevant is below threshold t , then q is considered irrelevant and removed from the query.

Both n and t are user-defined parameters. The lower the confidence in the relevancy classification, the more documents the user may wish the algorithm to consider before removing a feature (therefore, they should increase n). How far the user is willing to allow the adaptive stream to wander into potential irrelevance governs the value of the threshold t (lower t to permit further wandering). Note also that

⁴ This is not information that *TWITTER* provides; in order to find which query produced the tweet, *ATS* searches the tweet for each query term in the fields which *TWITTER* states that it uses for its search (<https://dev.twitter.com/streaming/overview/request-parameters#track>).

relevance assessment is a repeating process (every n documents containing q for each q); it is not assumed that the relevance of any term is fixed (other than the seed permanent terms, see Section 5.8). The scope of the definition of what constitutes relevance is affected most by the relevance labels assigned to the NBC's training data; if the classifier is trained to classify many types of text as relevant, then the ATS will be allowed more freedom in the notion of a relevant term.

The above constitutes the basic mechanism by which query term relevance is assessed and used to maintain precision. The remainder of this section deals with further complications and their solutions with relevance assessment in ATS.

5.6.1 *Blacklisting Irrelevant Terms*

In the DD study (and in basic tests with just the term "Brexit"), the term "EU" was frequently discovered as surprising due to its high frequency. With the introduction of a relevancy classifier, which the users had already created in order to score the relevance of tweets by a candidate IS supporter, "EU" does not pass the relevance test because the tweets it produces are too general and varied, and therefore, as hoped, is not permitted to remain in the adaptive query introducing noise into the dataset. However, for as long as an irrelevant key term remains in our query, it is occupying space that a relevant term could be occupying. A frequent irrelevant term like "EU" arises in many iterations of the surprisingly frequent phrase detection, so would be re-incorporated into the query, and have its relevance reassessed before being removed to make space for another query term. In order to avoid this problem, another threshold is established: the blacklisting threshold b (where $b \leq t$), which can also be configured by the user. If the fraction of relevant documents is not only less than t but also less than b , then the algorithm will never again propose the now blacklisted term. When $b = t$, all terms having failed the relevancy test will be blacklisted, and when $b = 0$ no term will be blacklisted. Again the value of this parameter depends on the user's confidence in the relevancy classification and also the likely irrelevant topics, and will therefore be problem-specific, hence a configurable parameter.

With the blacklisting threshold, "EU" was blacklisted early and not re-introduced ($t = 0.4$, $b = 0.3$). This not only allows more space in the query over time, but also reduces the risk of encountering TWIT-

TER collection rate limits with frequently occurring irrelevant terms like “EU”, and thus potentially losing relevant information.

5.6.2 *Qualifying Relevant Terms & Data Collection*

The goal of [ATS](#) is to collect additional useful data for analysis. We would fail in this aim if for every term yet to be discovered as irrelevant we had to contaminate our dataset with 15 minutes of irrelevant data before determining its irrelevance. Therefore, upon successfully passing a relevancy test, a term becomes *qualified*. This qualify instruction is passed to the `TWITTER STREAMER` component, which then places an annotation on the data from `TWITTER`, marking whether each document was produced by a qualified query term. This allows the analyst to ensure that only data containing qualified terms is saved to the output dataset.

Furthermore, using this annotation, the analyst should ensure that the [SFPD](#) component only attempts to discover new query phrases in qualified documents. Otherwise, finding phrases in data that has not yet been deemed relevant would increase the risk of incorporating irrelevant terms. The unqualified documents are still required by the [SFPD](#) component for the relevancy tests to be applied, so this cannot be accomplished by just filtering out unqualified documents. This problem is solved in [Section 5.7](#).

The user may also manually remove or qualify terms. This permits the kind of user interaction set out as necessary at the beginning of the chapter; the analyst should have the final say in what is relevant to their research interest. Often, problems require problem-specific intervention in order to make methods useful. In the second iteration of [DD](#) currently under way, Donald Trump is such a well-discussed topic, that it is necessary to blacklist “trump” or filter out data containing the name. Otherwise, it is very quickly discovered as surprisingly frequent. It is possible that the relevancy tests will lead to its blacklisting automatically anyway, but if analysts determine early from manual `TWITTER` querying that “trump” is contaminating their dataset, they can immediately apply this knowledge to the adaptive query if they wish.

The relevancy test approach requires some amount of streaming on the query terms in order to perform the relevancy test and qualify them. This process can contribute to a delay on initialisation and data collection, since newly proposed terms may not be immediately qual-

ified if not enough data has been collected to assess relevance. But during the process of training a relevancy classifier, and collecting data on seed terms, the user will no doubt acquire a certain amount of relevant data. A method for avoiding this delay on document collection is to supply on start-up to the [ATS](#) all of this relevant data (manually marked relevant as if by the classifier) in an effort to increase the chance that any terms found to be surprising may already pass their relevancy test on this data.

5.6.3 *Classifier Scope Problem*

In the [DD](#) study, users had already built relevancy classifiers (covering both English and Arabic separately), trained on data collected from their seed permanent query terms, which were used as part of a number of filters for assessing candidate [IS](#)-supporters based on their historic tweets. The English classifier was incorporated into [ATS](#) using the technique described above so that terms like “EU” did not pass relevancy tests, and were quickly blacklisted for their low relevancy score, as expected. However, there were some very general terms that were still being suggested as both surprisingly frequent and relevant. They followed a similar pattern: “killing”, “attacks”, and “fighting”. This section examines the cause of this, and explains the solution that was implemented.

As data is streamed from [TWITTER](#), potential useful phrases are discovered in batches and added to the query. The new query acquires additional data, which is then examined for surprisingness and relevance. The problem when using the classifier to assess the relevance of the new terms is that the classifier was only trained on the type of data produced by the seed permanent terms. But the data-stream now contains terms like “EU” before it fails relevance. Note that although [SFPD](#) can be applied on only data produced by qualified terms, relevancy tests must be applied to all data, otherwise we could never discover whether terms are only relevant in the context of our qualified terms, or whether they are relevant in their own right, and should therefore have their own place in the [API](#) query.

It turned out to be the case that “killing”, “attacks”, and “fighting” are all very indicative of the relevant classification in the classifier’s training data. But this is in the context of data produced by much higher precision manually chosen query terms, not data produced by potentially very general query terms which are expected to be

potentially irrelevant. Therefore, the classifier was relying on features that were only useful in the more narrow context of its training data.

This would seem to be a fundamental flaw in using a trained classifier to judge the relevance of data which is known to be acquired using a broader query than that which generated the training data. However, the `METHOD52` classifiers are not only trained with labelled documents, but also unlabelled. The problem was alleviated by changing the classifier's unlabelled data to data collected using the `TWITTER` sample [API](#)⁵. The classifier kept its labelled training data, including labelled features, but was re-trained using the more general sample of tweets for its unlabelled training data.

Given that the labelled features are responsible for incorporating a classifier's unlabelled data, these should be carefully examined to check whether any are too generic, which would lead to marking many unlabelled documents as relevant when the unlabelled set is switched to the `TWITTER` sample. Once the switch is complete, the user must ensure that the performance of the classifier on the test set is not degraded, or is within acceptable limits.

With this strategy in place in `DD`, terms with the killing/attacks/fighting problem were no longer being qualified by the system.

5.6.4 Classifier Drift Problem

The greatest difficulty with this overall relevance assessment approach lies in the upkeep of the relevancy classifier. As the streaming query adapts, following the drift of the topics of interest over time, the training of the classifier becomes slowly more irrelevant, because the vocabulary used to discuss the training topics will change over time, and the nature of the topics will change over time.

The training data for the classifier will need some amount of additions and alterations to continue to perform well in the long term. The severity of the problem will depend very much on how long the `ATS` is run, and how dynamic collected topics are. But the purpose of the `ATS` technique and `METHOD52` itself is not to be a tool that is turned on and left to collect some dataset, and perform some generic shallow analysis. It is to support analysts' detailed exploration and engagement with the data. So it is expected that the users would be analysing the acquired data and updating theirs and their classifiers' notions of relevance.

⁵ The [API](#) which returns a 1% sample of tweets.

5.7 SELECTIVE TARGET & RELEVANCE

Section 5.6 demonstrated how to keep the [ATS](#) qualifying only relevant phrases for any data amenable to a relevancy classifier. It suggests ensuring that the [SFPD](#) is only applied to qualified documents (documents produced by qualified query terms), but that the unqualified documents must still be passed to the [SFPD](#) component for it to perform relevancy tests. In order to accomplish this, we cannot simply filter out unqualified documents, otherwise the relevance tests could not be administered. Instead, the [SFPD](#) component was modified so that the user can manually specify whether incoming data should be used for the [SFPD](#) target data and/or the relevancy tests. In the component configuration tab, the user can assign a “use for tracking” annotation, which if possessed by any incoming document, that document will be used in relevance tests. Similarly, a “use as target” annotation can be assigned so that only data possessing this label will be used in the target set for the [SFPD](#) analysis. This permits complete control over what is target data, background/reference data, or relevancy test data.

The remainder of this section considers the use of this flexibility to exploit other sources of relevant data.

In the [DD](#) project, when monitoring the suspension rate of [IS](#) supporters, the tweets of the [IS](#) supporters themselves are the most relevant source of text. Given that the goal of [ATS](#) here was to identify query terms which would collect more tweets from other pro-[IS](#) accounts, any tweet that is actually published by such accounts is by definition relevant. Examples of these tweets should obviously be used in the training of the relevancy classifier. But it would be useful to use them in the [ATS](#) framework directly for assessing relevance of query terms, since they represent the actual content of currently tweeting accounts of interest.

The relevancy classifier strategy in Section 5.6 was used because at the beginning of the study, sufficient pro-[IS](#) accounts had not been discovered to utilise their tweets effectively. Additionally, by the end of the study, most pro-[IS](#) accounts had been suspended, their accounts no longer useful, and their tweets no longer available. However, the second iteration of the [DD](#) study (currently in progress) is interested in a wider selection of extremist groups, which are not under the same level of scrutiny and disruption as the [IS](#) accounts; more accounts are being discovered and remaining active longer. This opens

another possible strategy for assessing tweet relevance, which is generalisable to any study which involves identifying certain types of TWITTER user accounts.

Instead of relying on the classifier to extract a relevancy decision from a very broadly sampled set of tweets as in Section 5.6, it is possible to add TWITTER accounts to the API query. Using the public API, up to 5000 accounts can be followed at once, in order to receive the accounts' tweets in real-time. Accounts of interest can be added to the query, and their tweets can be used for determining the relevance of new query terms.

The "use as target"/"use for tracking" flexibility allows DD analysts to use the data collected from their query terms as the target set, but use the data collected from identified extremist accounts for the relevancy judgements. This means that, after collecting data from the query terms and accounts for some time period, the user can analyse the period for surprising phrases, and those phrases can be checked for relevance by determining whether they occur sufficiently often in the tweets of the already identified accounts.

If this approach could be used successfully, it would firstly mean that the classifier becomes redundant or just one vote in the system. Additionally, the analyst could keep a current estimate of relevance without manual intervention, since it would be based in the real-time discussions of the currently identified users of interest, instead of a trained classifier whose training loses relevance over time.

