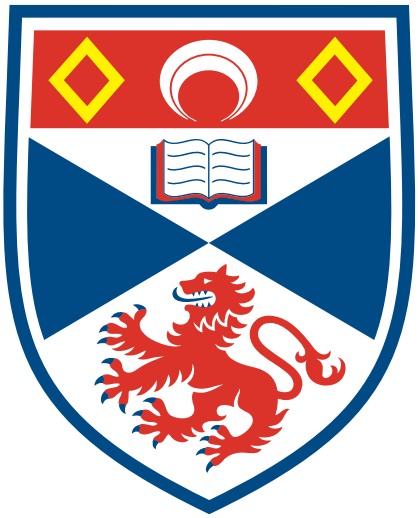
The positive, the negative and the irrelevant: real-time topic sentiment monitoring on Twitter

By

MARIJA NEDJALKOVA

Supervisors:

PROF SIMON DOBSON, DR ALEX VOSS



School of Computer Science
UNIVERSITY OF ST ANDREWS

MAY 2017

Abstract

This report discusses in detail all the stages of design, implementation and evaluation of models that perform sentiment analysis and topic modelling on publicly available social media. The sentiment analysis model attempts to define whether a tweet has a positive or negative 'mood' and expresses this 'mood' as a number. The topic model was initially supposed to detect whether a tweet talks about any environmental issues that are related to the human impact on nature in urban areas (namely the Grangemouth area). Later, however, the model was changed to detecting whether a tweet talked about the UK General Election planned for June 2017 or any other election.

This report focuses on different aspects of accessing and processing social media data, as well as using it for document classification.

The goal of this report is to evaluate the models built for the document classification, the results given by these models, as well as to discuss any possible improvements and further work in the field.

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is 18,272 words long, including project specification and plan. In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the report to be made available on the Web, for this work to be used in research within the University of St Andrews, and for any software to be released on an open source basis. I retain the copyright in this work, and ownership of any resulting intellectual property.

TABLE OF CONTENTS

List of Figures

CHAPTER 1

INTRODUCTION

Natural Language Processing (NLP) is being increasingly used by different organisations to improve the dialogue with the customers, efficiently analyse the opinion about a number of topics while using the existing social media tools, and offer additional services which would not have been possible without computational linguistics - such as speech recognition. In a way, Natural Language Processing allows the organisations using social media to have an additional sensor of the public opinion without asking people to give explicit feedback.

The huge amount of unstructured data that is being created by both humans and machines daily has been one of the major unsolved problems of Computer Science since the early days of digital computing. Natural Language Processing allows to systematise, organise and process it. One of the applications of these techniques could be to create a seamless experience for the people living in a certain council area to communicate with the estate, provide feedback and receive answers to their questions. The main advantage of such a technique is the fact that such organisations can use popular social media platforms that already exist such as Twitter and Facebook and can avoid setting up their own separate communication channel.

1.1 Falkirk Council and Grangemouth

Grangemouth is a Scottish town and a part of the Falkirk council area. Its population was estimated to be around 17000 people by the 2011 census [22]. The growth of the town has mostly been defined by its location: Grangemouth was a port and used to play an important part in trading and the transportation of imported goods using the Forth and Clyde Canal. Nowadays, a lot of the town production is defined by the oil refinery, which is commonly called Grangemouth, too, and has changed its owners several times in the last decades.

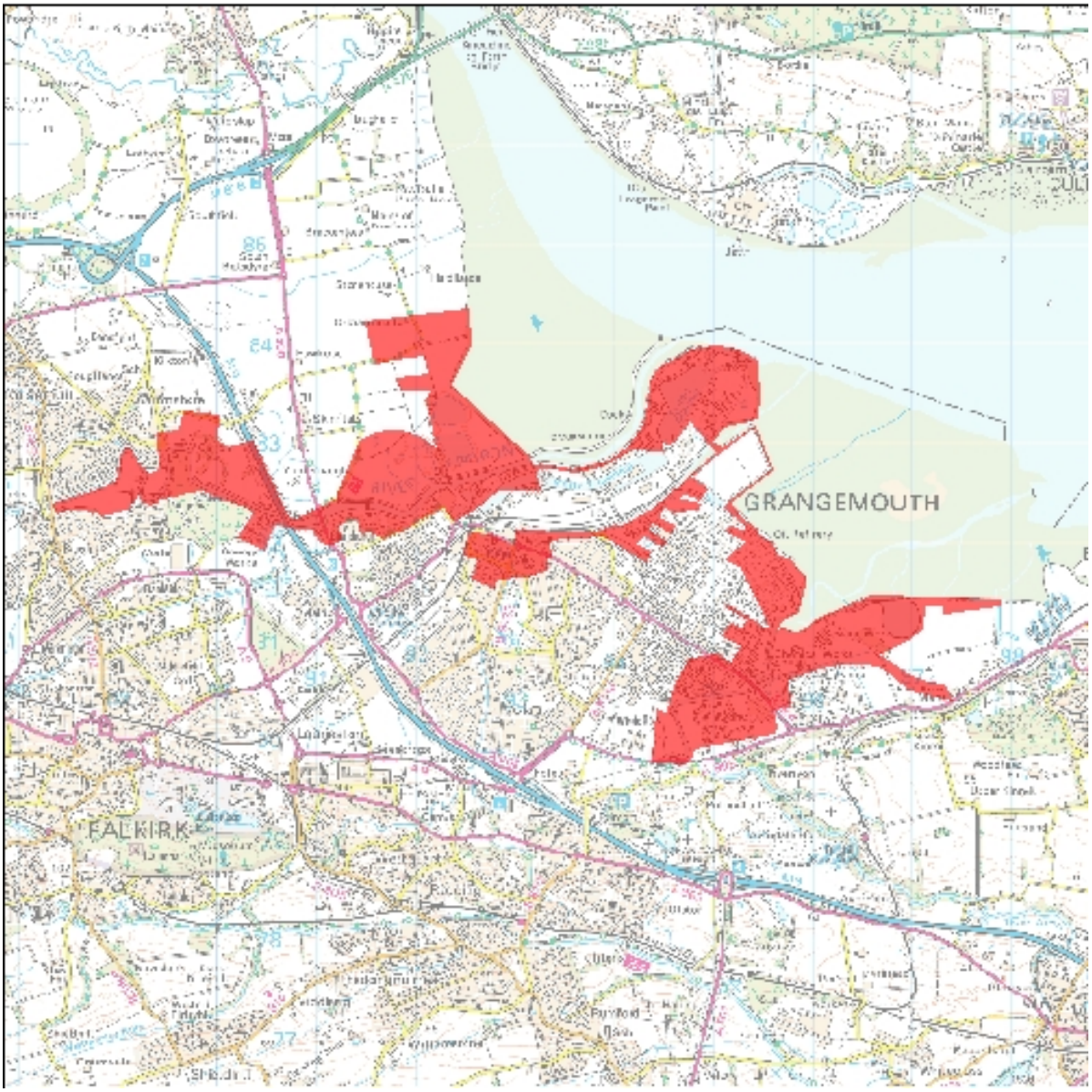


Figure 1.1: Target area that SEPA will issue a flood warning for (from http://www.sepa.org.uk/)

1.1.1 Environmental Issues in Grangemouth

The Grangemouth town is surrounded by three major infrastructure objects: the Grangemouth Port, the Avondale Environmental (commonly knows as the Avondale landfill), and the Grangemouth refinery.

The Grangemouth Port is Scotland's largest container port which is used to handling more than 9 million tonnes of cargo per year [25].

The Avondale landfill has been a source of odour problems, which could be due to Avondale's Landfill Gas Management System [23].

The Grangemouth refinery represents the largest manufacturing site belonging to INEOS. It is also home to Petroineos, the only crude oil refinery in Scotland which produces different kinds of fuels.

All of these objects affect the environmental state of the area and the lifestyle of people living there. In addition, figure 1.1 shows that some of the areas of Grangemouth could be under a risk of flood in certain cases.

The Falkirk council mentions a whole list of existing environmental issues such as the following ones [21]:

- Climate change is likely to make the occurrence of extreme flooding events more common and more severe due to increased, more intense precipitation.
- The large industrial areas in Grangemouth have high-energy consumption, with potential for on-site energy recycling.
- Odour from landfill sites within the Council area particularly at Avondale and at West Carron is a particular nuisance.

These circumstances make the environment an important topic of discussion in Grangemouth which the general public is often involved in.

1.1.2 Grangemouth in Social Media

Some public posts mentioning Grangemouth can be found among the public posts on such social media platforms as Facebook and Twitter, although the amount of these posts is relatively small due to the low number of people living in that area.

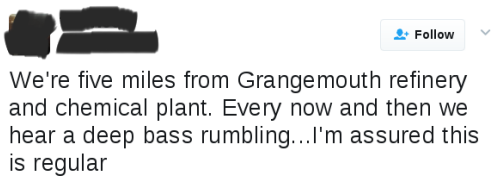
Some of these posts talk about environmental issues, too, however, it is important to note that there is no official communication channel established by any of the organisations which could be interested in the environmental issues in the Grangemouth area. So, in most of the posts regarding the environmental issues, people simply complain about different aspects, without expecting these complaints to solve anything or even to receive a reply from anyone who could contribute to solving the problem.

Some of the examples of tweets mentioning environmental issues can be seen in figures 1.3(a) - 1.3(e). Overall, several things could be noted when talking about such posts:

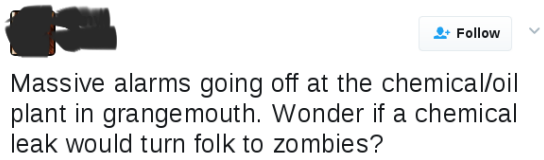
- Such posts are quite rare, one can find one or two posts every two weeks, on average
- Twitter is more popular among those who prefer to post about the environment publicly then Facebook
- Even when mentioning the environment, people use sarcasm and irony a lot, which would make it harder for a natural language processing technique to detect the correct context. In

Figure 1.2: Map of Grangemouth Surroundings

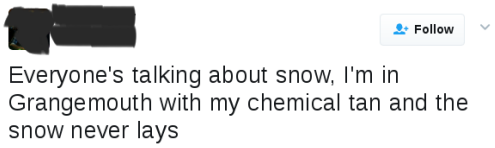
(a)



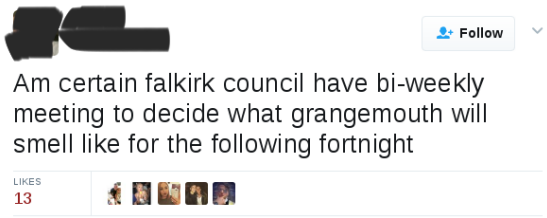
(b)



(c)



(d)



(e)

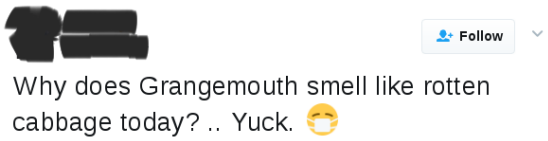


FIGURE 1.3. Examples of Tweets mentioning Grangemouth.

addition, people joke about the possible effects the Grangemouth refinery can have on their health.

- No processing of social media data has been done by any of the companies causing the environmental disturbances or the Falkirk Council, so, supposedly, not a lot of people prefer

to post about any of the problems, especially if they occur often, such as smells.

- People living in the Grangemouth area seem to be used to different environmental issues and are generally unhappy with them.

1.2 Project Objectives

This project is a collaboration with the Falkirk Council. The main goal of the project, as well as some of the details, have been defined by the Falkirk Council.

The main goal of the project is to create a tool that would act as an additional sensor by gathering and processing data from the social media and notifying the council in case of serious complaints about the environmental situation in the Grangemouth area.

The objectives of the project have been outlined initially and separated into two groups: primary objectives and secondary objectives.

The **primary objectives** are the following:

- Extract the existing data from Twitter.
- Study the existing text classification techniques, compare them and identify those that could be used for the current project.
- Train a model that recognises negative tweets about Grangemouth.

The **secondary objectives** of the project build upon the primary ones and expand the performance of the model to also recognise the topic of the tweets:

- Create a mechanism to send the selected tweets to the Falkirk Council as notifications.
- Train another model.
- Analyse and compare the models and their effectiveness.
- Create a live tool that will receive a stream of tweets and classify them 'on the go'.
- Look into existing studies of detecting irony and sarcasm in text.

1.3 Extent of completeness

All of the primary and secondary objectives have been implemented. The rest of the report will discuss in details the techniques used in this project and how every objective was met.

CHAPTER 2

CONTEXT SURVEY

The problem of computers and humans understanding each other has existed ever since the very first, non-digital, computers. Even though computers are typically much better at some tasks than humans, understanding human speech still remains an unsolved task for Computer Science.

The history of Machine Translation could be traced to philosophical sources from as early as 17th century, for example, the ideas of Descartes and Leibniz about the so-called 'universal' languages. Descartes proposed cataloguing all the possible concepts and 'elements of human imagination' [16]. Leibniz came up with an idea of the 'Characteristica universalis', a formal language which would be capable of expressing not only scientific statements but also metaphysical concepts [40].