A problem, however, is that not all tweets from a pro-extremism account are actually indicative of pro-extremism language. And furthermore, the output volume of the currently identified accounts may not always be sufficient to judge relevance, either simply in terms of the number of documents requested by the user to use for a relevancy test, or due to the language across currently flagged and candidate accounts being too heterogeneous for sensible relevance comparison. Relying on a small number of individuals to provide the relevancy test data is a much taller order than all tweets containing a given term.

The most fundamental problem with this approach, however, is the lack of analysis of the generality of newly proposed terms. It could be the case that a potential query term occurs in many of the tweets of pro-extremist accounts, but the term may also occur in many other contexts. If such a term (perhaps "Allah") were to be added to the TWITTER query, then too much noise would be introduced into the

collection. This issue suggests that it would not be possible to remove the need for the relevancy classifier strategy completely.

It may be the case that this strategy is best employed as an additional source of term discounting. In other words, we could maintain use of the relevancy classifier approach, but if a query term would be qualified but does not occur sufficiently often in the tweets generated by pro-extremism accounts, then remove it instead.

For the current iteration of [DD](#), it remains to be seen whether this strategy is ultimately beneficial, since it is still under consideration.

We could also consider the pro-extremist tweets as our target data. And in fact, an offline [SFPD](#) analysis on a selection of these tweets did reveal some useful query terms which then were added to the analysts' permanent query terms. The analysis was especially good at finding place names that were important in discussions by the pro-extremist accounts. But there was also considerable noise, perhaps suggesting that there are indeed many tweets not containing extremist language.

Bot accounts on [TWITTER](#) are becoming increasingly common. They are likely to influence the [SFPD](#) results and relevance calculations. It is likely to be study-dependent whether the analyst prefers to exclude bot accounts from their analysis. [METHOD52](#) provides the user with bot detection components, but they have yet to be used in the adaptive streaming framework.

It is always possible to manually influence the relevance calculations by manually delivering documents (via [METHOD52](#)'s manual record input component) containing specific terms to the [SFPD](#) component, annotating them with relevant/irrelevant, and with the "use for tracking" label. This once again ensures that the analyst can finely tune the framework if needed.

5.8 QUERY TERM EXPIRATION

Given the limit on query terms for the [TWITTER](#) key term tracking [API](#), there must be some method of untracking query terms to make space for new ones. This section describes the full [ATS](#) approach for untracking terms.

As previously described, query terms will be removed if they fail relevancy tests, but if sufficient data is not collected to complete a relevancy test, i.e. when a query term has not produced enough data from [TWITTER](#), the query term would be occupying space that a more

productive query term could be using. For this reason, a temporal expiry was introduced. The expiry time is configurable by the user. With an expiry set to 30 minutes, any query term that was last proposed as part of the TWITTER query more than 30 minutes ago will be removed from the query.

It is possible the user has no idea how much time to devote to a given query term. The simplest recourse is to match the query expiration time to the duration of the period during which the SFPD collects data before processing it. This ensures that if a term is found to be still surprising at the end of the period, it will be re-added upon its expiry, thus allowing a term's own surprisingness to govern its expiry.

This does imply that the rate of expiry of terms depends on the number of phrases that are requested from the top n . If the top 50 most surprising phrases are proposed each iteration, terms are more likely to be re-queried and protected from expiry than if only the top 5 phrases were being proposed.

It is possible to set up METHOD52 components to count the occurrences of query terms in timed batches of documents, and periodically send an instruction to remove query terms that have not produced enough data. This would permit the use of an expiry based on the raw amount of data that a query is producing, instead of whether it continues to be surprisingly frequent. If users commonly use this type of structure, then this method of expiry will be incorporated into the SFPD component for better usability.

The form of expiry based on surprisingness is currently favoured over the latter document-quantity-based expiry due to the following reasoning.

The fundamental hypothesis behind ATS using SFPD is that if a phrase occurs surprisingly frequently in the data of interest, then it likely correlates with the topic(s) of interest, and therefore may represent a related topic. If it is no longer occurring surprisingly frequently even with the advantage of being present in the API query, this would imply that it no longer correlates with the topic(s) of interest according to the hypothesis, either because its rate of occurrence is only now due to uncorrelated topics, or because it is not being used much any more. In any case, it is no longer occurring more than one would expect in the reference corpus.

It could be potentially less harmful to miss some very infrequent but relevant data, than it is to incorporate medium frequent but no

longer surprisingly frequent terms, and hoping that the relevancy tests exclude them.

Ideally, if the term becomes relevant again, it should then begin to occur surprisingly frequently, and be flagged by the [SFPD](#) process. It may be beneficial to allow such recurring features to be immediately qualified if they were qualified in the fairly recent past, rather than needing to collect enough data to be tested again. But more studies are required to test this.

If all queries were subject to both either of expiry, the system would likely fail to remain on topic as soon as there was a lull in discussion of a given term. The original query terms would expire in favour of either no surprising phrases, or noise that made it through the relevancy filters without the presence of more surprising features to outrank them. This is another reason to maintain the permanent query terms which the user manually configures before starting an [ATS](#) instance. The permanent queries are not permitted to expire or fail relevancy tests (though, as noted previously, it is logged when they *would* have failed).

Deciding which query terms should be permanent is an important task. The more permanent queries, the more tightly defined the scope of the stream (and therefore topic) is, since these terms will ensure there will always be documents containing them if they exist and are not drowned out by other popular terms and rate limits. However, with a single feature defined, the [ATS](#) is free to add many terms and explore related topics, with only a loose core definition of the stream. In the project following from [DD](#), the analysts with [SFPD](#) help are currently using over 100 permanent query terms for English alone. The [ATS](#) is currently only infrequently finding additional terms that have enough relevant examples to become qualified.

No system is perfect. The adaptive stream will at some point incorporate an undesirable term. The simple remedy for this is to allow users to interfere. As mentioned previously, the framework allows the user to send query term removal instructions to the [ATS](#) system.

5.9 DOMAIN ADAPTATION

Section [3.7](#) describes how [SFPD](#) approaches domain adaptation. This section explains why domain adaptation was necessary for the [DISRUPTING DAESH \(DD\)](#) project, and how it can be leveraged in various manners in order to lend generality to the [ATS](#) framework for other

types of project. Strategies discussing applying [SFPD](#) to corpora in Section [4.4](#) also complement [ATS](#) given that fundamentally it is still just applying [SFPD](#) to corpora, albeit corpora growing in real-time obtained through use of an adaptive query.

In [DD](#), a vast number of the tweets requiring analysis were written in Arabic, so with a reference corpus in English, every Arabic word would have seemed incredibly surprising. Instead, a collection of Arabic tweets were gathered as a reference corpus by using the [TWITTER 1% sample API](#), and filtering out all tweets that were not marked as being Arabic.

The default English [WIKIPEDIA](#) reference corpus is large and broad in its coverage of English, so it often can be sufficient even for finding important terms in social media text. However, if the dataset is particularly filled with conversational language, especially chat language and profanity, terms in this category will begin to seem more surprising to the algorithm, since [WIKIPEDIA](#) itself lacks pervasive use of them. This effect is usually undesirable, since these elements of language often do not carry much information about the content of the tweets.

Given that an Arabic [TWITTER](#) dataset was used for the Arabic reference corpus, using an English [TWITTER](#) corpus was appropriate for consistency of methodology.

A phenomenon to be mindful of when using a [TWITTER](#) dataset for the reference corpus is the ubiquity of retweets. Many near-duplicates of the same text will greatly down-weight the surprisingness of the words contained within text in the target set. This problem was avoided in [DD](#) by applying the same techniques used in Section [5.5](#) to filter retweets.

There are several ways that we could construct the reference corpus in order to better exploit the surprisingly frequent phrase analysis, depending on the particular problem scenario. The following Sections [5.9.1](#), [5.9.2](#), & [5.9.3](#) describe potential strategies to be explored and their motivations.

A general point that applies to many of the approaches in this section is: domain adaptation does not have to be a one-time operation. In fact, in an adaptive streaming setting, the notion of the target domain is quite fluid, given that the query generating our data is changing over time. Several of the techniques described below suggest a continuously adapting reference corpus. The issue with this approach, is that if the reference is constantly growing, and keeping

all old information, any fairly small new addition could be drowned out by the existing growing reference corpus, such that instead of observing that the word “badger” occurs in a reference corpus of sensible size with a probability p , we find some probability q where q is far less than p , thus not having the effect of down-weighting the surprisingness of “badger” in the target set. A methodology for dealing with this issue, is to build the reference corpus in chunks of temporal intervals, and remove chunks after some expiry period. But this is not examined here, since evidence of the above problem has not yet been confirmed.

5.9.1 *Continuous Sample Integration*

Section 5.6 demonstrated the usefulness of the TWITTER sample endpoint in providing a general sample of TWITTER language for use as unlabelled data in the relevancy classifier. And the DD study used a dataset generated from the sample API as a reference corpus for the SFPD component in ATS in order to provide a reference which knows about TWITTER language and recent trending topics. However, trending topics and language aren’t static phenomena, instead changing over time. In order to capture this and avoid the reference corpus becoming obsolete, the SFPD component was modified to be able to continuously integrate new reference documents. This permits the methodology of continuously integrating tweets obtained from the TWITTER 1% sample endpoint, so that during each inspection of the target data, the background expectation includes an up-to-date sample of all tweets up until the moment of inspection.

Note that this still requires an initial reference corpus, whether that is simply WIKIPEDIA or a previously collected TWITTER sample. Otherwise, the reference corpus would be too small, and the surprisingness of all terms would be greatly overestimated.

This technique has the effect of down-weighting the surprisingness of terms that are being used in TWITTER-wide trending topics, and retaining an up-to-date knowledge of the latest chat language for the reference corpus. This ensures that irrelevant but incredibly frequent terms arising from trending topics, which are likely to contaminate our tweets, are less likely to be picked up as surprising. The problem of newly arising chat trends is more important in long-running adaptive streams, when trends have had a chance to pervade TWITTER discourse.

If the analyst’s topic of interest is a major trending topic on Twitter, we risk losing sight of it, since the terms associated with our topic will no longer be so surprising compared to the background noise of TWITTER. Therefore, this technique is most useful when dealing with non-major trending topics, which may produce very conversational data.

In the METHOD52 [ATS](#) framework, this functionality is accomplished simply by piping the output of a TWITTER SAMPLE component with background data annotations to the [ATS’s SFPD](#) component.

5.9.2 *Noise Reduction with a priori Topic Knowledge*

It is sometimes the case that query terms defining a topic of interest have known ambiguities. A term could be used by another irrelevant topic, e. g. when tracking tweets containing “badgers” for the animal, tweets concerning the Wisconsin Badgers athletic team will no doubt make an appearance.

It is possible to filter such data in various ways. If we have a notion of relevant/irrelevant users, we can filter tweets by their author or their author’s follower network. We could also train classifiers to recognise tweets with more relevant language, and use them to filter out the irrelevant.

An alternative is to select a reference corpus which reduces the surprisingness of terms likely to arise from the irrelevant topic. Following the “badgers” example, we could collect a dataset of tweets mentioning the Wisconsin Badgers, and other terms more specific to the athletic teams. This would down-weight the surprisingness of terms that often occur in that type of tweet.

This will of course lower the surprisingness of the original ambiguous term “badgers”, but in this case it would be one of our permanent query terms, so this does not matter.