Later in the course of history, artificial languages such as Esperanto and Interlingua appeared. Esperanto was constructed earlier, in the 1880s. Its author, Ludwik Zamenhof mentioned that his main goals for creating the language were to make it easy to learn for people of as many nationalities as possible and to "find some means of overcoming the natural indifference of mankind" [72]. Interlingua is another auxiliary language, however, it was created later, in the 1940s. Its goals differ significantly from the goals of the inventors of Esperanto. With Interlingua, the main idea behind the language was to create a naturalistic language (as opposed to Esperanto which was systematic) that would deny the naive idea that a language is simply a tool and would promote an understanding that a language represents a culture [30].

An interesting pattern could be noticed at this point in history: simultaneously with developing the first digital computers and, hence, the first programming languages, humanity was working on the so-called auxiliary languages, the main purpose of which was to help people from different parts of the world understand each other better. The first computers, thus, could be

seen as another 'nation' that had to be understood by everyone else and that had to be taught to understand humans, too. At that time, with the creation of the first digital computers and Alan Turing's publication of his famous article "Computing Machinery and Intelligence" [64], the two problems could be united into one — a meta-problem of understanding different ways of communication.

It is often considered that the history of the Natural Language Processing starts with George Artsrouni's patent (received in 1933) for his "electronic brain" which was supposed to be able to translate the input text into one of the three languages pre-programmed for the machine [34]. The machine was based on a simple automatic bilingual dictionary and used paper tape.

In a way, Artsrouni's patent defined the direction of the machine translation-related research for the next decades and resulted in the famous Georgetown-IBM experiment, the results of which were published in 1954 [17]. The experiment demonstrated the translation of approximately 60 sentences from Russian to English mainly using the lexicographical approach [35]. The system had six grammar rules and a vocabulary of around 250 lexical items. The experiment was deemed a huge success by media, and it was promised that the machine translation problem would be solved "within three to five years" [56]. However, the problem of machine translation still remains unsolved. Later analysis showed the disadvantages and limitations of using lexicographical approach [27]. The approach did not try to go into the semantics of the sentence or distinguish between different parts of speech, or count for the fact that sentences in different languages are normally constructed in different ways. Most of the sentences presented during the experiment were constructed in the same way in English and Russian. However, this is not always true, and the system would not have been able to translate sentences which could not be translated word by word into English.

Gradually, machine translation was announced unsuccessful in the famous ALPAC report [51], due to the limitations and the irregularities of the natural languages. The classic joke about the translation of "The spirit is willing but the flesh is weak" into "The vodka is good but the meat is rotten" appeared [33]. It is probably important to note that most of these jokes are merely myths, however, they demonstrate the limitations of mechanical translation without an attempt to recognise the context and, ultimately, the meaning of the text.

Despite the loss of interest in machine translation after the lack of progress and success in the area, the development of research in Natural Language Processing continued. During the 1980s, a new wave of research started, which looked into using machine learning for computational linguistics. Machine learning was not possible before that due to the limitations of processing power, however, ever since the 1980s, a big part of the research in Natural Language Processing is based on statistical supervised approach [3, 48]. The constant increase in computational power due to the Moore's Law allows processing large amounts of data and use it to construct complex language models which could be applied in a vast number of areas, such as marketing [31], medicine [15], and automated translation [7].

The biggest advantage of this approach is the fact that a large amount of data can be processed quickly. However, in order to use supervised learning algorithms, all the training data has to be labelled manually. This often proves to be a tedious, if not an impossible, task. Only large companies and research groups can afford to perform manual labelling on a scale which then allows training complex models. For example, the SNLI corpus created by the Stanford Natural Language Processing Group has 570 000 manually labelled sentences [24]. In some cases, this task can be outsourced to the general public, and platforms such as Amazon's Mechanical Turk are becoming increasingly popular [8]. However, often deep linguistic knowledge is necessary to perform the labelling correctly. At the same time, some research suggests that, when certain quality and bias control are introduced properly, crowdsourcing can have an immensely positive impact on Natural Language Processing research [58, 60].

The latest research in Natural Language Processing has turned to the semi-supervised and unsupervised methods of learning [44, 63]. This avoids having to label the training data, however, is much more difficult than supervised learning, and, therefore, the current results are generally less accurate than those given by the supervised models [41].

The current project focuses mostly on text classification in terms of sentiment and topic.

2.1 Sentiment Analysis

Sentiment analysis is an area of Natural Language Processing that has a goal of determining the attitude or an emotion expressed in a text. The common existing sentiment analysis systems try to detect how positive or negative the text is [71]. The ultimate goal would be to determine the sentiment of the given text on a much wider spectrum (such as sadness, anger, envy etc.), however, since this field is still quite young, even the problem of determining whether the text is positive or negative proves to be a complicated problem already. The simplest models look at just two classes: positive and negative (as Booleans). More complicated systems introduce the third class - neutral. Another approach expresses the sentiment as a score with clearly defined lower and higher limits, where the lowest score means the most negative text and the highest score stands for the most positive text.

The approaches used to construct sentiment analysis models can roughly be divided into three groups: knowledge-based techniques, statistical approaches, and hybrid approaches.

The knowledge-based approaches use different ways to store the existing data (called the knowledge base, hence the name of the approach) about the subject, and check for the presence of the words describing that data [12]. The data can be expressed in a variety of ways, for example, with a simple dictionary or with a general-purpose semantic knowledge base, or on a concept level [9].

The statistical approaches employ machine learning techniques to construct the models. The researcher needs to extract the correct features and give them to a classifier to learn [10].

This approach allows working with large amounts of data. Different techniques are used in this approach, such as the "bag-of-words" approach, as well as the well-known classifiers such as the Support Vector Machines and Naive-Bayes classifiers [50, 61].

The hybrid approaches aim to combine the knowledge-based and statistical approaches [28].

Overall, the state-of-the-art statistical models claim to reach accuracy around 80%, whereas the hybrid ones reach 84% [62]. This number might not seem very high to some, however, in fact, accuracy around 80% means that the model is as good at classifying text as a human. This is due to the fact that people do not always agree when manually labelling text as positive or negative. In fact, research shows that the inter-rater reliability among people labelling polarity of text varies and can be as low as 79% [5]. Hence, since a training set is assumed to represent the opinion of one person, if another person manually labelled the same set, they would probably achieve an accuracy of around 80%. Therefore, any computer system that achieves this level of accuracy is usually assumed to be as good as a human.

2.2 Topic modelling

Topic modelling is another area of Natural Language Processing that looks into discovering the topics that occur in a certain document or a collection of documents. This task is far more complex than the task of determining the polarity of a text, and the existing research is all based on statistical models. It is one more tool that allows extracting information from unstructured data and, hence, use the vast amount of data being generated by humans daily. Interestingly enough, topic modelling is not only used on text but also on images, networks and even DNAs [11, 42, 46].

The main assumption of the topic modelling is that a piece of text that, supposedly, talks about a certain topic will mention words associated with that topic frequently enough. The usual machine learning techniques are used to train different models to recognise certain topics.

The most common model used for topic modelling is the Latent Dirichlet allocation. The group of scientists that first proposed the model defined LDA as "is a three-level hierarchical Bayesian model, in which each item of a collection is modelled as a finite mixture over an underlying set of topics" [4]. Overall, the researchers claim that each document can be assigned a set of topics characterised by a distribution over words and the name Dirichlet appears in the name of the model because it is assumed that the topics are distributed with a Dirichlet prior probability.

2.3 Text classification

Classification is one of the most common tasks in machine learning. The main goal of the classification is to separate a collection of items into separate groups defined by the different features. This involves making a decision about every element of the collection of items.

There are different techniques used for classifying text documents and in each case the technique should be used depending on the goal of the research, the amount of data and its

structure. However, the most well-known techniques allowing to classify text documents are the following:

- Naive Bayes classification
- Support Vector Machine
- K-means classification
- Hierarchical classification.

2.3.1 Naive Bayes Classification

Naive Bayes is a statistical method for classification which uses the Bayes' theorem. Bayes' theorem describes a probability of an event given a prior event and has the following form:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{2.1}$$

The theorem could also be written in plain English as:

$$posterior = \frac{prior \times likelihood}{evidence}$$

The Naive Bayes Model assigns a probability to each class of the C_k, given a feature vector $x = (x_1, ..., x_n)$:

$$p(C_k \mid x_1, ..., x_n)$$

This rule could be expanded using the chain rule as the following:

$$\begin{aligned} & p(C_k, x_1, ..., x_n) \\ & \quad = p(x_1, ..., x_n, C_k) \\ & \quad = p(x_1 \mid x_2, ..., x_n, C_k)p(x_2, ..., x_n, C_k) \\ & \quad = p(x_1 \mid x_2, ..., x_n, C_k)p(x_2 \mid x_3, ..., x_n, C_k)p(x_3, ..., x_n, C_k) \\ & \quad = ... \\ & \quad = p(x_1 \mid x_2, ..., x_n, C_k)p(x_2 \mid x_3, ..., x_n, C_k)...p(x_{n-1} \mid x_n, C_k)p(x_n \mid C_k)p(C_k) \end{aligned} \tag{2.2}$$

The Naive Bayes classifier assumes that the events **A** and **B** do not depend on each other. The 'naive' feature comes from the fact that it is assumed that each feature is independent of the other features. This means that on every stage of the chain rule, as in for every feature, we are only interested in the current feature and can get rid of the rest of them, i.e.:

$$p(x_i \mid x_{i+1}, ..., x_n, C_k) = p(x_i \mid C_k)$$

Overall, the model can then be expressed as the following product:

$$p(C_k \mid x_1, ..., x_n) \propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k) \tag{2.3}$$

This assumption allows for complex models to be maintainable and scalable, but, at the same time, this is considered a rather daring assumption since, in real life, it is almost impossible to have a model where the events would be completely independent of each other. However, the classifier is considered by the researchers to perform well in text classification tasks such as sentiment analysis and spam filtering. To create a classification model, the formula above needs to be combined with a decision rule, which, in this case, is pretty simple as the model simply chooses the largest probability:

$$\hat{y} = \underset{k \in \{1,...,K\}}{\arg\max}\, p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k) \tag{2.4}$$

2.3.2 Support Vector Machine

A baseline SVM algorithm is a supervised learning algorithm that splits the data into two classes. It works by constructing a hyperplane which divides the space in which items are located in two parts [68]. The goal of the algorithm is to maximise the distance from both classes to the hyperplane. The distance to the class is the distance to the closest element of that class from the hyperplane.

The researchers talk about a hyperplane because it allows separating the groups which could not be separated by a line in two dimensions. Hence, the idea of assuming the hyper-dimensional space was accepted because it allowed separating these groups in higher dimension by a hyperplane.

A training set consists of pairs of objects, in each pair, there is a vector which describes an element of the collection, and a label which says which group this element belongs to. Each vector is a multidimensional real vector. The formula of the resulting model can then be written as:

$$\vec{w} \cdot \vec{x} - b = 0$$

where the hyperplane is defined by the vector $\vec{x}$ and the vector $\vec{w}$ defines the vector to the hyperplane.

In later studies, a nonlinear classification was introduced by replacing dot products in the formula above with a nonlinear kernel function [6].

Multiclass Support Vector Machine has been created in 1999, and its work is based on the principle that a multiclass labelling problem can be split into several binary classification problems [55]. However, later studies suggest that such complicated algorithm, which involves solving a number of binary classifications problems instead of one, could be avoided, and one optimisation problem could be solved instead [13].

The authors of the “Latent Dirichlet Allocation” paper show how the allocation model could be used to train a Support Vector Machine for document classification [4]. In this case, the LDA model is used to reduce the feature set. Often, to train a Support Vector Machine for text documents, the texts are simply split into words and almost all of the words (except the so-called stop words) are then used in a feature vector. However, when working with large diverse corpora, using all the words creates a very large model, which is sometimes not feasible. The use of the LDA for topic modelling allows reducing the feature space drastically (by 99.6% in the paper). The LDA reduced any document to a fixed set of real values, which are the posterior Dirichlet parameters. The results of the experiment showed that the LDA model could be used for fast calculations and classification.

2.4 Twitter Data Analysis

Twitter is one of the most popular social media platforms, with 310 million monthly active users and 1.3 billion accounts ever created [47, 69].

A tweet, as defined by the organisation, is an “atomic building block of all things [on] Twitter”, and could also be called a “status update” [65]. A set of actions which can be performed on an existing tweet are the following:

- tweet embedding
- reply to a tweet
- like a tweet
- unlike a tweet
- delete a tweet.

A tweet object, extracted directly from Twitter using Twitter API, has 33 fields, among which are the following:

- text (String)
- source (String)
- retweeted (Boolean)
- retweet_count (int)
- lang (String)
- id (Int64)
- favorited (Boolean)

- entities (Entities)
- coordinates (Coordinates).

The Entities object contains information about the URLs mentioned in the text, as well as hashtags and user mentions. All of these things, however, could be parsed out of the text manually.

The Coordinates object contains information about the geographical coordinates of the user who posted the tweet, as the tweet was being published. However, the coordinates field of a tweet is 'nullable', which means that the user can opt out of providing that information. In fact, data shows that only a very small portion of users provide this information [43]. The paper says that during the study which was conducted in 2013, only 2.02% of all tweets included the location data. The researchers have, however, since found ways to locate tweets even if the location data is not included as metadata. A lot of different techniques and heuristics are being used to achieve that goal [32, 43].

The existence of the Twitter API which allows requesting data for analysis makes Twitter attractive for research in different areas, especially Natural Language Processing [65].

At the same time, analysing Twitter text has its own specifics which are defined by the philosophy of the platform. The main rule of Twitter is that posts are no longer than 140 characters, and it defines the style of the posts in a major way. In a way, it even defines the styles in different ways for people speaking different languages, since the linguistic research shows that the amount of information which can fit into 140 characters varies a lot between the different languages. For example, when writing in Chinese, one can express almost twice as much as an English speaker [52].

The microblog format of the platform means that people have to express their thoughts and opinions in a very concise manner. This makes the users try to insert as much meaning — in terms of both information and emotion — into one sentence. This means that often the tweets will have a certain aphoristic meaning to them, will try to have a whipping effect. Sharp language, slang, performativity, poetry, scepticism, weird abbreviations, special user inventions like mentions and re-tweets, emojis — all of these are the most common tools for expression on Twitter.

Depending on the goal of the research, it might be necessary to deal with some or all of these specifics. The deepest sentiment analysis will take all of them into account. Most of the modern tools focus on emojis to extract the sentiment of a tweet because they themselves often hold a certain amount of sentiment, which is relatively easy to extract [53]. However, dealing with some of the other tools such as slang, abbreviations and mentions is still necessary during the preprocessing of a tweet.

If one was to summarise all the techniques used to reprocess Twitter data, the general procedure would look similar to the following one:

1. Extract the valuable parts of a tweet. These can be the text, the coordinates, information about the poster and whether the tweet has been favourited (if yes, how many times).

2. The tweet is then stripped of all the things that do not constitute natural language. These are the mentions, the URLs, the hashtags and the emojis. These entities do, in fact, hold important information and can then be processed. For example, a study has shown that tweets containing URLs tend to be positive [29]. Other tools have polarity extraction mechanisms for emojis [53].

3. After the tweet has been 'cleaned out', a text analysis can begin. First, the researchers have to deal with abbreviations and slang. It can be useful to extract noun phrases, for which part-of-speech tagging may be necessary. This is quite a complicated procedure, and some researchers choose to work with unigrams or bigrams instead [49], however, it has been proven that noun phrases provide more information and work better than unigrams or bigrams [70]. Studies also show that unigrams outperform bigrams when doing sentiment analysis [26].

4. Finally, all the extracted data has to be put into a feature vector which can then be used in whatever machine learning technique is chosen for the study.

Even though this process represents preprocessing as performed in the most recent research, anyone performing Natural Language Processing on Twitter data should keep track of the latest changes in the Twitter policies and social trends in media. For example, Twitter started officially supporting hashtags in 2007, a year after the company was established [66]. The geolocation service was first introduced in 2009 [67]. Twitter continues to evolve, and it means that new features might be introduced later, and these features might affect the way the users express their emotions and thoughts.

CHAPTER 3

REQUIREMENTS

The initial requirements for the project were provided by the Falkirk Council. The organisation wanted to have another 'sensor' and be able to keep track of the public opinion about the environmental issues in the Grangemouth area and be able to respond to the complaints before the problems become large.

The organisation wanted to have a fully automated service which would work with all the most popular social media platforms and did not involve any man-hours. It was planned that the classifier would be running on a separate server and would send notifications about any posts of interest to the Falkirk Council.

The organisation provided a list of keywords they were interested in keeping track of. It can be found attached in appendix B.

Due to the fact that the organisation was represented by a person who was not familiar with the possibilities of Natural Language Processing and Machine Learning and did not have the resources to become more familiar with this field, detailed requirements were produced without the guidance of the organisation.

Overall, the requirements of the project separated the work on the project into separate stages:

1. Get access to the necessary social media APIs and stream data about Grangemouth.
2. Train a sentiment analysis model which would extract any negative posts from the stream.
3. Train a topic modelling model which would extract any posts talking about the environment.
4. Get posts that satisfy both requirements (are negative and talk about the environment) and send them as push notifications to the Falkirk Council.

CHAPTER 4

SOFTWARE ENGINEERING

The software engineering aspect of this project depended mostly on the amount of communication with the client. For this project, the role of the client belonged to the Falkirk Council.

Generally, the project followed an iterative approach, and the whole workflow could be separated into several iterations. The rest of this chapter will describe the iterations involved in the software engineering side of the project.

4.1 Iteration 1

The communication with the client was not very systematic due to a variety of reasons, such as the project being experimental and, thus, not having a very high priority. As a result, the project was mostly an individual research project. However, communication with the Falkirk council gave some positive results. Initially, the communication with the Falkirk Council was done through email, where the main idea and the main requirements were discussed.

The main requirements were to build a model which would access social media data and notify the Council about the possible problems around the Grangemouth refinery area. The client also mentioned dangers of flooding and chemical leaks. As a result, a lot of initial research was spent on reading about the ethical concerns with using social media data in crisis situations as well as about the techniques used to locate an environmental disaster. Also, at that point, it was clear that a sentiment analysis model had to be created. This gave enough information to start the project.

In the first phase, it was revealed that the privacy settings provided by Facebook API do not allow accessing any posts, not even the public ones. This meant that the project had to evolve around using the Twitter data, which is available publicly through the Twitter API. The survey

of the existing techniques used for sentiment analysis of Twitter showed that mostly large data sets are used for training such models, and some of the datasets created by different research groups such as Stanford NLP Group are available for general use.

During the first iteration of the project, the sentiment analysis model was created. It might have been helpful to create a joint model that would perform both the sentiment analysis and the topic modelling at the same time, however, that model would require a large amount of training data which was not available at that point, and the availability of the sentiment analysis training set made it convenient to work on the models separately.

By the end of the first iteration, the sentimental analysis model had been created, tested and evaluated.

4.2 Iteration 2

In the second iteration, the topic model had to be created. The meeting with a company representative showed that the project would have to change the requirements slightly. Instead of detecting upcoming disasters such as flooding and chemical leaks, it became clear that the main goal of the application would be to detect small occurrences of complaints in order to be able to work on the problems proactively. This changed the cases we were looking for and, hence, the whole approach to topic modelling.

Falkirk Council provided a list of keywords which they wanted to trace. The list of keywords can be found attached in the appendix B. However, the very first API request showed that it was impossible to get any training data at all. Grangemouth area is fairly small in terms of population, which affects the number of posts on Twitter about it. Even though some posts could be found through the website, it would still not be enough to train a successful model.

Due to these reasons, it was decided that the project had to change direction. So, for the topic modelling phase of the project, another topic was chosen. This time, it was decided to focus on elections, which would give a wider dataset. In the second iteration, the topic model was created, tested and validated.

Both models were then combined to produce a filter for live tweets and to determine whether a tweet is a negative tweet that also talks about the 2017 UK General Election.

CHAPTER 5

ETHICS

The project deals with social media data which is publicly available and can be requested using the Twitter API. Research that uses public social media posts is becoming increasingly popular due to the availability of the data and the ease with which it can be obtained. For Natural Language Processing, where a lot of user-created data is necessary to train different models, the Twitter platform is proving to be especially useful.

Taking into account that using social media platforms for research is a relatively new phenomenon, there are no clear rules yet defined for using this data, especially when it comes to the ethical reasoning.

The main problem surrounding the usage of social network data is the fact that it is not clearly defined what constitutes private and public data. From the point of view of the availability of the data, the user defines what posts are public and what data is private using the settings usually available for them. This does not, however, define the cases in which the data could be used by third parties, such as advertisement companies and researchers. To partially solve this problem, some social media providers notify the users in the terms and conditions that their data could be used by third parties. Despite the fact that this mostly solves the problem from the legal point of view, the ethical issue remains.

Traditionally, when an experimental study is conducted, all the participants are volunteers and all of them have to give consent for researchers to use their data in the study. They get explained the goals of the study, what they need to do and can ask any questions they have. Even though these rules impose certain constraints on some of the research (for example, scenarios where the scientist needs the participant not to know what is going to happen), they ensure that the study in conducted in an ethical manner. Another key concern for the researchers is the participants' anonymity. Traditionally, all the data used in research is anonymised.

When the point of the study is to gather as much data as possible (from thousands of participants, probably), it is very hard to gather all of these volunteers' consent to use their data from each one of them efficiently. When all of the necessary data is available online, and the users have given their consent for the data to be used by third parties by agreeing to the Terms and Conditions of the platform they are using, it can seem that the problem is solved and that one only needs to write a short script to obtain the data and use it as one pleases. Additionally, it is harder to anonymise the social media data due to the fact that most of the social media platforms allow for the searching of the public posts via keywords. So, for example, an author of a tweet could easily be identified using the keyword search (assuming, of course, that the tweet is not a re-tweet of a very popular tweet). The potential breach of anonymity increases a risk of harm of the participants, and it is impossible to know or predict which data is safe to use to avoid doing harm. Some studies suggest that when working with Big Data, the assumption that no other harm can be done to posts which have already been published is invalid [37].

When talking about environmental issues, research often looks into crisis data. Reuters reported on the record number of tweets after the Sandy hurricane struck in 2012 [59]. In addition, a number of studies have been based on the social media data received after different disasters such as earthquakes [54] and other natural of man-made disasters [19]. However, a discussion has started about the fact that despite the fact that the data is available and the purpose of such studies is to help the humanity recover from the disasters faster, the use of this data might not be ethical [14].

At this point, a lot of Big Data research is backed by different committees which make sure that the data is used ethically. The decision is normally based on several rules and depends on how the data is used, whether the data is sensitive, whether the organisation is allowed to collect the data and so on. For example, Glasgow University provides an ethics guide for researchers who want to work with Big Data. The guide was created at the University of Aberdeen and gives an ethics framework for those working with social media data. The framework asks the researchers questions such as: "Can the social media user reasonably expect to be observed by strangers?", "Are the research participants vulnerable? (i.e. children or vulnerable adults)", and "Is the subject matter sensitive?" [18]. This project, too, works with social media data. Judging by this framework, the work done in this project is safe for the Twitter users. Twitter API allows data collection and users agree for their data to be used by agreeing to the Terms and Conditions. All the data for this project is anonymised and will not be published. In addition, despite the fact that the environment could be considered a sensitive topic, the existing data on Twitter does not mention any crisis, only routine environmental issues, so the project is not working with crisis data. In addition, ethical self-evaluation, which is a mandatory part of writing a project at the University of St Andrews, also revealed no ethical issues.

CHAPTER 6

DESIGN & IMPLEMENTATION

This chapter will discuss in detail the design decisions made in this project, and any specifics of the implementation phase. The whole process of the application could be divided into several stages and described using figure 6.1. The tweets are taken from Twitter using the Twitter API, after which they are passed through two models: a sentiment analysis model and a topic modelling model. Two questions are asked: "Is a tweet positive or negative?" and "Does the tweet talk about a certain topic?". If at least one of the answers is "No" then the tweet is discarded. Otherwise, it is marked as being relevant.

The project has been written entirely in Python because most of the existing Natural Language Processing tools, especially the NLTK library, are implemented in Python.

The rest of the chapter will discuss in detail both of the models and the design decisions behind them.

6.1 Existing Tools

The main constraint of this project has been the time limitation. A lot of small steps involved in the processing of data could have been separate study fields, and an immense amount of research could have potentially been devoted to these fields. Because this project is limited in time, only the existing libraries could be used for some techniques. Some of such methods are imperfect and could be improved. This will be discussed in the conclusions chapter in greater detail, but in this chapter, it will be mentioned a lot that certain trade-offs have been made to make the project possible. In most cases, existing libraries have been at least tried, mostly because it is complicated and time-consuming to implement techniques related to Natural Language Processing from scratch, and it also often requires large data sets.