It will still be necessary to have an amount of other more general reference data to avoid the [ATS](#) focusing solely on the “badgers” features that are explicitly not related to the Wisconsin Badgers.

It would be interesting to follow the example of Section [5.9.1](#), and continuously integrate the irrelevant data stream into the reference corpus. However, TWITTER prohibits the simultaneous use of the same [API](#) endpoint, which here would be simultaneous connections to the keyword tracking [API](#) (whereas using track and 1% sample concurrently is permitted).

5.9.3 *Topic Specificity*

Section 5.9.2 focused on using the reference corpus to *exclude* terms from irrelevant topics, which may have otherwise confused *ATS* due to topic term ambiguity. This is accomplished by defining a reference corpus that typifies the irrelevant topic.

The *topic specificity* approach, however, seeks to better define the domain in which our target documents occur, so that the only language considered surprising is that which makes our target data special compared to other language in the same domain.

Using a *TWITTER* reference corpus instead of *WIKIPEDIA* to decrease the surprisingness of chat language is an example of this type of domain adaptation. But we needn't stop there, for the "badger" example, we may choose to not only collect a *TWITTER* reference, but also tweets about badgers in general to further increase our specificity for the domain of interest. For this example, we would perhaps hope to catch only newly arising badger discussions. This will raise the bar for a term in the badger target documents being surprising, so that it might require news of a badger cull to spark argument on *TWITTER* and lead to a surprisingly frequent use of badger culling terms.

5.10 EVALUATION ON BREXIT

In this section, the adaptive streaming framework is tested on a new domain. A simple initial query of "brexit" is used to collect tweets about Brexit in an adaptive streaming job. The job is then monitored to determine which topics it decides to follow. Section 5.10.1 describes how the *METHOD52* job is constructed, and Section 5.10.2 discusses the output of the framework.

5.10.1 *Setup*

Figure 5.8 shows the complete *METHOD52* job for adaptive streaming over the initial query "brexit".

The term "brexit" is a good query term because it is very precise, or selective: anything mentioning the word "brexit" is likely to be relevant, because it has no alternative meaning. The closest to irrelevant its use is likely to get is when used as a metaphor, but even this could be of use to someone studying the impact of Brexit on social media discourse. For this reason, the *CUSTOM LOGIC ANNOTATOR* component

sends all tweets to be marked as relevant simply if they contain the word “brexit”. However, a trained classifier is used to determine the relevance of all other tweets. At first, all tweets will contain “brexit” since it is our initial query, but as the stream is adapted to include more query terms, this will not always be the case.

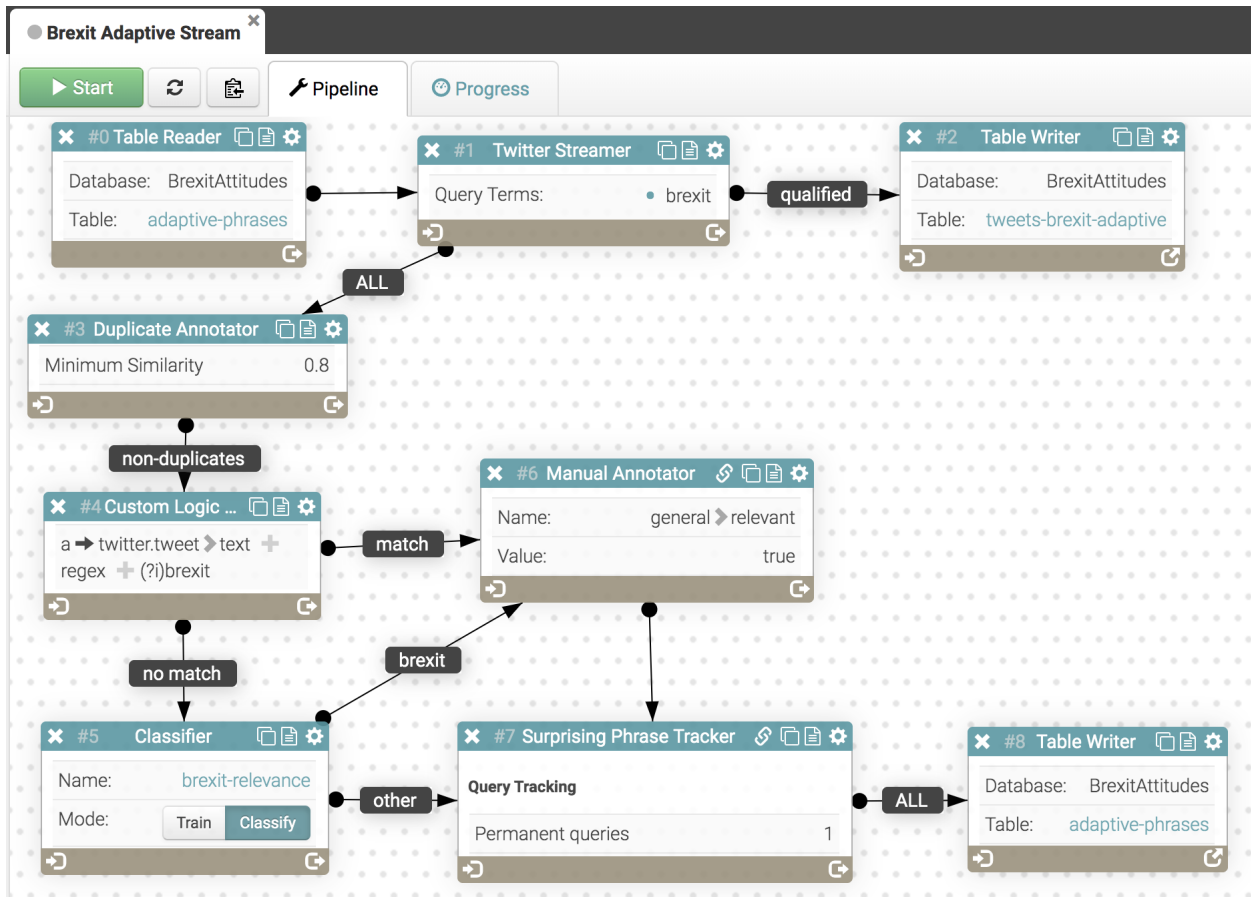


Figure 5.8: The `METHOD52` pipeline created for the Brexit adaptive streaming job. Tweets coming from the `TWITTER STREAMER` are first assessed for relevance, then passed to the `SURPRISING PHRASE TRACKER`, which manages the suggestion of new phrases, and removal of existing phrases from the current `TWITTER` query by sending instructions to the “adaptive-phrases” database table, which the `TWITTER STREAMER` monitors for changes.

The classifier was trained first on tweets collected using the query term “brexit”, where all tweets were relevant. But the classifier also needs examples of irrelevant text. Therefore, the `TWITTER` sample [API](#) was used to collect a random sample of 100,000 tweets over a two week period. More evaluation and training data was then taken from this data. This not only helps the problem of “brexit” tweets always being relevant, but is also the process of addressing the “classifier scope problem” as detailed in Section 5.6.3. Three hours were given to

train the classifier, which resulted in test data consisting of 92 relevant, and 78 irrelevant tweets, and training data consisting of 55 relevant and 107 irrelevant. This produced a relevant f-score of 94.4% and irrelevant 93.8%.

The SURPRISING PHRASE TRACKER was set up to propose up to 30 new phrases when either of the following two criteria were met: a: 10,000 tweets had been collected, or b: 20 minutes had passed. This should give the framework sufficient text to produce useful key phrases. Terms that had not been re-suggested after 25 minutes were expired. Terms had to achieve a relevance rate of at least 30% every 100 tweets that contained them in order to remain in the query. The likelihood-lift ratio for the surprising phrase analysis was set to 0.4. These were parameters deemed sensible after experimentation for the DD study.

The DUPLICATE ANNOTATOR for filtering out near-duplicate tweets is an optional step, but it helps the system to not be side-tracked by bot spam, or individual massively retweeted tweets. However, this can make it more difficult to identify commonly retweeted news headlines.

The SURPRISING PHRASE TRACKER communicates instructions to the TWITTER STREAMER using the database table called “adaptive-phrases”. This table then is also a log of what the framework has been doing with the query. The user can click the table name to see this log (Figure 5.9 shows what screen is presented to the user when they inspect the table).



The screenshot shows a web application window titled "BrexAttitudes:adaptive-phrases". It has a "Configure" button and an "Export" button. On the right, there are filters for 10, 25, 50, and 100 items. Below the header, a table lists phrases with their timestamps and instructions.

chronology/timestamp	general/features	general/instruction
27-05-2019 15:06:40	brexit party	ADD
27-05-2019 15:06:40	nigel Farage	ADD
27-05-2019 15:06:40	lib dems	ADD
27-05-2019 15:06:40	remainers	ADD
27-05-2019 15:06:40	farage's brexit party	ADD
27-05-2019 15:06:40	revoke article 50 petition	ADD
27-05-2019 15:06:40	for the libdems	ADD
27-05-2019 15:06:40	remainer	ADD
27-05-2019 15:06:40	leavers	ADD
27-05-2019 15:06:40	brexiteers	ADD
27-05-2019 15:06:40	may's deal	ADD
27-05-2019 15:06:40	ann widdecombe	ADD
27-05-2019 15:06:40	bollocks to brexit	ADD
27-05-2019 15:06:40	sajid javid	ADD
27-05-2019 15:06:40	corbyn's	ADD
27-05-2019 15:06:40	remoaners	ADD
27-05-2019 15:06:40	corbyn	ADD
27-05-2019 15:06:40	bxp	ADD
27-05-2019 15:06:40	deluded	ADD
27-05-2019 15:06:40	heartlands	ADD
27-05-2019 15:06:40	libdem	ADD
27-05-2019 15:06:40	the Tories	ADD
27-05-2019 15:06:40	a brexiteer	ADD
27-05-2019 15:06:40	brexiters	ADD

Figure 5.9: Screenshot of the view given to the user when they inspect the “adaptive-phrases” table, which is used by the SURPRISING PHRASE TRACKER to communicate instructions to the TWITTER STREAMER regarding the API query. This is at the beginning when the first batch of phrases is suggested for the query.

5.10.2 Results

The adaptive streaming job was started at 14:46 on 27th May, 2017, shortly after the EU elections were held, the UK still participating due to the postponement of Brexit. The framework was monitored for two hours; this is the period with the most activity as the system settles on a core set of query terms. Then for the next two days, terms that were qualified for longer than 5 minutes were examined. Finally, a sample of the output over the two days was hand-annotated to quantitatively estimate the performance of the framework, and the

relevancy classifier was used to estimate the relevance rate of all the output.

Table 5.1 shows the phrases which were proposed as query terms after analysing the first batch of data collected over a 20 minute period. More phrases are proposed each 20 minutes, but the framework quickly settles on a core set of relevant terms, and in addition, a few that are bordering relevant; after two hours, “globalist” terms and “revoke article 50 petition” are added and removed a number of times as potentials, though no longer producing enough relevant documents to reach qualified status.