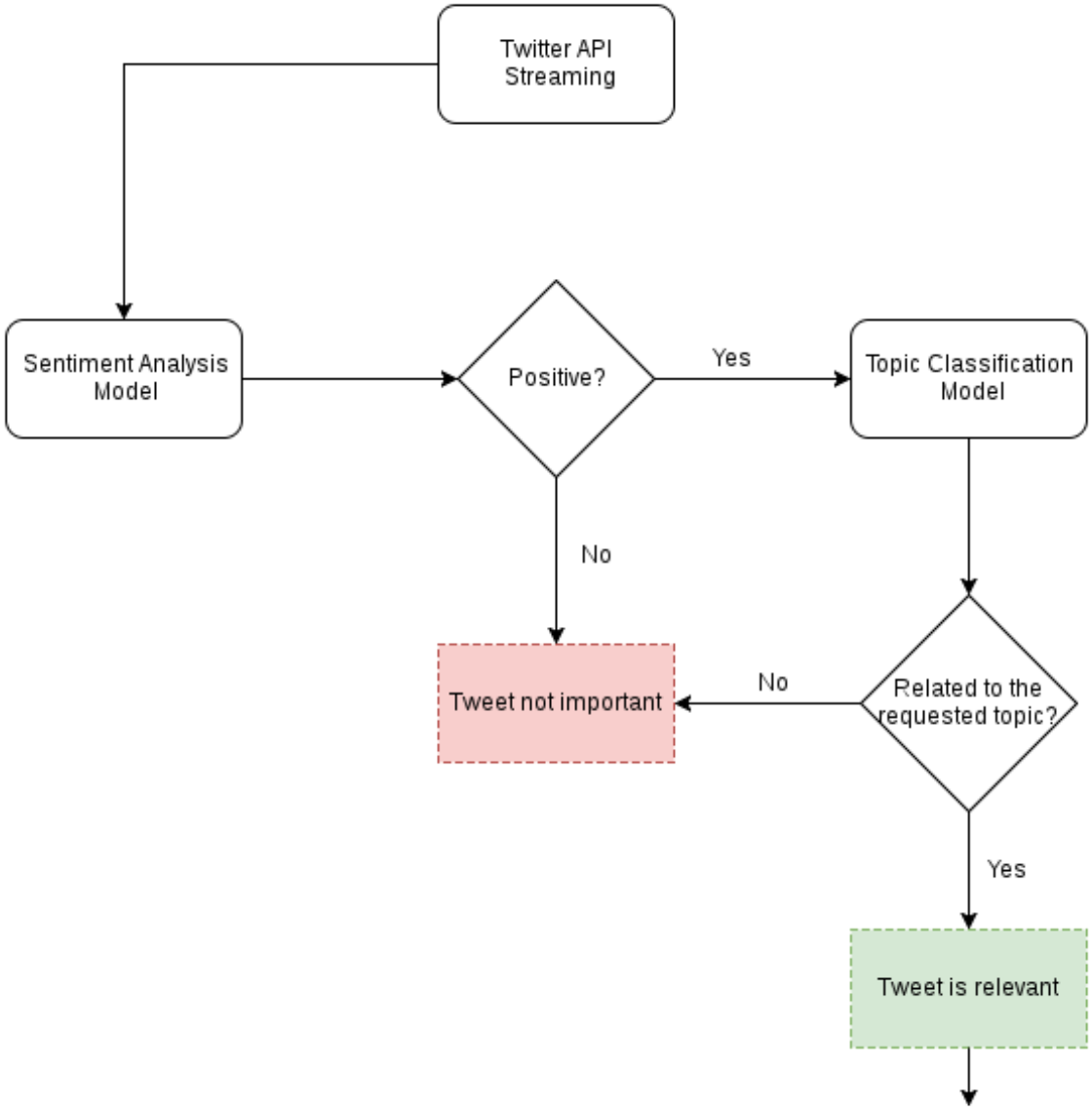


Figure 6.1: General Workflow

Twitter API

For the first stage of the project, tweets were obtained from Twitter using Twitter API. The keyword used for the search was "Grangemouth". Ideally, it would be better to locate tweets geographically and select the ones coming from the Grangemouth area. This would also allow including tweets that do not mention the area but talk about it. However, as was discussed in the Context Survey chapter, it is quite complicated to extract the geographical data out of the tweets. Most of them do not contain coordinates, and the heuristics which are widely used in research are neither ethical nor accurate. This is why it was decided to focus only on the tweets found through the search of the "Grangemouth" keyword. Streaming of live tweets was also implemented using the same API.

Natural Language Processing Libraries

NLTK[1], the Natural Language Toolkit, is the most well-known platform for Natural Language Processing. It provides access to a variety of methods used to process text and has interfaces which help access some of the largest corpora such as WordNet.

TextBlob[2] is a Python library for processing textual data. Although it is often seen as an alternative to NLTK, in come cases it builds on the functionality provided by the NLTK.

Both of these libraries have been used extensively for different purposes in the project such as noun phrase extraction, text tokenization, stemming, part-of-speech tagging, parsing, document classification and other.

Another library used is the preprocessor[3]. It is a library for tweet preprocessing in Python and helps extract Twitter-specific entities such as emojis, URLs, and hashtags.

6.2 Sentiment Analysis

6.2.1 Training datasets

For sentiment analysis, a large dataset is required to train the statistical model. Creating such a dataset is a hard and time-consuming task not only because one would have to go through a large amount of data but also because people do not always agree on the polarity of a sentence. Hence, most of the researchers use several well-known datasets with tagged data.

In addition, performing sentiment analysis on Twitter data is a specific task, and a model trained on, say, Shakespeare texts could not be used on Twitter data. So, the dataset had to consist of tweets.

Probably, the largest dataset available publicly and referenced on the The World Wide Web Consortium website [4] is the Thinknook dataset. It consists of 1,578,627 classified tweets, where

[1] http://www.nltk.org/

[2] https://textblob.readthedocs.io/en/dev/

[3] https://github.com/s/preprocessor

[4] https://www.w3.org/community/sentiment/wiki/Datasets

each tweet is tagged by 0 if it is negative or 1 if it is positive. The advantages of the dataset are, apart from its size, the fact that the data is shuffled within the dataset, so the researcher would not have to put any effort into making sure that when training or testing data is selected from the dataset, approximately half of the tweets are positive and half are negative. However, at the same time, a closer look and the first tests showed that, in fact, the dataset consists of the tweets taken from two different sources: Sentiment140 dataset and Kaggle dataset, and the Kaggle dataset only contains tweets about movie reviews, which would make the model biased and not work properly on data related to life in a Scottish town or environmental issues. So, it was decided to only focus on the Sentiment140 dataset.

Sentiment140 dataset [5] is one of the most well-known Twitter sentiment analysis datasets and was created by several Stanford University graduates. In the dataset, which is available for access, tweets have several fields and are tagged as positive (4) or negative (0). This dataset, however, is not perfect. For example, the creators say that the data was not labelled manually but automatically. This is usually fine, except, in this case, the authors say that the criteria for automatic tagging have been the smileys: "we assume that any tweet with positive emoticons, like :), were positive, and tweets with negative emoticons, like :(, were negative" [2]. There are several big issues with this approach. First, a smiley can be deceiving or used ironically, as a sarcasm. Secondly, Twitter currently supports emojis, which are complicated Unicode characters, so the heuristics chosen by the researchers would not be efficient. Also, even though roughly half of the tweets in the dataset are positive and half are negative, the tweets are not shuffled, which means that they had to be shuffled initially and extra care had to be taken to ensure that the training and testing sets had roughly equally spread data in them.

In addition, the same authors also created an online tool that accepts a tweet text request and returns a response with the polarity of the text. Presumably, this tool also uses a model created from the same dataset, so it would make sense to, when validating a newly created model, compare its responses to the ones provided by that model. However, this is impossible because, unlike the dataset, which uses only two possible labels: "positive" and "negative", the online model adds a third label — "neutral". This makes it impossible to compare the two models.

Unfortunately, the rest of the datasets available would either not be large enough, or not focus specifically on Twitter data, or have other flaws, so the Sentiment140 dataset seemed to be the best one of all the available ones.

When the main objective is to work with human-created data, it is not easy to generalise the behaviour of different language models. A model might work well on some data but perform much worse on another data set. That is why throughout this project, the evaluation of the model consisted not only of the statistical evaluation based on testing data but also of manual evaluation based on the 'live' data coming directly from Twitter. When creating the model, the main ideas have been based on the results of the most recent research, which has been discussed

[5] `http://help.sentiment140.com/home`

in the Context Survey chapter, however, in some cases, certain changes have been made due to, for example, better performance of such modified models. Overall, the workflow of the model creation has been following an Expectation-Maximisation process: a model was created, tested, evaluated, and then the loop would start again, with modifications.

6.2.2 Feature Vector Extraction

In order to train a classifier, the input data has to be represented as a feature vector. The success of the entire model depends on the structure of the vector, and it is up to the researcher to design it. In Python, a feature vector is usually a dictionary where the keys are the names of the properties and the values are usually numeric or Boolean values of those properties. For example, it could look like this:
`{'Length':139, 'hasHashtags':True, 'containsNegativeWords':False}`.
Alternatively, a typical unigram model will look somewhat like this:
`{'I':True, 'am':True, 'happy':True, 'today':True}`.
Note that while in the first case, the vector will always be of the constant size, the latter structure will have a variable number of key-value pairs. Also, since the approaches create a dictionary, which is an unsorted data structure in Python, the order of the words appearing in the input does not matter. Such an approach is called the "bag-of-words" approach and is one of the basic feature extraction techniques in Natural Language Processing.

A lot of models created for research working with the natural language data use the unigram model [1]. It performs well when the researcher initially does not have a lot of knowledge about the training data, or when it is too vast (like anything concerning natural language) to be summarised in several fields. Also, there is not enough research looking into how different structures of the feature vector affect the performance of the sentiment analysis model and how to best design a feature vector for an input consisting of natural language. It would be interesting to look at this problem, but it is a case for future studies and does not lie within the scope of this project. In this project, the unigram model is the basis model of the feature vector.

Tokenization

The first step of feature extraction is tokenization. Tokens are 'units' of text that the input string is being split into; they are most often words but could also contain URLs, emojis, hashtags, or other Twitter-specific data. The initial simple manual tokenization showed that simply tokenizing text using NLTK provided default methods would not work particularly well. Figure 6.2 shows the pairs of tuples, where the first item is the token, and the second item is a number obtained by adding 1 every time it appeared in a positive tweet or subtracting 1 every time the token appeared in a negative tweet. The figure shows the first and the last twenty items of an entire list. Note that the results are not normalised, since it was a small bit of the initial exploratory analysis, and the goal was simply to see what the data looks like, in general. As it can be seen in

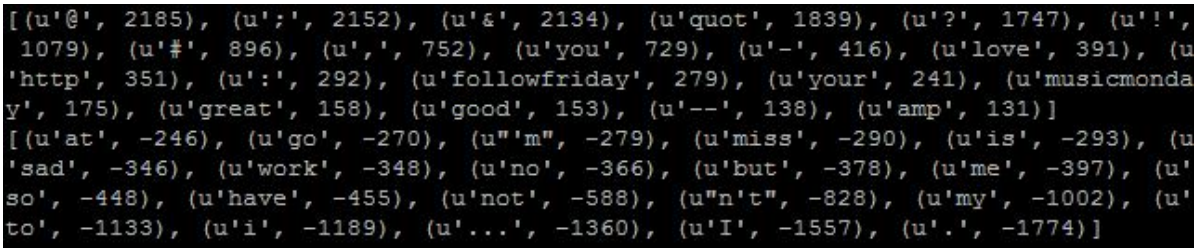


Figure 6.2: Tokens that appear in most positive and most negative tweets, default tokeniser

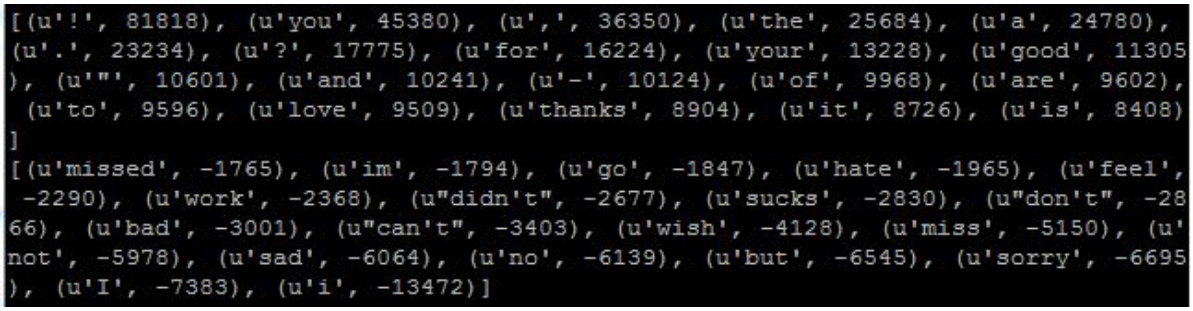


Figure 6.3: Tokens that appear in most positive and most negative tweets, TweetTokenizer

figure 6.2, most of the most popular tokens are not words but punctuation or the '@' character which most often refers to a mention on Twitter. At the same time, this initial simple analysis showed some really important facts about the dataset which would allow making an educated guess about the data. For example, we can say that the word "love" is predominantly used in positive tweets, together with words "great" and "you". In addition, the '#' sign is also in the first half of the image, which could mean that, overall, hashtags are more popular in the positive tweets.

Predictably enough, the list of token appearing mostly in negative tweets mentions the words "work", "sad" and "miss". It could be surprising for some (or explained in quite a philosophical manner) that this list also contains the pronoun "I" in three different forms.