Phrase	Notes
bexit party	
nigel Farage	Current leader of the Brexit party
lib dems	
remainers	Supporters of remaining in the EU
Farage's Brexit party	
revoke article 50 petition	
for the libdems	The Liberal Democrats performed very well in the EU elections, potentially due to their stance on a second referendum.
remainer	Supporters of remaining in the EU
leavers	Supporters of Brexit
Brexiters	Brexit supporters
May's deal	Theresa May's Brexit deal
Ann Widdecombe	Ann Widdecombe became MEP
bollocks to Brexit	
Sajid Javid	Announced his intention to stand for Conservative leadership, saying, “first and foremost, we must deliver Brexit”
Corbyn's	
remoaners	Derogatory word for those in favour of remaining in the EU
Corbyn	
bxp	Brexit party
deluded	Popular at the time of streaming. While fairly generic, the majority of tweets matching this word seem to be about Brexit, especially discussion about critics of Brexit being deluded for considering the EU election result a victory for Remain
heartlands	The part of the country where there is traditional support for a political party. Arises in discussion as parties like the Brexit Party take vote shares that were traditionally safe for other parties.
libdem	
the Tories	
a Brexiteer	Brexit supporter
Brexiters	Brexit supporters
the globalists	Reference to the contrast between “nationalists” and “globalists” in the EU elections.
Anna Soubry	Member of Change UK attempts to explain the party's performance in EU elections.
Macron	Emmanuel Macron, president of France arising in comments about EU elections.
ffs	“For f**k sake”
MP's	
the backstop	An arrangement which would apply to the Irish border after Brexit

Table 5.1: The phrases that were proposed after the first 20 minute batch of streaming Brexit tweets on the 27th of May (15:06), with explanatory notes.

Most of the phrases proposed seem to be highly related to the Brexit discussion. A clearly too general term that [SFPD](#) did propose is the acronym “ffs”. It is added to the query, but fortunately never reaches qualified status, since its lack of relevance outside of the Brexit keyword leads to it being blacklisted and removed from the query 5 minutes later. Table 5.3 shows when queries were removed (“ffs” is the first). Note that proposed phrases containing “brexit” do not contribute to the query usefully, since the original “brexit” keyword will already catch this data. However, common phrases containing “brexit” could provide useful context for the analyst. In future versions, the framework could distinguish between phrases provided as information for the user and separate phrases which are actually included in the query, so that phrases already covered by the original query need not occupy space in the query.

Phrase	Qualified	Phrase	Qualified
brexit party	15:08:17	remoaners	15:38:51
corbyn	15:09:29	the lib dems	15:39:24
the tories	15:10:16	libdem	15:40:02
remainers	15:12:04	globalist	15:41:46
macron	15:14:04	bxp	15:42:14
nigel Farage	15:15:35	mp's	15:47:22
lib dems	15:16:16	Farage	15:49:06
deluded	15:20:07	a Remainer	15:51:39
Sajid Javid	15:22:47	a Brexiteer	15:52:26
leavers	15:23:05	Brexiteers	15:52:32
Remainer	15:25:13	Ann Widdecombe	15:58:16
brexit	15:27:12	globalists	16:01:50
for the libdems	15:28:04	bollocks	16:06:43
Brexiteers	15:33:05	May's deal	16:21:42
Corbyn's	15:33:22	the backstop	16:24:13
the libdems	15:36:08	revoke article 50	16:35:56

Table 5.2: The above are all phrases that were qualified in the two hours after starting the adaptive streaming. Qualified terms are those that have passed the relevance test. Tweets produced by qualified query terms form the output dataset.

It is interesting to note phrases which are qualified for a period then removed. This ideally indicates terms which are momentarily relevant perhaps due to a real-world event, but not core, always-relevant terms. For example, the term “globalists” was qualified for around 30 minutes. A number of news articles had been published describing the [EU](#) elections as “nationalists vs globalists” and quoting Marine Le Pen (a French politician) saying “Polarisation between nationalists and globalists will last”, and this was picked up on [TWITTER](#). But after a while, the tweets mentioning “globalists” are far more gen-

eral than just Brexit or EU politics, so the term is removed. A similar example is “macron”.

However, there are some terms like “remainer” which we might expect to remain relevant for longer but which were removed after a while. Though the framework did retain “remainers” and “a remainer”. When desired terms are removed over time, this could be an indication that we should lower the required relevancy threshold or increase the expiry time in order to raise recall of actually relevant documents. In future versions of the framework, details like how well terms pass the relevance test, and how often a term has been qualified will be more exposed to the user to help them make these decisions. It may also be beneficial to dynamically alter the expiry time based on the total volume of tweets obtained to avoid losing relevant terms that are currently undergoing low traffic. Furthermore, while it is possible using METHOD52 for the user to find a random sample of tweets mentioning a term in order to help diagnose why it was removed from the query, this could be made simpler by storing a small random sample of documents specifically around each query decision so that it can be presented to the user immediately on request.

Phrase	Removed
ffs	15:11:47
for the libdems	15:31:41
bollocks to brexit	15:31:41
farage's brexit party	15:31:41
lib dems	15:31:41
remainer	15:31:41
the globalists	15:31:41
brexit party	15:31:41
heartlands	15:31:41
revoke article 50 petition	15:31:41
macron's	15:51:41
nigel farage	15:51:41
anna soubry	16:11:43
nigel farage's brexit party	16:11:43
deluded	16:25:56
farage's	16:31:44
globalists	16:31:44

Table 5.3: The above are all phrases that were removed in the two hours after starting the adaptive streaming. Some are terms which were once qualified but then no longer produced sufficient relevant data to remain in the query (such as “deluded”) and others were never qualified because outside the context of the term “brexit” their usage is too general, and therefore irrelevant (such as “ffs”).

Table 5.2 shows when added terms passed the relevance test and first became qualified. Therefore, these are the terms in addition to “brexit” which produce the qualified output dataset. To give an idea of what these additional query terms gain us, a random sample of 20 tweets produced by qualified terms excluding “brexit” is shown in Table 5.4. The majority of the tweets are useful, relevant additions to the Brexit dataset. There is, however, one clearly irrelevant tweet about Canadian wildfires and carbon tax, because the URL it shared contained the qualified term “the tories”, so TWITTER returned it. Since we do not use the text of URLs in classifier training or SFPD, it may be sensible to exclude this kind of tweet from the analysis anyway. Additionally, we could also apply the relevancy classifier together with the qualified status to filter this kind of tweet from the output. The Brexit relevancy classifier does indeed mark this tweet as irrelevant, but it incorrectly marks a number of the others as irrelevant also. So it is a precision/recall trade-off.

Next we examine which phrases were qualified over the following two days. Table 5.5 shows the phrases and their period of qualified status, and notes explaining their relevance. A number of useful terms are identified, including: “the erg”, the European Research Group focused on the UK’s withdrawal from the European Union (EU); “fptp” found in discussion about how first-past-the-post is to blame for Brexit’s success; and “bercow” as Speaker John Bercow became a topic of conversation for suggesting that no-deal Brexit cannot be forced through against the will of parliament.

Interestingly, some of the qualified terms (“the ehrc”, “anti semites”) linked into the discussion of anti-semitism in the Labour party, which might seem to be straying from the Brexit discussion, but it is linked through people’s voting habits:

@user That’s right my vote for the @LibDems was as much about now supporting a party that feels safe & welcoming for Jews as it was about Brexit and the climate emergency - also sick of a two party #fptp ‘democracy’ stitched up by @UKLabour & @Conservatives

If this is a tangent too far for the analyst, then the relevancy classifier can be updated to consider such phrases indicative of irrelevance. We could even resort to using the CUSTOM LOGIC ANNOTATOR to identify these phrases and always mark the containing document as irrelevant.

For a period of just over an hour in the early hours of the second day, the term “trump’s” was qualified. Relevant discussion in the form of comparisons preceded it, e.g:

@user Kind of explains why most Brexit Party supporters buy into Poundshop-Trump’s BS. Most of them wouldn’t be able to cope with the big words in this passage anyway. Some of the ‘University of Life’ types would probably stop reading at “accustomed”.

However, such a hugely popular term on TWITTER then drew in large amounts of irrelevant Donald Trump-related tweets. The relevancy classifier failed to stop these for just over an hour. This could suggest that either the classifier needs more Trump-aware training data, or the 30% relevance rate is too lenient. We could also consider making it harder for popular terms to enter the query using an up-to-date sample of current tweet traffic, but it can be hard to distinguish this data from a query that is purposefully examining trending topics like Brexit. While this data is easily filtered out using the CUSTOM LOGIC ANNOTATOR and the query at the time, a warning to the user for an influx of data on a new query, and an simple button for automatically cleaning the data if this is undesired would be useful. The “smears” term introduces the same problem.

A slightly different variation of this problem is the “revoke” term. It repeatedly arises in the “revoke article 50” conversations, and appears sufficiently frequently in this domain that it gets qualified multiple times for small stretches of around an hour. These times do seem to correlate with Article 50 conversation, but they do begin to encounter a irrelevant data. The term usually scrapes just above the 30% relevance, so this could be another indication of the need to raise the threshold for this data. Another indicator that future versions could use to automatically notify the user, is the fact that the term is found to be relevant many times in short bursts. It is unlikely that so many bursts are being instigated by new real-world events, and more likely that the feature is on the edge of a threshold.

If we had acquired data from TWITTER using only the query term “brexit”, then from 14:46 on Monday 27th May, 2017 until midnight Thursday, we would have collected 923,932 tweets. However, after removing “trump’s”, “smears”, and non Article 50 mentions of “revoke”, the adaptive streaming approach recalled an additional 562,598 that could be of interest to the analyst. The idea behind the framework is not to replace the human in data collection, but to support

them in gaining greater coverage of the domain. Therefore, this sanity-checking of phrases to find a few outliers is an expected part of the methodology.

A topic tracking system with similar aims for exploring broad topic areas as this [ATS](#) framework was introduced by [Magdy and Elsayed \(2014\)](#). In order to quantitatively estimate the usefulness of their framework’s query adaptation, the authors first evaluated their system using precision on a hand-annotated random sample of 100 documents from their framework’s output. Then using this precision, they estimated a “relative recall”. It is not possible to calculate recall, since it is not known how many possible relevant tweets there could have been, but *relative* recall, uses a precision measure and the number of extra documents found by the framework to estimate how much the recall has increased from baseline (hence “relative”). Equation 5.1 shows how relative recall is calculated:

$$Recall_{rel} = \frac{D_{adaptive} \cdot precision(adaptive)}{D_{baseline} \cdot precision(baseline)} \quad (5.1)$$

Where $D_{adaptive}$ is the number documents produced by the adaptive system in total, and $D_{baseline}$ is the number of documents the system produces without adaptation (i.e. for this case, this is the number of documents produced by the “brexit” query term alone).

The precision of the adaptive system is estimated with a random sample of 100 tweets taken from the extra 562,598 and annotated for relevance to Brexit (a more lenient evaluation would sample from all the system’s output, including those tweets produced by the “brexit” query term alone). The baseline precision could be estimated in the same way, but in this experiment, anything containing the word “Brexit” is by definition relevant, so we

A total of 62 were directly discussing issues involved in Brexit, 9 were discussing the anti-semitism of the Labour party, 19 were personal remarks about Jeremy Corbyn (who himself had become a query term due to his presence in Brexit discussion), 3 were about European politicians involved in the [EU](#) elections, and 7 were completely irrelevant. Therefore, this is a precision of up to 93% depending on whether the analyst is interested in these extra related categories of tweet. If the extra categories are not useful, then they can be removed by keyword search and filtering. The only complication would be if e.g. *some* of the Corbyn remarks were useful and others were not. This would require adaptation of the relevancy pipeline (e.g. keyword filtering, classifier training, etc.). This is a relative recall of

149.63%, which is a decent amount of extra signal that would have been missed with a query consisting only of “brexit”. The precision would have been lower if the tweets produced by “trump’s”, “smears” and non Article 50 mentions of “revoke” had not been removed (a total of 12,636, or 2%), but as noted above, this simple procedure is an expected part of the methodology of keeping the user in the process. Magdy and Elsayed (2014) found that the recall varied greatly across topic domain, finding a range of recall increase from 32% to 200%.