Thankfully, NLTK also provides a `TweetTokenizer` which, unlike the default tokenizer, would, for example, see "@marianna" as one token and not two different ones. Figure 6.3 shows the result of the same analysis as in 6.2, but using `TweetTokenizer`. It can be seen that the list has changed significantly, even though "I" and "i" are still the tokens that are mostly seen in negative tweets (at this point of the initial analysis, the tokenizer was case-sensitive).

Pre-processing

Due to the nature of natural language, before any features can be extracted from the input data, it has to be processed. Human-produced data is non-structured, and using the vast amount of unstructured data has been one of the largest problems for data science. Pre-processing data can help make it more structured for later stages of feature extraction.

Pre-processing natural language input usually involves dealing with such problems as difference between upper or lower cases, usage of stop words, abbreviations, any other structures. In the case of this project, it has to be remembered that Twitter has certain structures not normally used in other texts, such as URLs, mentions, hashtags. For the ease of work with different words rather than individual symbols, the pre-processing can be performed immediately after tokenization.

For this project, `ngram_classifier.NGramClassifier.preprocessing_extractor()` does all the preprocessing.

First, the tweet often has to be decoded. One of the hardest tasks of this project was keeping the data consistent. The data was coming from different sources such as text files of different formats, as well as the live Twitter feed. Some of the input was represented with strings, whereas in other cases it would be in Unicode. The text had to be encoded, and the potential Unicode decode and encode errors had to be accounted for. In some cases, the tweet would not be encoded successfully because it would contain characters that do not exist in English language and could not possibly be translated to ASCII. In cases where no sense could be made out of a tweet, the tweet would have to be skipped.

The `preprocessor` library was used to extract URLs, mentions, emojis and hashtags from a tweet. For **URLs**, if there were any in a tweet, they would be each summarised to a single dictionary entry with a 'True' value (`{''$URL$'': True}`. Most tweets contain at most one external link, so it made sense for the URLs to be summarised to a Boolean value. Deeper research might be able to detect how different links affect the polarity of a tweet. This could potentially be done by accessing the data from the link and trying to determine the polarity of that data, or, alternatively, maybe a certain amount of polarity could have been extracted from the domain name itself. However, this is also a case for a further study and is not in the scope of this project.

With the **mentions**, the situation is not so clear, so the approach taken is just one of a number of possibilities. Similarly to the links, all the mentions are summarised into a dictionary entry of the form `{''$MENTION$'': True}`). Alternatively, it was possible to record the number of mentions used in every tweet. However, that number would still be quite low and probably wouldn't affect the resulting model much. Some might argue that it would be useful to use the mention as a source of information about polarity since a mention is a '@' character followed by the username of a user being mentioned, however, it would not be reasonable to assume that each Twitter user is either happy or negative the whole time, even though some people are generally more positive than the others. Overall, this is yet another field of study and is also not in the scope of this project.

Unlike the URLs and the mentions, **hashtags** ('#' followed by a word or a combination of words that are not separated by any form of whitespace) actually contain text information which could

be interpreted in several ways. One of the approaches would be to try and parse the hashtags into the actual text. This would involve quite complicated parsing. Let's look at a sample hashtag, '#teamGBolympic'. When parsing this hashtag, the desired output would have been the following list:
`[‘‘team’’, ‘‘GB’’, ‘‘olympic’’]`.
After that, another bit of processing would have to determine that "GB" stands for "Great Britain". The result of such parsing would provide data which could be added to the other words and interpreted as additional input for the model.

On the other hand, it is possible to argue that hashtags are not simply combinations of words. They usually define a concept in a very concise way and are used by a large number of people, becoming trends and being passed between people. If the concept was not trendy and Twitter did not have support for hashtags, the same exact concept could have been expressed in a large number of ways, like almost all concepts in natural language. Yet this specific choice of words and, sometimes, characters, makes hashtags more than words. For that reason, it seemed more sensible to record the hashtag as it is instead of trying to parse it into a list of words.

Finally, the **emojis** were processed separately. Unlike emoticons, which are simple combinations of text characters, emojis are 12-by-12 pixel images. When received from Twitter as a part of the text, they are represented by a Unicode sequence of characters. Every emoji has its own unique Unicode codepoint, but the emojis are not sorted in any way which would allow extracting any meaningful information from the code alone. At the same time, each emoji clearly carries a lot of information about sentiment.

Fortunately, some research has been done to detect the polarity of each emoji. An online ranking exists where each emoji has been given a sentiment score from -1 to 1. The "Sentiment of Emojis" paper describes the construction and analysis of the ranking [39]. Unfortunately, the ranking only exists as a web page and has no API, so in order to use it, it was necessary to scrape the page. The `BeautifulSoup` library allows parsing the HTML received from a URL and extracting information from the DOM of the page. So, the information from the web page has been scraped and saved locally as a CSV file which contains a list of a format `"0x1f44f": 0.52`, where the key is the unique codepoint of the emoji and the value is its sentiment score. This allows avoiding scanning the page every time the application is used. If at any point the file cannot be found, the page will be scraped again automatically, otherwise, the local file will be used.

When an emoji is encountered in the tweet, it is looked up in the local file and its sentiment score is extracted. If the emoji is not found (the 'vocabulary' of emojis keeps developing), it is skipped. The sentiment scores of all emojis are summed up into one number and added to the feature vector in the following format: `"$EMOJI$":0.42`.

After all the pre-processing all the URLs, emojis, hashtags and mentions are cut out of the text. The rest of the string is tokenized using `TweetTokenizer`. All of the tokens are converted to **lower case**, apart from the cases where the entire word is in upper case. This is just one approach and it could have been done in another way, of course. Usually, it is useful to have all the text in one case (say, lowercase), since it, for example, makes "Today" (first word in the sentence) be equal to "today" (a word in the middle of a sentence) and ultimately improves the statistical model. However, when the data is as informal as it can be on Twitter, it might sometimes be useful to save the case of the words. For example, if the whole sentence is in uppercase, it might mean that the sentence expresses strong emotions, and it can help detect the polarity of the sentence.

Finally, the resulting list of tokens is stripped off the so-called **stop words** and the punctuation. The punctuation, however, does not seem to affect the performance of the model at all. Stop words are the non-significant words that, for example, would not be used in a Google query. They act as a glue for the rest of the words in the language. For example, nouns and most adjectives would not be considered stop words, but articles and conjunctions would be classified as stop words. In a lot of applications related to Natural Language Processing, including sentiment analysis, removing stop words is considered a good practice [38, 57]. However, experiments showed that the model where the stop words are stripped performs slightly worse compared to the model where the stop words remained. Research suggests that in some cases, certain stop words are useful for defining sentiment of a text [45].

Noun Phrase Extraction

Further inquiry into the results of the existing sentiment analysis research opened another topic for discussion. The initial versions of the classifier were working with individual words. However, in natural language, meaning (and, thus, polarity) of individual words often depends on the context. Because of this nature of the natural language, it would be more beneficial to work with phrases rather than individual words.

Phrase extraction is, however, quite a complicated task since the term "phrase" could not be defined formally. In natural language, phrases take different forms, have different lengths and structures. Noun phrase extraction is an important task in Natural Language Processing. The NLTK book [6] describes a way of noun phrase chunking based on a context-free grammar. The context-free grammar can be defined in a number of ways, and the authors of the book define quite a simple one in the example [7]: an optional determiner, followed by zero or more adjectives, followed by a noun.

The grammar looks as follows:

```
grammar = "NP: <DT>?<JJ>*<NN>".
```

[6] `http://www.nltk.org/book/ch07.html`

[7] `http://www.nltk.org/book/pylisting/code_chunkex.py`

Writing a context-free grammar that would cover all of the different forms of noun phrases is a complicated task. In addition, in order to use such a grammar, it is necessary to know the parts of speech of every word. Even though NLTK provides access to large corpora of text, including online chat texts, it is unreasonable to expect every word mentioned on Twitter to be known to NLTK. It would require manually tagging a large amount of text, including abbreviations and slang. In addition, it would require performing error correction on the data because it is unreasonable to expect that the tweets contain no typos or grammatical errors. Overall, it seemed that implementing a noun phrase extractor from scratch could not possibly fit into the scope of the current project.

The TextBlob library provides two methods allowing automatic noun phrase extraction: `Fast_NP_extractor` and `ConllExtractor`. The second extractor "uses the CoNLL 2000 corpus to train a tagger" [8]. However, neither of the extractors performed on a good level. Both would skip a lot of important noun phrases, find at most one noun phrase in the given tweets, and take around three seconds to extract noun phrases from one tweet. For example, from "what a lovely day it is right now, going for a walk. I hope to see the sun" only "lovely day" and "what lovely day" were being extracted by each of them correspondingly. Even if we assume that the extractor only accepts a sentence at once, both of the extractors still skipped the "a walk" phrase.

In a situation where a model has to be trained on tens of thousands of tweets, such poor performance would not be acceptable, and, unfortunately, it was impossible to use noun phrase extraction successfully as a part of this project.

N-Grams

The existing research and different tests show that using n-grams might give as good results as noun phrase extraction, with much less computation. In fact, some papers on Twitter sentiment analysis mention that unigram model outperforms the bigram or trigram model [1]. Yet, because it seemed counter-intuitive, bigram and trigram models were also tested.

Using NLTK library, it is possible to see the "most definitive features" of the model once it is trained. The most definitive bigrams are shown in figure 6.4. Each row in this figure shows a feature and how it affects the result given by the classifier. For example, the first row shows that, when the input text contains the combination of tokens "100" and "followers", then it makes the tweet positive ("1:0" shows which label of the two was more likely to be chosen) with the probability 57.2:1.0, which is quite a large proportion. The items in the list are sorted by this proportion, so 57.2 is the largest proportion in the entire sample set.

The bigram model would, indeed, show slightly lower accuracy than the initial model, where the classifier would use default methods for obtaining unigrams, even though it would still be above 70% for most of the parameters. However, it was much faster and could be trained on an

[8] `http://textblob.readthedocs.io/en/dev/advanced_usage.html`

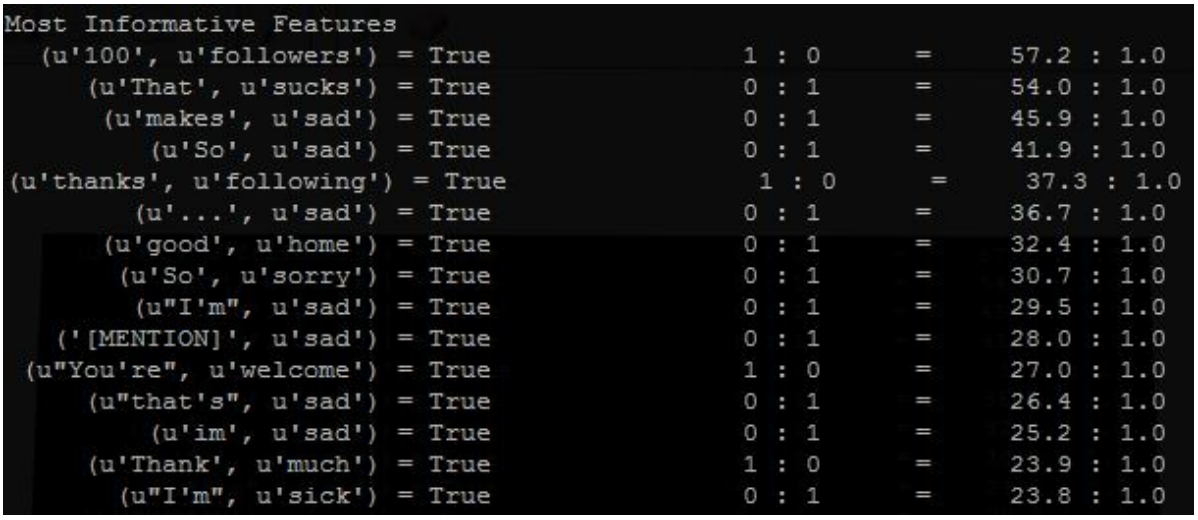


Figure 6.4: Most definitive bigrams, using Naive Bayes model and TweetTokenizer

amount of data that was several times higher than the data that could feasibly be used by the initial model.

With trigrams, the accuracy dropped to around 60%, proving the reference made in the paper of Agarwal et. al. that trigrams perform worse than bigrams and unigrams [1]. The model would still be able to train on large amounts of data in a matter of seconds. With Twitter, it could probably be argued that trigrams are too long as constructs to act as meaningful phrases. In a text full of special symbols, mentions, abbreviations, slang and links, and in a text that is also less than 140 characters long, three words will probably have quite a lot of content squeezed into them. Another issue is that removing the so-called “stop words” is considered a good technique in sentiment analysis. Stop words are usually short words that do not carry a lot of meaning. When stripped of these words, the text loses its fluency, so the three tokens taken in a row might not even carry any meaning. Another issue is that, statistically, chances of seeing the same trigram several times even in a large dataset are usually quite low, which makes it hard to train a good statistical model.

It is important to note that, for all the experiments with n-grams, NLTK method `ngrams(input, n)` was used. In accepts the input string and n, which defines how many tokens there are going to be in one n-gram. An n-gram is a tuple of length n. For example, when `n = 2`, the output list would have items of a format `("alpha", "beta")`, which is quite predictable. When `n = 1`, the method still outputs a list of tuples, even though each tuple will only contain one element and, if printed out, will have a format `("alpha",)` (by putting the comma after the only element, Python specifies that it is, in fact, a tuple).

When the `ngrams()` method was used to obtain unigrams, surprisingly enough, the resulting model training was also much faster and memory-efficient than when the default methods were used initially. It could probably be explained by the inefficiency of the NLTK library methods

splitting the input. So, this unigram model was used for all the consequent experiments with different classifiers.

6.2.3 Classifiers

Since the classifiers have already been implemented in the NLTK, it provided an opportunity to test the behaviour of the same dataset on different classifiers and compare them, with minimal effort and time.

TextBlob polarity

The first model that has been tried for the sentiment analysis was the TextBlob's sentiment analysis model which can be accessed using the `polarity()` method, which returns -1 if the text is negative and 1 if it is classified as positive. It is an example of a knowledge-based classifier. The results given by the classifier were not very high. When tested on the Sentiment140 dataset, the accuracy of the classifier was 60.04%. When working with live untagged data, it could be clearly seen that, in a lot of cases, the model would disagree with a general human opinion. The rest of the models created and tested are statistical.

Naive Bayes Classifier

In most of the papers used as a basis for this project and referenced in the Context Survey, Naive Bayes classifier is mentioned as a very simple classifier that is rarely expected to perform well due to the independence assumptions made, but also as a classifier that often performs almost as good as the state-of-art classifiers.

Indeed, the initial model proved to be quite successful. At different stages of the project, its accuracy would be between 53% and 88%.

The initial versions of the classifier were very slow to train. In these versions, a classifier would accept a tweet as a string and use default methods to tokenize the tweet and perform some pre-processing on the input data, such as removing punctuation.

Figure 6.5 shows the most definitive features of a model trained on 4000 tweets, where each tweet was tokenized using `TweetTokenizer`.

Later the default methods of obtaining feature vectors were replaced by more efficient methods which perform tokenization and extracting unigrams from the list of tokens.

Support Vector Machine

The Support Vector Machine instance is based on the `LinearSVC` implementation by the `sklearn` library. Ultimately it was the fastest one to train and the most precise one in terms of the results. The model would accept a large number of tweets as a training set. Empirically, the training size of 300 000 tweets was found to be the best for the purpose of training this sentiment analysis model. The model is used later throughout the project.

```
Most Informative Features
          contains(sad) = True              0 : 1      =     28.1 : 1.0
          contains(yay) = True              1 : 0      =     14.7 : 1.0
         contains(hate) = True              0 : 1      =     14.5 : 1.0
         contains(Good) = True              1 : 0      =     13.0 : 1.0
      contains(excited) = True              1 : 0      =     13.0 : 1.0
        contains(enjoy) = True              1 : 0      =     11.3 : 1.0
           contains(IS) = True              1 : 0      =     11.3 : 1.0
         contains(yes) = True               1 : 0      =      9.5 : 1.0
     contains(remember) = True              1 : 0      =      7.8 : 1.0
        contains(Hello) = True              1 : 0      =      7.8 : 1.0
          contains(ask) = True              1 : 0      =      7.8 : 1.0
       contains(Thanks) = True              1 : 0      =      7.8 : 1.0
          contains(YAY) = True              1 : 0      =      7.8 : 1.0
        contains(HAPPY) = True              1 : 0      =      7.8 : 1.0
         contains(lets) = True              1 : 0      =      7.8 : 1.0
```

Figure 6.5: Most definitive feature of a Naive Bayes Model, using TweetTokenizer

Other classifiers

The other two classifiers available in the NLTK library are the Maximum Entropy Classifier and the Decision Tree Classifier.

Unlike the Naive Bayes model, the Maximum Entropy model does not make the independence assumption. From all the models that fit the training data, the Maximum Entropy approach selects the one which has the largest entropy. Intuitively, the model should perform well on the tasks involving Natural Language Processing. However, its current implementation in NLTK did not produce good results. The model was too slow during training and was not memory-efficient (would constantly cause the program to run out of memory). At the same time, the existing SVM and Naive Bayes classifiers performed on a high enough level, so it was decided not to use the Maximum Entropy model. Finally, the results received from that model were significantly lower than the ones returned by the previous models. With 6000 tweets as a training set, the final accuracy was 63%, which was lower than the Naive Bayes Model or the Support Vector Machine.

The Decision Tree classifier was the slowest yet. The execution time of training a model on just 1800 tweets exceeded twelve minutes and even though it produced a high accuracy of 80%, the result was comparable to the one produced by the previous faster models, so it was decided not to use it either.

Resulting platform

The resulting application that performs the sentiment analysis training can be seen as a platform. The main method accepts a number of parameters such as the name of the classifier to use, how many items there should be in an n-gram, the length of training and testing sets, and the name of the feature extractor, because several test ones were written throughout the project. The program can train four different classifiers on custom-defined training and testing data, use

three different feature extractors (a simple n-gram extractor that does almost no pre-processing, a noun phrase extractor that tries extracting noun phrases using library methods — rather unsuccessfully — and the pre-processing extractor, the most developed feature extractor that does all of pre-processing described above and then extracts n-grams). This allows creating and comparing different models, making the application scalable and maintainable. It could later be used in other work related to Natural Language Processing.

6.3 Topic Modelling

For the topic modelling, the Falkirk Council provided a list of concepts they would like to trace. The first step in designing a model is similar to the sentiment model and requires obtaining enough training data. However, unlike the previous case where the training data could be obtained online, the training data for such specific topic as this could not possibly be found online. There was only one way to form training data: to request a certain amount of tweets using Twitter API and to label them manually or in a semi-automatic way.

However, it was impossible to get any training data for this topic at all. Most of the concepts have never been mentioned in any tweets mentioning Grangemouth, and the ones that were mentioned did not have enough occurrences to form a plausible training set.

In addition, Twitter API has certain limits on both number and age of tweets that could be requested [9]. Currently, the search function allows making 150 requests in a time bounded by 15 minutes when the user authentication is used and 450 requests for the same time window through the application authentication. For this project, application authentication was used. However, there is also a time limit for the tweets that could be requested. Twitter API only allows the request of tweets from the last several days (the exact number of days has been changing frequently throughout Twitter history). Due to the low Twitter activity around the Grangemouth area and the fact that the tweets mention environmental issues quite rarely, none of the relevant tweets could have possibly gotten into the required time window. Hence, the training data could not be obtained.

6.3.1 Rule-Based Model

As discussed in the Context Survey, most of the existing topic modelling techniques are based on statistical models, and, hence, all of them require labelled data, even when it comes to unsupervised learning techniques. Thus, none of them could have been used for this task.

At this point, it was clear that any model created could not possibly be made scientifically sound. Introducing sample training data would require a lot of time and effort. However, the biggest problem with this approach is that it would have been impossible to introduce data that

[9] `https://dev.twitter.com/rest/public/rate-limits`

would have been purely "natural" and not biased, even if the data was created in the process of a user study.

Therefore, it was decided to create a small manual filter for the given topic which would work with a decent accuracy and could be shown to the client and then change the topic in order to keep the scientific side of this project sound.

The so-called rule-based model is contained in the `src/rb_environmental_classifier.py` file. The pre-processing methods used in this module were later re-used in the creation of a statistical model. The main approach to this classifier was taken from "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text", a paper on sentiment analysis which also used a rule-based approach and claimed that the resulting model outperformed the most common sentiment analysis models [36]. Hutto, the author of the paper, also provides the implementation of the classifier[10].

VADER

The VADER tool is based on the generalizable rules which were created and validated using qualitative and quantitative methods. The implementation uses constants such as booster word coefficient increase `B_INCR = 0.293` and an ALLCAPs increase `C_INCR = 0.733`. In every such case, the number is said to have been obtained empirically. The tool uses a list of punctuation, a pre-defined list of negations, some of which are not just the usual negations such as "couldn't" but also the less obvious options such as "uh-uh". The authors also used a list of degree adverbs, but it was never specified where the list was taken from, and it can be clearly seen that the list only contains a small part of all the degree modifiers used in the English language. Finally, a local vocabulary is also used, which contains words together with their individual sentiment scores.

The processing VADER uses is based on a single coefficient that is increased or decreased depending on the appearance of different features. First, sentiment score is extracted for every word or emoticon and stored in a list. Then, every word is checked for ALLCAPs and, if it is written in all uppercase letters, the valence for that word is increased by `C_INCR = 0.733`. Similar slight modifications happen for cases such as negation, idioms, and others. Finally, the list of individual scores is summed up and, in case of special punctuation appearing in the input, "amplified". The resulting score is then returned.

Grangemouth topic classifier

The Grangemouth rule-based classifier tried to follow the same approach, however, it was clear from the beginning that it would not be very successful. In the case of VADER, the rules — of the format `condition -> modify(coefficient)` — were carefully designed using both qualitative and quantitative methods. All of these methods require an immense amount of time and need users to label statements manually. Such project could not possibly fit as a part of this project,

[10] `https://github.com/cjhutto/vaderSentiment`

hence all the coefficients, constants and rules are based on what seemed to be common sense, but obviously none of them was based on scientific results.

The classifier uses the given list of keywords provided by the Falkirk Council as well as some other pre-defined lists such as a list of mentions which could be relevant to environmental issues. For example, the user names of the Falkirk Council Twitter account (falkirkcouncil), the INEOS and BP Twitter usernames. The latter two companies have a number of accounts each, all starting with "INEOS_" or "BP_", so regular expressions were constructed for these cases. Also, names of the companies were added as well, because some users might not know that the companies have a presence on Twitter, and SEPA (Scottish Environment Protection Agency) was included, too. Another list was constructed for anti-mentions since the initial analysis of typical tweets mentioning Grangemouth showed that some accounts will almost exclusively specialise in one topic. For example, `MadrasRugby` mentions Grangemouth in a lot of its tweets, however, the account only ever talks about rugby.

Overall, the **initial lexicon** was defined as a collection of groups of words of different parts of speech (e.g. nouns, adjectives).

As the first step, the lexicon is initialised for the classifier. In order to do that, the program goes through the initial lexicon and, for every word, performs the following:

1. The word is **stemmed** using the `WordNetLemmatizer` provided by `NLTK`. Stemming involves removing any morphemes of a word that are not in the core of a word. For example the word "unbelievable" should return "believe" as a stem. Stemming allows recognising groups of words regardless of their form. For example, we would usually want to recognise words "burn" and "burning" as one without having to store both words in the vocabulary and without having to generate all possible forms of the same word since every stem can be involved in the creation of quite a large list of words. Hence, it seemed reasonable to store stems rather than full words. One could argue that other morphemes also carry some amount of information, however, they are not useful when it comes to topic modelling. For example, the form "cats" has an affix '-s' which carries a meaning of plurality, however, it does not change the topic itself. So does the suffix 'over-' in the word "overachiever". When expressed in terms of the First-Order Logic, it can be seen that the affixes provide the functions rather than the arguments. Hence, for the purpose of topic modelling, stems are sufficient sources of information about meaning.

2. The stem is added to a list of stems of the derived lexicon.

3. Synonyms of the word are found using the `wordnet.synsets(word)` function. Potentially, the word network could be grown further by finding synonyms of every synonym. Every synonym is then also stemmed, and the new stems are also added to the dictionary.

At this point, the lexicon is considered finalised, and the classification can begin. Again, the classifier works with a single number which is modified slightly depending on the appearance of different features in a tweet. The amount of change is a guess based on the knowledge of the data and the fact that all the tweets about Grangemouth going back to as far as 2009 were read. The classification procedure follows the following approach:

1. First, the input is checked for special characters. A horizontal ellipsis is mostly used in tweets containing links, which are mostly news tweets. Currency symbols are normally used in economic news and almost never is complaints about the environment. Hence, both types of special character reduce the coefficient.

2. The tweet is parsed using the `preprocessor` library. URLs generally reduce the chance that a tweet talks about environment, especially if they are not links to Instagram or Twitter internal links (usually link posts together). All the links on Twitter are currently shortened with the `t.co` domain, however, it is possible to extract the actual URL using the `urllib2` library. Seeing a name mentioned in names or mentions increased the coefficient, whereas seeing a name from the "anti-mention" list decreases it.

3. The tweet is stripped of URLs and mentions.

4. The tweet is tokenized using `TweetTokenizer`. Every token is stemmed and the stem is then looked up in the dictionary. Known stems increase the coefficient.