The Brexit relevancy classifier used for the adaptive streaming considers 71% of the additional 562,598 to be relevant, which is substantial. However, we must be cautious with this estimate of the usefulness of the output. The classifier’s notion of relevance is very much tied to both the initial un-adapted training data, and our 30% relevancy threshold for newly proposed terms producing the output. So we can trivially raise this estimate by requiring a higher relevance rate for a proposed query terms to be qualified, so that we accept fewer riskier terms.

<p>@johnmcdonnellMP @user Time to remove the Blairite remoaners from the Labour Party they have been exposed as liars in the betrayal of the greatest democratic vote in our history. #removeBlairiteremoanersfromLabourParty</p>
<p>USA - Donald Trump Australia - Scott Morrison Brazil - Jair Bolsonaro France - Marine Le Pen Hungary - Viktor Orban India - Narendra Modi Italy - Matteo Salvini Poland - Andrzej Duda UK - Nigel Farage The right is winning all over the world. #EUElections2019 #EP2019</p>
<p>Trump was the beginning. Now the globalist one nation movement is in trouble all over the world. Support has blossomed for Nationalist candidates like Salvini in Italy, Le Pen in France, Farage in UK, Tarczynski in Poland, Morrison in Australia and more. Globalism is dying.</p>
<p>Orban, Salvini, Farage, Le Pen Retweet if you are happy with the result and follow @UnityNewsNet for real news. HTTPLINK</p>
<p>@user @user @user We should have done a Remain Alliance with the Lib Dem's and Greens and whoever other party or/and individual that wanted to join. That was the single best idea Farage had this election and STILL %-wise, Remain won.</p>
<p>A reminder of how European elections are an unreliable guide to general elections: In 2014 Ukip got 27% but then achieved below 13% at the 2015 general election In 2014 Labour beat the Tories (25:24) only to be thrashed by them at 2015 general election (37:30)</p>
<p>BXP results make a new Tory leader backing No Deal by Oct 31st position almost certain... But... 9.1 million people just voted for parties explicitly against No Deal (LD/G/L/CH/SNP...) ... vs 5.8m explicitly in favour...(BXP+Ukip) /1</p>
<p>No Morals or principles just a vote catcher... Jeremy Corbyn appears to change Labour policy as he backs second referendum on any EU deal HTTPLINK via @Telegraph</p>
<p>Who is Nigel Farage? What's his relationship to Trump? And why did he lie about his meeting with Roger Stone? (Who's now awaiting trial). This is what you won't be reading in the press today. From November but nothing's changed.. HTTPLINK</p>
<p>Leave wins 52% in the EU referendum in 2016. Remoaners: but not by enough. Leave wins 80%+ the 2017 general election. Remoaners: but it wasn't clear. Leave wins 2019 European elections. Remoaners: but the combined vote of the Remain losers is higher. Anyone seeing a pattern here?</p>
<p>Jeremy Corbyn has now lost: - a general election - two sets of local elections - a European election - a confidence vote from his own party - over 100 MPs from his front bench - 8 MPs to ChangeUK But apparently it's outrageous to suggest he should resign.</p>
<p>An interesting take on the EU results! Forget what the BBC and Remainers say, this is based on manifesto promises..... HTTPLINK</p>
<p>@user @user @user @user @UKLabour You should apply this to everyone rather than accusing people of rape just because they disagree with you politically. The evidence you have presented finds you an ignorant Corbyn cultist; not good vibes for happy families. Could you be taking out your anger at your son on me?</p>
<p>@PeperHade & Co, Farage just called the bluff of the Conservative and Labour. Interesting times ahead.</p>
<p>NATIONALISTS WIN EU Elections! - Farage wins in UK. - Le Pen wins in France - Salvini wins in Italy. - Orban wins in Hungary. - Right wing surging in Sweden. Massive success for the right-wing and EU skeptics this election! @POTUS HTTPLINK</p>
<p>BC has the worst wildfires in all of Canada. We have also had Canada's highest and oldest carbon tax for over 10 years now. Why are B.C. wildlife's only getting worse despite our carbon tax? Help me out here @CBC HTTPLINK</p>
<p>@Trickyjabs If she has Farage's attendance record they won't be seeing a lot of her</p>
<p>Hi @UKLabour, As you're still contemplating which side of the fence to sit on let me help you with the decision: Are you on the same side as the xenophobes who heckled your own MEP and told her to "go home"? Or are you on the right side? HTTPLINK</p>
<p>Sajid Javid to run for Tory party leader No thanks He said he was going to stop illegal immigrants arriving in dinghies, he then sent out boats to pick them up and hasn't returned any to France. We want a PM that's actions speak louder than words HTTPLINK</p>
<p>WATCH @user: "It is not only Lega that is the first in Italy, Nigel Farage is first in Britain. It is the sign of a Europe that is changing, of a Europe tired of the powers of the elites and multinationals. Tomorrow, we will have to redouble our efforts!" #EP2019 HTTPLINK</p>

Table 5.4: A random sample of documents produced by qualified terms after two hours. Most are related to the Brexit debate, or how the EU elections affect it. There is, however, a clearly irrelevant tweet about Canadian wildfires because the URL when resolved contains the term “the Tories”, which was a qualified term.

Term	Day	Start	End	Notes
fptp	Tues	08:39:16	12:52:45	First-past-the-post is blamed for Brexit's success
the erg	Tues	09:52:10	11:32:40	European Research Group whose focus is the UK's withdrawal from the European Union
michael gove	Tues	09:16:29	10:12:37	Michael Gove pledges free UK citizenship for 3m EU nationals if he becomes PM
	Tues	11:14:11	15:12:51	
	Tues-Weds	16:50:45	01:33:22	
the ehrc	Tues	12:11:00	01:20:57	Equality and Human Rights Commission launches investigation into Labour party anti-semitism
kate hoey	Tues-Weds	12:59:42	04:33:31	Kate Hoey MP was criticised for sharing platform with Nigel Farage during the Brexit referendum campaign
labour expels alastair campbell from	Tues	13:25:42	14:32:50	Alastair Campbell expelled from Labour for backing Lib Dem in EU elections.
heseltine	Tues	13:27:35	17:12:57	Michael Heseltine was suspended last week by the Conservatives for revealing he will vote Lib Dem. Arising in discussions mentioning Alastair Campbell
anti semites	Tues-Weds	13:33:53	03:33:28	Labour party accused of anti-semitism
the blairites	Tues	15:09:44	15:32:53	Conversations about whether Tony Blair supporters ruined or saved the Labour party in the context of whether Jeremy Corbyn ruined or saved the party
bercow	Tues	16:12:17	17:12:57	Speaker John Bercow suggested that no-deal Brexit cannot be forced through against the will of parliament.
trump's	Weds	02:11:38	03:29:02	The only consistently mentioned term at this time of night. Classifier had "trump" as an irrelevant token, but not this.
john bercow	Weds	06:40:23	08:01:20	A lot of talk about John Bercow being biased.
kate hoey	Weds	07:23:51	19:34:16	Continued discussion about Kate Hoey and whether she should be expelled too.
michael gove	Weds	08:03:17	11:33:53	Ongoing discussion of Michael Gove suitability for leadership and PM
	Weds-Thurs	14:54:34	00:34:31	
revoke	Weds	Short bursts all day.		Always occurs in context of revoking article 50, but is also a very generic term.
expelling	Weds	08:24:58	15:34:05	Discussion about Alastair Campbell about expulsion from political parties
bojo	Weds-Thurs	12:42:44	00:54:33	Boris Johnson ordered to appear in court over £350m claim during Brexit campaign. Discussions for leadership suitability.
smears	Weds	14:57:07	17:29:12	Became popular phrase about anti-Labour and anti-Corbyn sentiment, but outside of Brexit context includes a lot of other politics.

Table 5.5: A summary of qualified terms over the next two days (Tues 28th and Weds 29th of May), with notes explaining the relevance of the term.

5.11 CHAPTER SUMMARY

This chapter introduced and refined a framework for adaptively streaming TWITTER data, a process which is broadly applicable to other domains where term-based queries are used to acquire corpora, in order to answer the following research question:

How can feature discovery support the collection of data relevant to classes of text?

The framework is inspired by topic tracking literature and keyword extraction techniques in corpus linguistics. Its aim is to aid analysts' exploration of data, by discovering terms in the data that the analyst would find interesting and like to know more about, and then introduce these back into the data collection method. By partially automating this cycle of discovery and modification of collection methodology, the analyst develops a fuller picture of their topic of interest.

The assumptions of the framework about the structure of topics and how they are discovered were examined.

In an interactive loop, the framework streams data from TWITTER using the user's initial query, and passes the data to an SFPD analysis, which in turn sends instructions to the streaming component to adapt its TWITTER query.

User-trained classifiers (or any METHOD52 strategy for determining the relevance of documents) are used in the loop to administer relevancy tests for documents produced by each query term, so that the framework can retain some relevancy precision and avoid drifting off-topic. Irrelevant terms, or terms that have not produced sufficient data to justify their place in the query are removed and potentially blacklisted if their relevance is low enough. A technique was discussed for ensuring that relevancy classifiers can generalise to the broader sample of documents that can be expected from an adapting query.

Strategies for adapting and improving ATS by manipulating the reference corpus in SFPD were also discussed.

ATS is currently being used in a funded project, where it is undergoing more refinement. In its current state, it is capable of discovering newly arising discussion topics among the specified broader topics. A well-trained relevancy classifier, and suitably chosen reference corpus stops the query from becoming trapped on trending topics or overly general topics.

The system is also good at finding the different ways of referring to entities within the topic. This was demonstrated by allowing the framework to explore using the keyword "brexit". Not only does a stream searching on "Brexit" discover the term "remainers" for those who are against Brexit, but also the nickname used for them by pro-Brexiteers: "remoaners". And it discovers keywords that map to issues arising in the real world.

If the notion of relevance is very narrow or rare, and the seed permanent terms are comprehensive, the framework is more likely to

struggle to find qualifiable query terms. In the [DD](#) study, news of a chemical attack is related to extremists and discovered by [ATS](#), but isn't necessarily well-discussed by the extremists themselves, which was the analysts' actual notion of relevance.

CONCLUSIONS

In efforts to analyse large datasets, social scientists and humanities scholars face a challenge to find and isolate the data they are interested in order to perform highly bespoke analysis, and must turn to automated tools in order to better engage with the data. `METHOD52` (Wibberley et al., 2013, 2014) is a tool built for this purpose, and served as a foundation for this thesis to explore methods for improving engagement with this type of data.