5. The tweet is checked for idioms. Some tweets have been seen mentioning expressions "smell a rat" and "under fire" (for example, "the plan to start fracking is under fire"). The words "smell" and "fire" are in the list of the initial keywords provided by the Falkirk Council, so the classifier is biased when it notices the words. Hence, when these expressions appear in the tweet, the coefficient is decreased.

A positive coefficient means that the tweet does, indeed, talk about environmental issues.

Since this was a rule-based approach and none of the data was labelled, the only way to test a model was to do it manually. Several examples were copied from the existing Twitter history and some more examples were hand-written. The only result that could possibly be given by such imprecise evaluation is that the classifier seemed to perform reasonably well. The model gave no false negatives and several false positives. It could potentially be improved in the future.

6.3.2 SVM Topic Model

Topic Selection

A new topic that was supposed to be chosen for the model had to satisfy the following criteria:

- It had to have enough accessible samples to train a model.

- The topic had to be a sub-topic of a larger topic such that the larger topic was represented by a set of tweets roughly twice as large as the set represented by the sub-topic.

As a result, the larger topic was chosen to be “Elections” and the sub-topic was chosen to be “UK General Elections 2017”. At the time when the topic was chosen, the UK General Election had just been announced. At the same time, when searching for the keyword “election”, the search also returned a lot of tweets about the investigations concerning the US Presidential Election, the, at that time, upcoming French Election as well as some other elections in other countries and organisations.

The tweets were collected and saved in a JSON file (`resources/election_tweets.txt`) as a list of entries of the following format: `{"text": "What should the world expect if Marine Le Pen takes power? https://t.co/ZLRufRmkDW", "label": 0}`. The ‘label’ value is 1 if the tweet is related to the UK General Election of 2017 and 0 otherwise. The tweets were labelled manually. There are 870 tweets in the dataset, out of which 405 are positive and 465 are negative.

Vocabulary

In order to create a successful model, instead of using simply a bag-of-words approach, it seemed to make sense to use an approach that is a little more complicated. Twitter API does not allow accessing tweets that are older than several days. In addition, Twitter search by keyword returns every tweet containing that keyword, among which are also retweets, re-postings of the original tweets starting with an “RT” sign, if requested via the Twitter API. When it comes to popular topics and popular accounts, some tweets become ‘viral’ and get retweeted thousands of times within a time frame of just several hours. This makes the sample of tweets obtained from the Twitter API biased because it might focus on smaller topics that become very popular for a short amount of time and would bias the classifier in the long run.

So, in order to avoid creating this bias, it was necessary to create a basic vocabulary similar to the one provided by the Falkirk Council for the initial task. It was important to generate a vocabulary that would not introduce bias itself. One of the proposed approaches was to generate the dictionary from the news article tags taken from most popular newspapers available in the United Kingdom. Article tagging helps create a hierarchy for the publishers, thus introducing opportunities for additional automatic tools such as filtering of search results. Article tagging can, in fact, be an example of a task involving topic modelling, since today the largest publishers who have a large amount of new material published every day do it automatically. For example, Nieman Lab published an article in 2015 that described how New York Times uses machine learning to achieve that goal [20].

However, there is no tool that would allow growing such a network of tags automatically or extracting one from a publisher’s website. In addition, the tags observed on different websites were too ambiguous and there were simply not enough of them to start a dictionary. For example,

figure 6.6 shows how Guardian[11] tags its articles. The network obtained from it would be quite small and mostly only mention the names of parties and people involved. Figure 6.7 shows the same problem with the tags used on the Independent website[12].

Figure 6.6: Topics of an article on Guardian

Figure 6.7: Topics of an article on Independent

Because of this, it was decided to look into other tools that would help grow the network.

Google Tools

Some of the products created by Google are not well known but can be extremely useful when it is necessary to process an immense amount of data. Three Google products provide functionality that allows extracting networks of related words, with different contexts and for different reasons.

The first such tool is Google AdWords [13], which allows creating advertisements and promoting them through the Google platform. As a part of the functionality provided by the platform, Google offers to create a set of keywords that would be associated with the advertisement. Figure 6.8 shows the keywords found for “general election”.

As it can be seen, most of the keywords are not relevant to the 2017 UK General Election specifically, and the list is too small to operate on.

Another Google-provided tool that focuses on finding related keywords is Google Correlate[14]. The tool searches through the search patterns of people from the specific period of time and detects common patterns. Figure 6.9 shows the searches correlated with “UK general election”. It

[11] https://www.theguardian.com/uk
[12] http://www.independent.co.uk/
[13] https://adwords.google.com/
[14] https://www.google.com/trends/correlate

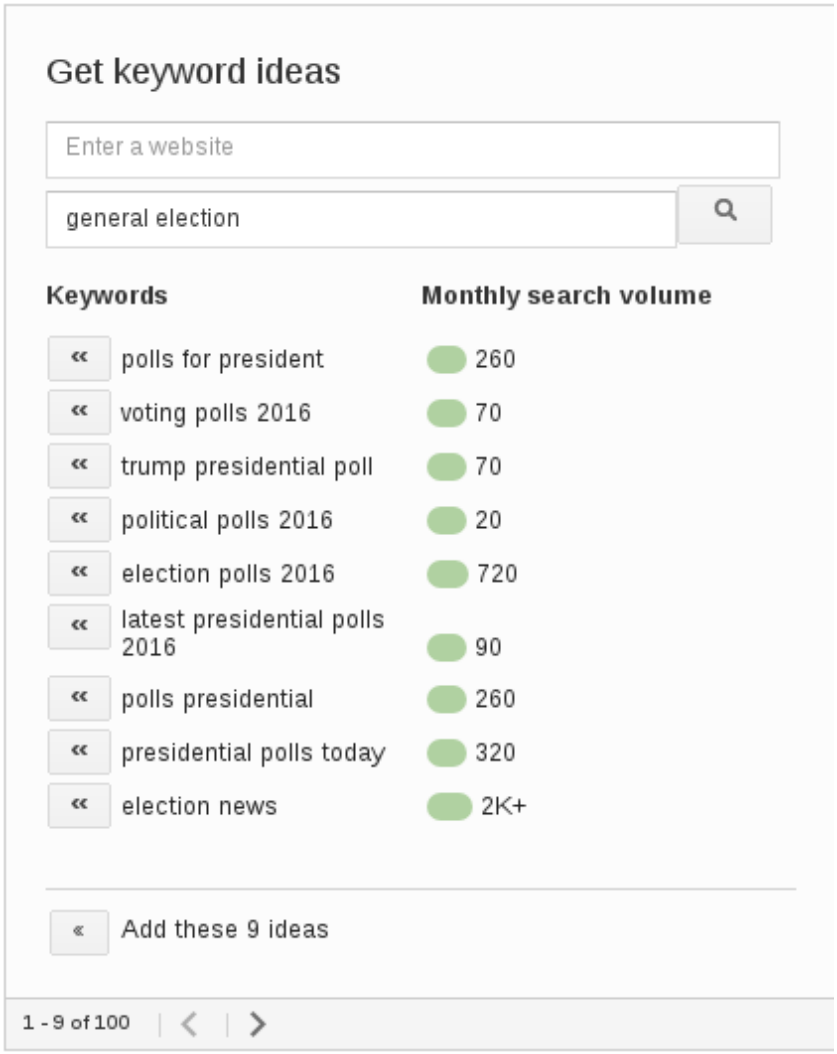


Figure 6.8: Related keywords from GoogleAdWords

can be seen that this result is also not perfect despite being far better than the one obtained from GoogleAdWords. The first two results are simply combinations of words from the request, and four other results are different forms of the same word. At the same time, the correlation with one party was found, and with one UK region, too.

Finally, the tool that returned the best results for the initial vocabulary was Google Trends[15]. Google Trends is a tool that also processes the searches to detect trends and is, in fact, at the core of the Google Correlate. The tool started as a mechanism to create a flu surveillance system based on the symptoms searched on Google by people at different locations. Now the tool provides similar functionality of detecting different trends about any topic requested by the user. When "2017 UK general election" was requested, the tool provided useful and interesting data summary

[15] `https://trends.google.co.uk/trends/`

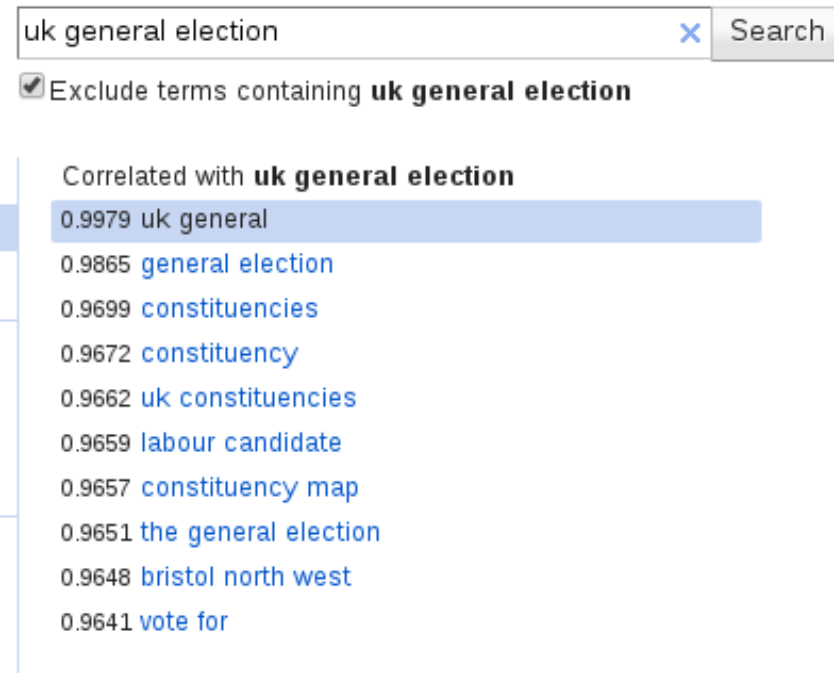


Figure 6.9: Related keywords from Google Correlate

including two sections which were later used as a basis for the initial vocabulary. The two sections are Related Topics and Related Queries, two lists of up to 25 entries. A list of related topics provides a list of topics that users searching for the original query (in our case, 2017 UK General Election) search for, too. The second list, related queries, is a list of unprocessed queries that people who search for the original topic also searched for. Both of these lists could be exported as CSV files. Additionally, the network could be grown by expanding topics exported in the Related Topics list.

After all the necessary CSV files had been saved, they could be used in the topic model. First, however, the dictionary had to be finalised. The `src/vocab_creator.py` deals with this. First, the CSV files are all read in and a common list of concepts is then written out to one CSV file. This file is the first draft of the vocabulary. Then the vocabulary was modified manually: names of some leaders of the existing UK parties were added, for example. In addition, names, nicknames and Twitter account names were added for every existing UK party.

After the initial version of the vocabulary has been finalised, it could be expanded in the same way as it was done for the rule-based classifier: by adding stemming the existing words and adding stems of their synonyms.

The Support Vector Machine

As it was discussed in the Context Survey, most of the existing topic classifiers use the usual

machine learning techniques including the Support Vector Machine approach and Naive Bayes classification. For this project, it was decided to use the SVM approach.

There were four feature vectors attempted:

1. The first extractor summarises the tweet into a vector of a small size and that looks like this: `{'titles': 1, 'names': 1, 'nouns': 2, 'stems': 3, 'abbreviations': 2, 'mentions': 1}`. The extractor uses all of the pre-processing described for the rule-based classifier. Its main advantage is that it allows the vector to be very compact. At the same time, it throws away all the information that does not fall into any of the known categories.

2. This extractor does not throw away the unknown words. Instead, it adds them to the feature vector with a True Boolean value: `{'unknown': True}`. This allows the SVM to learn more about the text itself, however, might also introduce bias and makes the vector much less compact. Overall, it is hybrid approach which uses a combination of human-defined categories and the Bag of Words approach.

3. The third extractor is very similar to the second one, except it also combines names and mentions into one category instead of two. This model produces almost the same results as the second feature extractor.

4. Finally, the fourth extractor is based purely on the Bag of Words approach. This extractor does not need the vocabulary at all and only learns from words seen before.

After that, k-fold validation was performed on the existing model, for all feature extractors. The results show that the models produce very similar results and it will be discussed in more detail in the next section.

6.4 Analysis of Results

The models can be evaluated both separately and together. Let's start with the sentiment analysis model.

The platform created allows for easy comparison of different models. The results of the SVM and Naive Bayes models for unigrams, bigrams and trigrams and for training sets of different sizes were obtained using 10-fold validation. For each model, we were interested in both accuracy and recall, since the combination of both describes a model best, and the best model will produce high results in terms of both accuracy and recall.

The model was trained using sets of different sizes: 20000 tweets, 50000, 75000, 100000, 200000 and 300000 tweets. After the series of experiments was run, the `matplotlib` library was used to plot the results on a graph.

Figure 6.10 shows accuracy results obtained in a series of experiments. The model based on unigrams performs best throughout the series. It also plateaus faster than the models based on bigrams and trigrams, but the difference in accuracy is also quite high.