This thesis began with a discussion of the general process of data engagement that `METHOD52` users undertake when using it. Broadly, it consists of two types of activity: (1) the discovery and definition of classes of data, the “explore” methodology, and (2) the modelling of classes in order to apply automated tools to isolate the classes from the rest of the data, the “search” methodology. And furthermore, the analyst alternates between these two in a cycle as the analysis proceeds deeper into the data.

The notion of a class of documents is at the core of the analysis. It was decided to take a bottom-up approach to supporting better discovery and isolation of classes. This approach emphasises the importance of surface features of the text, and that fundamentally the discovery of important phrases in a text can generate insight concerning the possible class definitions that could be defined over the data. This thesis therefore asked the following questions:

1. How can features be identified which provide useful bases for characterising classes of text?
2. How can feature discovery support the identification, definition, and characterisation of classes of text?
3. How can feature discovery support the collection of data relevant to classes of text?

To answer 1, the first contribution of this thesis was the introduction of a flexible method for discovering key phrases in text, which was named Surprisingly Frequent Phrase Detection ([SFPD](#)). The method can be directly adapted to different domains by establishing a suitable

reference corpus. The reference corpus provides a way of allowing analysts' prior knowledge to influence which features are interpreted as interesting. *SFPD* was applied for aspects of both contributions for 2 & 3.

For question 2, uses of *SFPD* were explored to improve the process of discovering and isolating classes. Techniques were used in conjunction with *METHOD52*'s classifier component, since the classifier is a core component for enabling the analyst to isolate of classes of data. By working through the Brussels example, several *SFPD* strategies were introduced and demonstrated to be useful for speeding analysis, and making classifier building a more informed process.

Furthermore, problems identified in *METHOD52*'s existing feature querying and labelling process (during active learning) were alleviated by introducing syntactic ngram features produced with a *TWITTER*-adapted dependency parser, and allowing users to sample the original contexts of features. A less naïve approach to incorporating feature labels into the classifier model was also introduced.

Question 3 is an important issue because when acquiring social media data, it is often the case (and indeed it is with *TWITTER*) that the data is collected using a query consisting of words and phrases. Therefore, the word and phrase features are even defining the space of possible classes, since they govern what data is actually collected. Given that *SFPD* showed a promising ability to discover key phrases describing topics of interest, a framework was built to use these phrases in an automatically adapting query, based on the phrases found in the streamed data.

Firstly, the assumptions about topic structure of an *SFPD*-based adaptive streamer were discussed. Then, using *DD* as a case study, the issues with the adaptive approach were investigated and discussed. It was determined that a careful encoding of relevance was necessary to maintain query precision, that near-duplicate detection was useful, and that there had to be multiple paths to terms being removed from the *TWITTER* query in order to make most use of *TWITTER*'s data within the limits they impose. Furthermore, the real-time streaming nature of the problem introduced new possibilities for domain adaptation.

Ultimately, a framework was built which works well for broad topics that experience updates according to events in the real-world. Well-performing relevancy classifiers ensure that precision is main-

tained in the face of much irrelevance in cases like [DD](#) where relevance is a very narrow notion, but can result in very little adaptation.

Together, these contributions represent new and improved strategies for obtaining more value from word and phrase features during data collection and the discovery and isolation of classes of data.

6.1 FUTURE WORK

The various avenues for direction of future work are considered below.

6.1.1 *Feature Discovery and Characterisation of Classes of Text*

6.1.1.1 *Feature Exploration*

Currently, the default use of `SFPD` in `METHOD52` extracts one phrase per surprising word found. The other top phrases for a given word tend to be variations on the theme of the topmost phrase, phrases that are essentially the same but that start from a different token in the phrase. Future work would develop a measure for determining both whether there are other interesting phrases, and whether they are distinctive enough to be interesting to report.

It would also be interesting to experiment with how different measures of surprise and the phrases they would produce would affect the corpus exploration.

Syntactic ngrams are only used in the classifier's feature extraction and high-Information Gain (`IG`) feature querying; the `SFPD`'s phrase expansion only expands to adjacent tokens. It would be worth exploring whether phrase expansion can produce useful results operating over syntactic dependencies. Common constructions could be discovered independent of word order problems of using only adjacent tokens. "Islamophobia is rife" and "islamophobia is utterly rife" would both count toward the frequency of the query term "islamophobia rife" (a query which would find any tweets containing both terms), since "islamophobia" and "rife" would be directly connected via the `NSUBJ` dependency relation. The strategy's usefulness is likely to be task dependent, there is a chance that terms not syntactically related could be more likely to be indicative of the discussions analysts wish to follow. Furthermore, introducing dependency parsing means that its performance on `TWITTER` language will affect phrase extraction.

6.1.1.2 *Feature extraction*

It is unclear whether users should be exposed more directly to how features were extracted from a document, for example by showing visually how features were extracted from the text. In some few cases,

users described suggesting a feature to the system’s original contexts interface which they expected to occur in documents they have seen during training, but found that it does not. This could be due to a misunderstanding of how the feature extraction process works, or an error in one of the models producing the features (e.g. the dependency parser failed to find a dependency relation that should be present). Showing the actual extraction process, or at least listing the features produced from any selected document could make this clearer. But the added complexity may slow the intuitive process that works well most of the time.

Currently, when extracting syntactic ngrams, prepositions are collapsed to ensure that there is more emphasis on meaningful words. The process could be extended to other less common relations. Alternatively, the ngrams could be extracted after some more general form of semantic role labelling has been applied using the syntactic analysis.

AS TWITTER increases their character limits for tweets, so they less resemble single sentences, dependency parser performance is likely to drop, since the dependency parser is trained on single full sentences. This suggests a requirement for a sentence segmenter adapted for TWITTER’s fragmented sentences, or a parser trained on similar structures (as TWEEBOPARSER is (Kong et al., 2014)).

Waseem and Hovy (2016) find *character* ngrams more useful to their model than word ngrams in their study of hatespeech on TWITTER, which is attributed to their reduced sparsity. Other research shows that character ngrams can help to account for common unusual spellings (Zhang and Luo, 2018). It would be interesting to incorporate character ngram features, though these may be less intuitive to the actual analysts.

Currently, despite quite complex criteria for classification definitions (such as sentiment toward a target), the classifier component is relied upon to discover patterns in ngrams alone to make these classification decisions. It could benefit from more feature extraction steps to identify common complications of online messages, such as identification of opinion target, or encoding how opinions are modified by phenomena like negation or sarcasm, for example as in work by Maynard et al. (2012).

6.1.1.3 *Feature incorporation*

While the scheme of weighting pseudo-counts by indicativeness is logically useful, since it ensures the user has a better idea of how indicative a feature really is of a classification, more case studies are required to confirm that the weighting is effective for classifier performance.

Feature selection is an area unexplored in this thesis. In general, classifiers have performed well enough in the DUALIST style of active learning. And confusion is avoided of users expecting features to be present when seen in documents. A separate study would be useful to test whether the benefits of feature selection methods would outweigh any user confusion.

Lastly, as previously stated, users tend not to want to include very long phrases into the actual classifier model, reasoning that shorter versions of the phrases introduce only a little ambiguity compared to how much sparsity the longer phrase is likely to introduce. It would be interesting to determine whether the ambiguity can be reduced without introducing sparsity, by allowing phrase *patterns* to be specified. For example, “trump was * about that”, where the asterisk could be any token(s). Then the feature extraction process would search for the pattern, and extract a feature based on it in texts that contained it. This would permit the user to annotate the pattern as indicative of a particular classification and assign it a weighting under the model. The patterns could operate over simple windows of tokens, or could even be patterns over syntactically related tokens, though this may be too much complexity for the user (unless patterns can be proposed to the user using SFPD syntactic phrase expansion or similar).

As identified in the introduction, the strategies of this thesis very much represent a bottom-up surface-feature-driven approach. After exploring all these avenues it would be interesting to combine with a more top-down approach, to bring in the advantages of utilising knowledge bases and semantic search, similar to (Maynard et al., 2017).

6.1.2 *Adaptive Twitter Streaming*

The adaptive streaming framework will be applied and refined in more case studies. For the next refinements, of particular interest is an investigation of other methods for providing relevance information to

the adaptive streaming process. The `METHOD52` influence network analysis component (based on `AUDCLUS` (Lin et al., 2015)) was created with such goals in mind. Saleem et al. (2017) define relevance by modelling language in defined communities like subreddits. `TWITTER` does not have such defined communities, but if a community could be inferred it might be able to support a similar analysis. The method of Lin et al. (2015) is particularly interesting since it is well-suited to inferring influence based on actual user actions, rather than a static friend/follower network.

Another exciting area for exploration is in the phrase generation process which extends surprising words to phrases. The `TWITTER API` does not require that the constituent terms of a query phrase appear contiguously in a matched document; the words of the phrase must simply all be present somewhere in the text. It could be that the most relevant context of a surprising word is not necessarily the immediately adjacent words, and perhaps some form of phrase expansion based on syntactic analysis would pick up these more relevant context words.

As discussed in Section 5.1, some existing topic tracking systems make use of the text in the `URLS` and their linked content. These are obviously indicative of the resources that are being shared online and the topics initiating discussion, but only indirectly about what is actually being discussed by `TWITTER`'s users, what terms are actually being used by users. So while useful for a topic modelling approach, may not be as immediately useful to find actual words with which to query `TWITTER`. It is slightly more complicated than this, since `TWITTER` does actually inspect tokens within `URLS` for the presence of query terms. However, the words used in `URLS` are not constrained to have to make sense in a sentence or discussion, so could introduce a lot of noise. Either way, this warrants further study.

Another area for study would be investigating how an indefinitely growing reference corpus affects the `SFPD` measure, and therefore how necessary pruning old count information is.

Different methods of term expiry should be explored. It needs to be determined whether using the surprisingness of a term is sufficient to govern when it should expire, and how the overall volume of data should affect the expiry.

Lastly, more succinct reporting of the framework's activity would be beneficial to the user. It would be useful, for example, to have a summary of how long and often each phrase has been qualified

or expired, and statistics like average and range of relevance rate, and a random sample of documents containing the phrase. These elements would improve the ability of the user to better tune the various parameters to their needs.

6.2 LIMITATIONS

The main limitation of this thesis lies in the evaluation strategy for the contributions of feature discovery methods. For some contributions (e.g. the introduction of the surprising phrase discovery method in Chapter 3), it is possible to evaluate through comparison to other similar methods or presenting and analysing output. Other contributions show clear theoretical benefit, such as the technique for weighting features in Section 4.7.2. However, some contributions introduce techniques which are fundamentally exploratory, and which would benefit from further study of user interaction to understand the impact (such as the introduction of syntactic ngrams to solve the “contextual indicativeness” problem in Section 4.6). In the evaluation of such contributions, a trade-off is found.

One possibility is that for every potential new setting or additional feature to be evaluated, user focus groups are formally conducted in order to precisely pin down the effectiveness of the addition. The limitation of this strategy is that it is much slower to cover many possibilities and more challenging to set up to reflect real-world usage of a system. This thesis instead focuses primarily on theoretical argument and comparison, using only informal/anecdotal feedback from METHOD52 analysts in the context of projects already using METHOD52.

While this approach has the advantage of being able to explore many approaches quickly, and permit evaluation within the context of ongoing real projects, it produces less strong conclusions for the effectiveness of the approach in terms of the impact on users than formal user testing would.

ACRONYMS

ATS Adaptive Twitter Streaming / Streamer

API Application Programming Interface

BNC British National Corpus

CASM Centre for the Analysis of Social Media

DD Disrupting Daesh project

EM Expectation-Maximisation

EU European Union

HMIC Her Majesty's Inspectorate of Constabulary

IDF Inverse Document Frequency

IG Information Gain

IS Islamic State

IR Information Retrieval

KL Kullback–Leibler

LDA Latent Dirichlet Allocation

LL Log–Likelihood

LOB Lancaster-Oslo-Bergen corpus

MI Mutual Information

MNB Multinomial Naïve Bayes

MNB-FM Multinomial Naïve Bayes with Feature Marginals

NB Naïve Bayes

NBC Naïve Bayes Classifier

NLP Natural Language Processing

PEEL Police Effectiveness, Efficiency and Legitimacy

PoS Part of Speech

PRF Pseudo-relevance Feedback

PTB Penn Treebank

SFPD Surprisingly Frequent Phrase Detection

SVM Support Vector Machine

TAG Text Analytics Group

TDT Topic Detection and Tracking

TF-IDF Term Frequency–Inverse Document Frequency

TIDES Translingual Information
Detection, Extraction, and Summarization

UI User Interface

UoS University of Sussex

TREC Text Retrieval Conference

WSJ Wall Street Journal

BIBLIOGRAPHY

- W. D. Abilhoa and L. N. De Castro. A keyword extraction method from twitter messages represented as graphs. *Applied Mathematics and Computation*, 240:308–325, 2014. (Cited on pages 38 and 42.)
- A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, pages 30–38. Association for Computational Linguistics, 2011. (Cited on page 140.)
- B. Agarwal, S. Poria, N. Mittal, A. Gelbukh, and A. Hussain. Concept-level sentiment analysis with dependency-based semantic parsing: a novel approach. *Cognitive Computation*, 7(4):487–499, 2015. (Cited on page 139.)
- A. V. Aho and M. J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975. (Cited on page 133.)
- J. Allan, J. G. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study final report. 1998. (Cited on page 174.)
- F. Alvanaki, S. Michel, K. Ramamritham, and G. Weikum. See what’s en-blogue: real-time emergent topic identification in social media. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 336–347. ACM, 2012. (Cited on pages 181 and 182.)
- A. Angel, N. Koudas, N. Sarkas, and D. Srivastava. What’s on the grapevine? In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 1047–1050. ACM, 2009. (Cited on page 176.)
- S. Arora, E. Mayfield, C. Penstein-Rosé, and E. Nyberg. Sentiment classification using automatically extracted subgraph features. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 131–139. Association for Computational Linguistics, 2010. (Cited on page 139.)
- A. Balahur. Sentiment analysis in social media texts. *WASSA 2013*, page 120, 2013. (Cited on page 142.)
- A. Baron, P. Rayson, and D. Archer. Word frequency and key word statistics in corpus linguistics. *Anglistik*, 20(1):41–67, 2009. (Cited on pages 41 and 50.)
- M. Baroni and S. Bernardini. Bootcat: Bootstrapping corpora and terms from the web. In *LREC*, page 1313, 2004. (Cited on pages 43, 53, 54, 58, 59, and 186.)
- J. Bartlett, J. Reffin, N. Rumball, and S. Williamson. Anti-social media. *Demos*, pages 1–51, 2014. (Cited on pages 11, 29, and 45.)
- S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010. (Cited on page 16.)

- D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003. (Cited on pages 42 and 181.)
- A. Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE, 1997. (Cited on page 195.)
- P. Buddhitha and D. Inkpen. Dependency-based topic-oriented sentiment analysis in microposts. In *SIMBig*, pages 25–34, 2015. (Cited on page 141.)
- M. Cataldi, L. D. Caro, and C. Schifanella. Personalized emerging topic detection based on a term aging model. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):7, 2013. (Cited on page 181.)
- K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990. (Cited on pages 41 and 51.)
- M. Coletto, C. Lucchese, S. Orlando, and R. Perego. Polarized user and topic tracking in twitter. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 945–948. ACM, 2016. (Cited on page 181.)
- M. Conway, M. Khawaja, S. Lakhani, J. Reffin, A. Robertson, and D. Weir. Disrupting daesh: measuring takedown of online terrorist material and it’s impacts. 2017. (Cited on pages 11, 45, 69, 125, and 182.)
- L. Crawford, J. Pollack, and D. England. Uncovering the trends in project management: Journal emphases over the last 10 years. *International journal of project management*, 24(2):175–184, 2006. (Cited on page 41.)
- B. Daille. Combined approach for terminology extraction: lexical statistics and linguistic filtering. 1995. (Cited on pages 40 and 51.)
- K. Dave, S. Lawrence, and D. M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM, 2003. (Cited on pages 139 and 140.)
- L. Di Caro and M. Grella. Sentiment analysis via dependency parsing. *Computer Standards & Interfaces*, 35(5):442–453, 2013. (Cited on page 141.)
- Q. Diao, J. Jiang, F. Zhu, and E.-P. Lim. Finding bursty topics from microblogs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 536–544. Association for Computational Linguistics, 2012. (Cited on page 182.)
- Y. Duan, F. Wei, M. Zhou, and H.-Y. Shum. Graph-based collective classification for tweets. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2323–2326. ACM, 2012. (Cited on pages 63 and 180.)
- T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):61–74, 1993. (Cited on pages 40 and 51.)
- M. Efron. Hashtag retrieval in a microblogging environment. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 787–788. ACM, 2010. (Cited on page 176.)

- D. Eichmann, M. Ruiz, P. Srinivasan, N. Street, C. Culy, and F. Menczer. A cluster-based approach to tracking, detection and segmentation of broadcast news. In *Proceedings of the DARPA Broadcast News Workshop*, pages 69–76, 1999. (Cited on page 174.)
- C. Fellbaum. *WordNet*. Wiley Online Library, 1998. (Cited on page 177.)
- E. Fredkin. Trie memory. *Communications of the ACM*, 3(9):490–499, 1960. (Cited on page 54.)
- R. Gacitua, P. Sawyer, and P. Rayson. A flexible framework to experiment with ontology learning techniques. *Knowledge-Based Systems*, 21(3):192–199, 2008. (Cited on page 41.)
- M. Gamon. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of the 20th international conference on Computational Linguistics*, page 841. Association for Computational Linguistics, 2004. (Cited on pages 139, 140, and 142.)
- S. Garcia Esparza, M. P. O’Mahony, and B. Smyth. Towards tagging and categorization for micro-blogs. In *Paper presented at the 21st National Conference on Artificial Intelligence and Cognitive Science (AICS 2010), Galway, Ireland, 30 August-1 September, 2010*, 2010. (Cited on page 180.)
- K. Gimpel, N. Schneider, B. O’Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics, 2011. (Cited on pages 93 and 146.)
- A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12, 2009. (Cited on page 140.)
- S. Granger and P. Rayson. Automatic profiling of learner texts. *Learner English on computer*, pages 119–131, 1998. (Cited on page 40.)
- Z. Han, X. Li, M. Yang, H. Qi, S. Li, and T. Zhao. Hit at trec 2012 microblog track. In *TREC*, volume 12, page 19, 2012. (Cited on page 177.)
- A. Hogenboom, D. Bal, F. Frasinicar, M. Bal, F. de Jong, and U. Kaymak. Exploiting emoticons in sentiment analysis. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 703–710. ACM, 2013. (Cited on page 148.)
- L. Hong and B. D. Davison. Empirical study of topic modeling in twitter. In *Proceedings of the first workshop on social media analytics*, pages 80–88. ACM, 2010. (Cited on page 182.)
- M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004. (Cited on page 139.)
- A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65. ACM, 2007. (Cited on page 41.)

- L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao. Target-dependent twitter sentiment classification. In *ACL*, pages 151–160, 2011. (Cited on page 139.)
- H. Jin, R. Schwartz, S. Sista, and F. Walls. Topic tracking for radio, tv broadcast and newswire. In *Proceedings of the DARPA broadcast news workshop*, pages 199–204. San Francisco: Morgan Kaufmann, 1999. (Cited on page 174.)
- A. Jordanous and B. Keller. Modelling creativity: identifying key components through a corpus-based approach. *PloS one*, 11(10):e0162959, 2016. (Cited on pages 38, 41, 47, and 51.)
- M. Joshi and C. Penstein-Rosé. Generalizing dependency features for opinion mining. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 313–316. Association for Computational Linguistics, 2009. (Cited on pages 139, 140, 141, and 142.)
- J. Kalyanam, S. Velupillai, M. Conway, and G. Lanckriet. From event detection to storytelling on microblogs. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 437–442. IEEE, 2016. (Cited on page 176.)
- A. Kilgarrieff. Using word frequency lists to measure corpus homogeneity and similarity between corpora. In *Proceedings of ACL-SIGDAT Workshop on very large corpora*, pages 231–245, 1997. (Cited on pages 39 and 40.)
- A. Kilgarrieff. Comparing corpora. *International journal of corpus linguistics*, 6(1):97–133, 2001. (Cited on pages 41, 50, and 51.)
- A. Kilgarrieff and T. Rose. Measures for corpus similarity and homogeneity. In *EMNLP*, pages 46–52, 1998. (Cited on pages 81 and 82.)
- Y. Kim, R. Yeniterzi, and J. Callan. Overcoming vocabulary limitations in twitter microblogs. Technical report, Carnegie-mellon Univ Pittsburgh Pa Language Technologies Inst, 2012. (Cited on pages 50 and 178.)
- L. Kong, N. Schneider, S. Swayamdipta, A. Bhatia, C. Dyer, and N. A. Smith. A dependency parser for tweets. 2014. (Cited on pages 147 and 229.)
- A. Krasodonski-Jones. Talking to ourselves? political debate online and the echo chamber effect. Demos, 2016. (Cited on pages 12 and 45.)
- S. Kübler, R. McDonald, and J. Nivre. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127, 2009. (Cited on page 146.)
- G. Kumaran and J. Allan. Using names and topics for new event detection. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 121–128. Association for Computational Linguistics, 2005. (Cited on page 175.)
- V. Lavrenko and W. B. Croft. Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127. ACM, 2001. (Cited on pages 173 and 177.)
- C. Li, Y. Wang, and Q. Mei. A user-in-the-loop process for investigational search: Foreseer in trec 2013 microblog track. In *TREC*, 2013. (Cited on pages 178, 179, and 186.)

- C. Li, Y. Wang, P. Resnick, and Q. Mei. Req-rec: High recall retrieval with query pooling and interactive classification. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 163–172. ACM, 2014. (Cited on page 179.)
- Q. Li. Literature survey: domain adaptation algorithms for natural language processing. *Department of Computer Science The Graduate Center, The City University of New York*, pages 8–10, 2012. (Cited on page 17.)
- Y. Li, Z. Zhang, W. Lv, Q. Xie, Y. Lin, R. Xu, W. Xu, G. Chen, and J. Guo. Pris at trec 2011 microblog track. In *TREC*, 2011. (Cited on page 177.)
- N. Limsopatham, R. McCreadie, M. Albakour, C. Macdonald, R. L. Santos, I. Ounis, et al. University of glasgow at trec 2012: Experiments with terrier in medical records, microblog, and web tracks. Technical report, GLASGOW UNIV (UNITED KINGDOM), 2012. (Cited on page 178.)
- S. Lin, Q. Hu, J. Zhang, and S. Y. Philip. Discovering audience groups and group-specific influencers. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 559–575. Springer, 2015. (Cited on pages 10 and 231.)
- Y.-Y. Lo and J.-L. Gauvain. The limsi topic tracking system for tdt2001. *Proc. TDT*, 1, 2001. (Cited on page 174.)
- M. Lucas and D. Downey. Scaling semi-supervised naive bayes with feature marginals. In *ACL (1)*, pages 343–351, 2013. (Cited on page 94.)
- Y. Lv and C. Zhai. Positional relevance model for pseudo-relevance feedback. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 579–586. ACM, 2010. (Cited on page 175.)
- W. Magdy and T. Elsayed. Adaptive method for following dynamic topics on twitter. In *ICWSM*, 2014. (Cited on pages 20, 21, 180, 219, and 220.)
- W. Magdy and T. Elsayed. Unsupervised adaptive microblog filtering for broad dynamic topics. *Information Processing & Management*, 52(4):513–528, 2016. (Cited on pages 180 and 186.)
- J. Makkonen et al. Semantic classes in topic detection and tracking. 2009. (Cited on page 174.)
- L. Marujo, W. Ling, I. Trancoso, C. Dyer, A. W. Black, A. Gershman, D. M. de Matos, J. P. da Silva Neto, and J. G. Carbonell. Automatic keyword extraction on twitter. In *ACL (2)*, pages 637–643, 2015. (Cited on page 38.)
- S. Matsumoto, H. Takamura, and M. Okumura. Sentiment classification using word sub-sequences and dependency sub-trees. In *Advances in Knowledge Discovery and Data Mining*, pages 301–311. Springer, 2005. (Cited on pages 139, 140, and 142.)
- D. Maynard, K. Bontcheva, and D. Rout. Challenges in developing opinion mining tools for social media. *Proceedings of the@ NLP can u tag# usergeneratedcontent*, pages 15–22, 2012. (Cited on page 229.)
- D. Maynard, I. Roberts, M. A. Greenwood, D. Rout, and K. Bontcheva. A framework for real-time semantic social media analysis. *Web Semantics: Science, Services and Agents on the World Wide Web*, 44:75–88, 2017. (Cited on page 230.)

- R. Mehrotra, S. Sanner, W. Buntine, and L. Xie. Improving lda topic models for microblogs via tweet pooling and automatic labeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 889–892. ACM, 2013. (Cited on page 182.)
- P. Melville, W. Gryc, and R. Lawrence. Incorporating background knowledge into text categorization for improved sentiment analysis. Technical report, Technical Report, IBM, 2008. (Cited on page 134.)
- P. Melville, W. Gryc, and R. D. Lawrence. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1275–1284. ACM, 2009. (Cited on page 134.)
- C. Miller, F. Arcostanzo, J. Smith, A. Krasodonski-Jones, S. Wiedlitzka, R. Jamali, and J. Dale. From brussels to brexit: Islamophobia, xenophobia, racism and reports of hateful incidents on twitter. Demos, 2016. (Cited on pages 11 and 45.)
- V. Nastase, J. S. Shirabad, and M. F. Caropreso. Using dependency relations for text classification. *University of Ottawa SITE Technical Report TR-2007-12*, 13, 2007. (Cited on page 142.)
- V. Ng, S. Dasgupta, and S. Arifin. Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 611–618. Association for Computational Linguistics, 2006. (Cited on pages 139, 140, and 142.)
- J. Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*. Cite-seer, 2003. (Cited on page 146.)
- O. Owoputi, B. O'Connor, C. Dyer, K. Gimpel, and N. Schneider. Part-of-speech tagging for twitter: Word clusters and other advances. *School of Computer Science*, 2012. (Cited on pages 93 and 146.)
- B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002. (Cited on page 139.)
- K. Paramesha and K. Ravishankar. Exploiting dependency relations for sentence level sentiment classification using svm. In *Electrical, Computer and Communication Technologies (ICECCT), 2015 IEEE International Conference on*, pages 1–4. IEEE, 2015. (Cited on page 141.)
- J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM, 1998. (Cited on pages 173 and 175.)
- P. Rayson. From key words to key semantic domains. *International Journal of Corpus Linguistics*, 13(4):519–549, 2008. (Cited on page 47.)
- P. Rayson and R. Garside. Comparing corpora using frequency profiling. In *Proceedings of the workshop on Comparing Corpora*, pages 1–6. Association for Computational Linguistics, 2000. (Cited on pages 37, 38, 39, 40, and 47.)

- P. Rayson and A. Wilson. The acamrit semantic tagging system: progress report. In *Language engineering for document analysis and recognition, LEDAR, AISB96 workshop proceedings*, pages 13–20, 1996. (Cited on page 40.)
- P. Rayson, G. N. Leech, and M. Hodges. Social differentiation in the use of english vocabulary: some analyses of the conversational component of the british national corpus. *International Journal of Corpus Linguistics*, 2(1): 133–152, 1997. (Cited on page 40.)
- J. A. Rodriguez Perez, A. J. McMinn, and J. M. Jose. University of glasgow (uog_tw) at trec microblog 2012. Technical report, GLASGOW UNIV (UNITED KINGDOM), 2012. (Cited on page 178.)
- K. Sabhnani and B. Carterette. Real-time topic detection and tracking in microblog: Towards a comprehensive tweet recommendation system. (Cited on page 179.)
- H. M. Saleem, K. P. Dillon, S. Benesch, and D. Ruths. A web of hate: Tackling hateful speech in online social spaces. *arXiv preprint arXiv:1709.10159*, 2017. (Cited on pages 43 and 231.)
- N. Sanchan, A. Aker, and K. Bontcheva. Automatic summarization of online debates. *arXiv preprint arXiv:1708.04587*, 2017. (Cited on page 42.)
- S. Satapathy and H. Karnick. Using higher order dependencies for sentiment analysis. 2011. (Cited on pages 141 and 142.)
- E. Schubert, M. Weiler, and H.-P. Kriegel. Signitrend: scalable detection of emerging topics in textual streams by hashed significance thresholds. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 871–880. ACM, 2014. (Cited on page 182.)
- B. Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1467–1478. Association for Computational Linguistics, 2011. (Cited on pages iii, 3, 14, 84, 85, 86, 91, 95, 134, and 164.)
- B. Settles and X. Zhu. Behavioral factors in interactive training of text classifiers. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 563–567. Association for Computational Linguistics, 2012. (Cited on page 86.)
- C. Sievert and K. E. Shirley. Ldavis: A method for visualizing and interpreting topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces*, pages 63–70, 2014. (Cited on pages 34, 42, 50, 74, 118, 184, and 186.)
- J. Smith. One-way conversations: Examining the tactics of twitter’s most talkative parliamentary candidates. *Demos*, 2017. URL <https://www.demos.co.uk/blog/one-way-conversations/>. (Cited on page 12.)
- I. Soboroff, I. Ounis, C. Macdonald, and J. J. Lin. Overview of the trec-2012 microblog track. In *TREC*, volume 2012, page 20, 2012. (Cited on pages 47 and 177.)
- F. Song and W. B. Croft. A general language model for information retrieval. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 316–321. ACM, 1999. (Cited on page 175.)

- B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842. ACM, 2010. (Cited on page 179.)
- T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, volume 2, pages 2–6. Amherst, MA, USA, 2005. (Cited on page 177.)
- M. Timonen, T. Toivanen, M. Kasari, Y. Teng, C. Cheng, and L. He. Keyword extraction from short documents using three levels of word evaluation. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management*, pages 130–146. Springer, 2012. (Cited on page 38.)
- K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003. (Cited on page 150.)
- R. Townsend, A. Tsakalidis, Y. Zhou, B. Wang, M. Liakata, A. Zubiaga, A. I. Cristea, and R. Procter. Warwickdcs: From phrase-based to target-specific sentiment recognition. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 657–663. Association for Computational Linguistics, 2015. (Cited on page 142.)
- Z. Waseem and D. Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93, 2016. (Cited on pages 169 and 229.)
- M. Weeber, R. H. Baayen, and R. Vos. Extracting the lowest-frequency words: Pitfalls and possibilities. *Computational Linguistics*, 26(3):301–317, 2000. (Cited on page 41.)
- S. Wiberley, J. Reffin, and D. Weir. Language technology for agile social media science. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, 2013. (Cited on pages 3, 7, 13, 46, 47, 86, and 225.)
- S. Wiberley, D. Weir, and J. Reffin. Method51 for mining insight from social media datasets. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 115–119, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/C14-2025>. (Cited on pages iii, 3, 7, 17, 22, 27, 28, 29, 32, 44, 45, 69, 86, 98, 99, 100, 110, 133, 168, and 225.)
- T. Wilson, J. Wiebe, and R. Hwa. Just how mad are you? finding strong and weak opinion clauses. In *aaai*, volume 4, pages 761–769, 2004. (Cited on pages 139, 141, and 142.)
- T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354. Association for Computational Linguistics, 2005. (Cited on pages 139 and 142.)

- I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255. ACM, 1999. (Cited on page 38.)
- Y. Wu, Q. Zhang, X. Huang, and L. Wu. Phrase dependency parsing for opinion mining. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1533–1541. Association for Computational Linguistics, 2009. (Cited on page 140.)
- R. Xia and C. Zong. Exploring the use of word relation features for sentiment classification. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1336–1344. Association for Computational Linguistics, 2010. (Cited on pages 140, 141, 142, 143, and 152.)
- W. Xie, F. Zhu, J. Jiang, E.-P. Lim, and K. Wang. Topicsketch: Real-time bursty topic detection from twitter. *IEEE Transactions on Knowledge and Data Engineering*, 28(8):2216–2229, 2016. (Cited on page 181.)
- J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, pages 4–11, New York, NY, USA, 1996. ACM. ISBN 0-89791-792-8. doi: 10.1145/243199.243202. (Cited on page 172.)
- J. Yamron, I. Carp, L. Gillick, S. Lowe, and P. Van Mulbregt. Topic tracking in a news stream. In *Proceedings of DARPA Broadcast News Workshop*, pages 133–136, 1999. (Cited on page 174.)
- C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214, 2004. (Cited on page 175.)
- J. Zhang, S. Chen, Y. Liu, J. Yin, Q. Wang, W. Xu, and J. Guo. Pris at 2012 microblog track. Technical report, Beijing univ of posts and telecommunications (china), 2012. (Cited on page 177.)
- Z. Zhang and L. Luo. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *arXiv preprint arXiv:1803.03662*, 2018. (Cited on page 229.)
- W. X. Zhao, J. Jiang, J. He, Y. Song, P. Achananuparp, E.-P. Lim, and X. Li. Topical keyphrase extraction from twitter. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 379–388. Association for Computational Linguistics, 2011. (Cited on page 42.)
- S. Zhu, Z. Gao, Y. Yuan, H. Wang, and G. Chen. Pris at trec 2013 microblog track. In *TREC*, 2013. (Cited on page 177.)
- Y. Zuo, J. Wu, H. Zhang, H. Lin, F. Wang, K. Xu, and H. Xiong. Topic modeling of short texts: A pseudo-document view. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2105–2114. ACM, 2016. (Cited on page 182.)