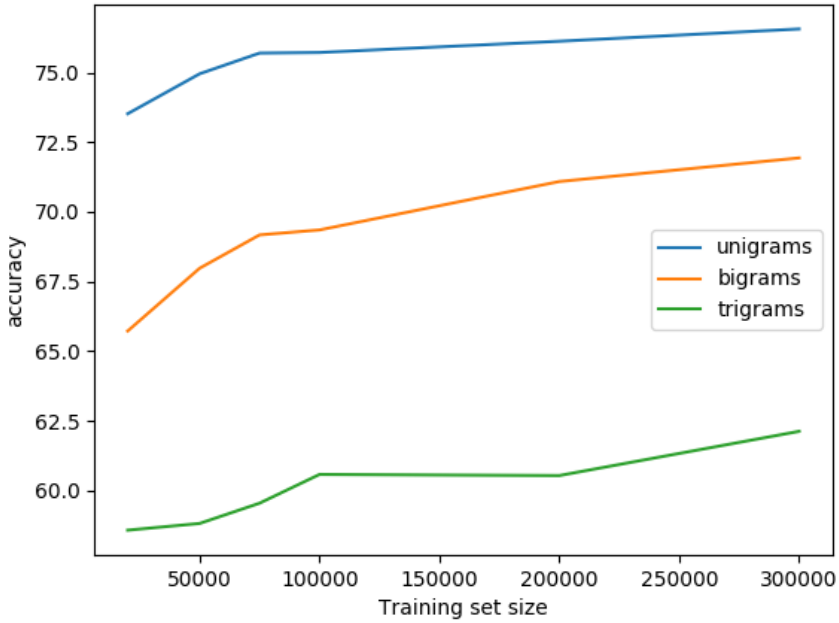


Figure 6.10: Accuracy of Naive Bayes model

Figure 6.11 shows that, for trigrams, the recall is the highest. On average, the recall for trigrams is about 5% higher than for bigrams and unigrams. The recall for unigrams and bigrams is similar for unigrams and bigrams, however, unigrams perform slightly better in almost all cases apart from the smallest training set.

Overall, it would be safe to say that the unigrams perform best of all the other architectures. Even though the recall is best for trigrams, the results for trigrams are also quite inconsistent and it falls when the dataset is large.

Figure 6.12 shows the accuracy analysis for the SVM model. In the case of this model, unigrams perform best, too, and the figure 6.13 shows that its recall is also better than for any other models. Another thing that could be noticed is that the values for accuracy and recall are pretty similar and that also recall grows as the accuracy grows.

Figures 6.14 and 6.15 compare unigram-based SVM and Naive Bayes models in terms of their accuracy and recall. The accuracy comparison shows that, for smaller datasets, Naive Bayes performs slightly better. The difference between the models is below 2%. For the largest tested dataset consisting of 300000 tweets, the SVM model performs better than the Naive Bayes. Another interesting fact that could be noticed is that neither of the models could be seen

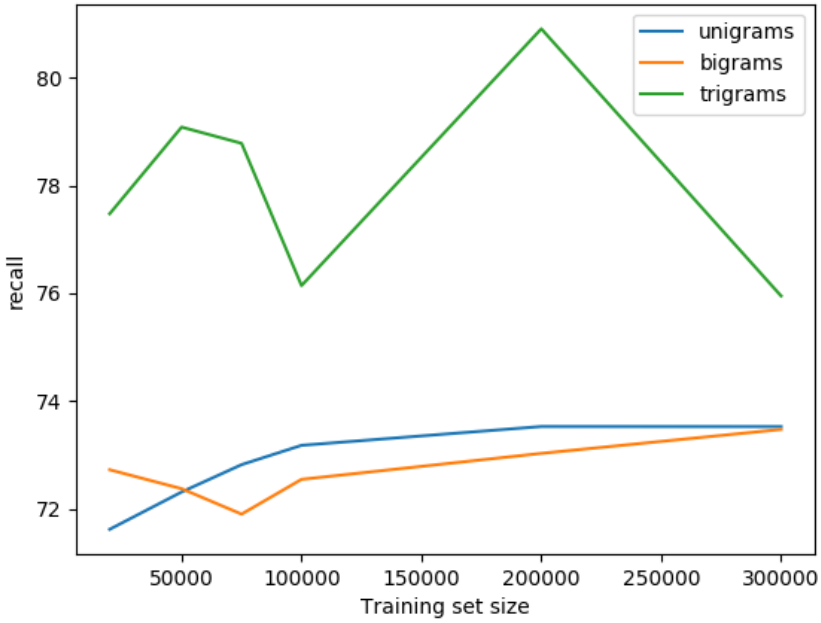


Figure 6.11: Recall of Naive Bayes model

plateauing, so the accuracy is likely to grow even more for larger training sets.

The recall comparison shows that SVM performs better than Naive Bayes, especially when it is trained using the largest training set. The recall for the SVM is just below 78%, whereas the same property for the Naive Bayes model is just above 73%. Moreover, it can be seen that the Naive Bayes model plateaus and might be prone to over-fitting if a larger dataset is used.

Overall, it can be concluded that the unigram-based SVM performs the best sentiment analysis.

Next, it is useful to talk about the topic model. In this case, the dataset was initially quite small (less than 900 tweets) since it had to be labelled manually. In this case, we can compare the performance of the models based on four different feature extractors. The first extractor summarised the data into a small dictionary which mentioned several word categories and the number of occurrences of words in a tweet. The second extractor also used the Bag Of Words approach for words that did not fall into any of the categories above. The third extractor also combined the mentions and names into one category. The final, fourth extractor, only uses the Bag of Words approach and does not consult the vocabulary at any point.

Figure 6.16 shows the results for different extractors. One important result here is that the extractors using the Bag of Word approach in addition to the human-defined categories — extractors 2 and 3 — perform better than the one using only categories. This shows that humans might not, in fact, be very good at identifying important features which help classify text. The

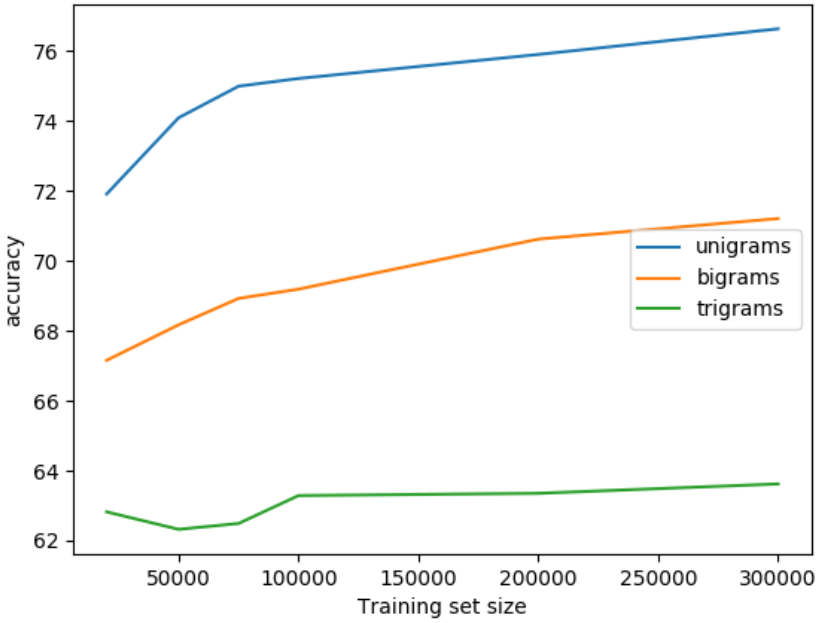


Figure 6.12: Accuracy of SVM model

first extractor was based on few intuitive human-created categories. In theory, a text talks about elections if it mentions words from the defined vocabulary and mentions people and parties specified in the vocabulary as well. In reality, the fact that the Bag of Words approach works better means that it is very hard to create a perfect vocabulary. The given vocabulary is not a perfect one and maybe does not mention some of the words which would help in the classification. However, in addition to that, a possible reason Bag of Words approach performs better could be that the words that are not directly related to elections might also carry additional meaning which, combined together in a certain pattern, also helps classify text correctly. Identifying such patterns is quite hard and adding them to the vocabulary is quite complicated, too, because they should not be treated on the same scale as the main words in the vocabulary. For example, a phrase “promised to” might theoretically be related to the elections and increase the chances of a tweet being about the UK general elections. However, adding it to the same list as the main words such as “Brexit” and “snap election” would bias the vocabulary. Instead, it might then be useful to determine the ‘weight’ of every word in the vocabulary. This, however, is hard to calculate manually. This means that the Bag of Words approach has an advantage when used with existing Machine Learning Techniques.

At the same time, extractor 4, which is only based on the Bag Of Words approach, performs worse than the extractors 2 and 3. This means that the categories do help the model perform better, and knowing the structure as well as the main features of the data given helps classify

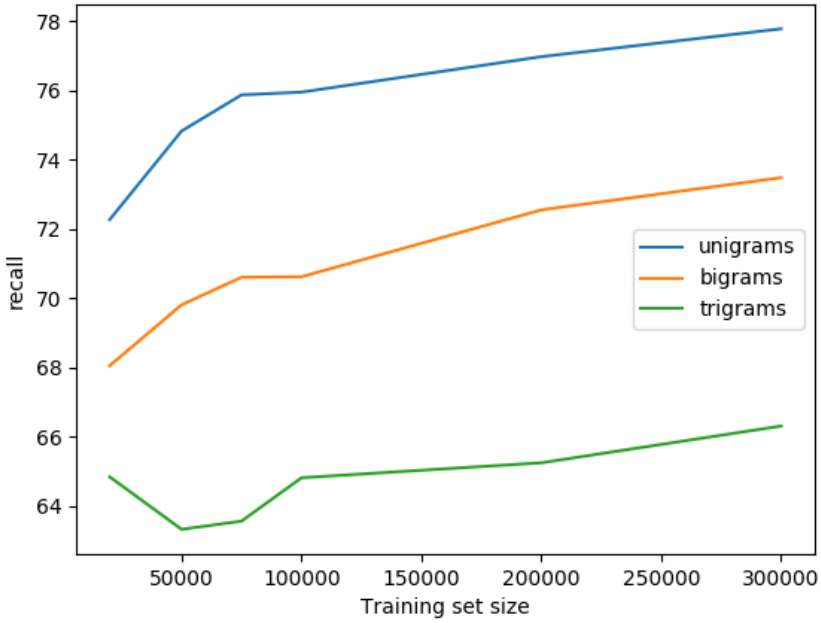


Figure 6.13: Recall of SVM model

the unseen data.

Of the extractors 2 and 3, the one where the Twitter mentions and names are combined into one category performs slightly better. The difference in the accuracy and recall is very small, but so is the difference in the feature vectors. This difference, again, means that sometimes it is hard to come up with meaningful categories by hand. The tweets usually do not use a lot of mentions or names in one tweet. This means that usually most of them will have values 0 or 1 for either of the categories. Combining the categories into one means that instead of having two small numbers, there is one larger number (usually up to 4). This changes the shape of the feature vector and, hence, changes the results of the model.

Overall, the best topic model is a hybrid one, one based on the combination of human-defined categories for an existing lexicon and the Bag of Words approach. The accuracy of this model obtained using 10-fold validation is 88% and the recall is 85%.

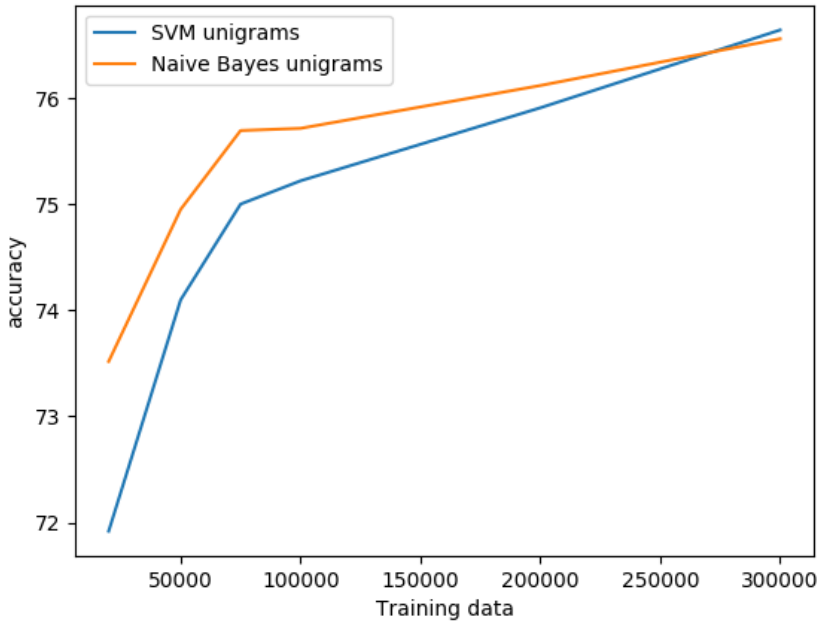


Figure 6.14: Comparison of accuracy results

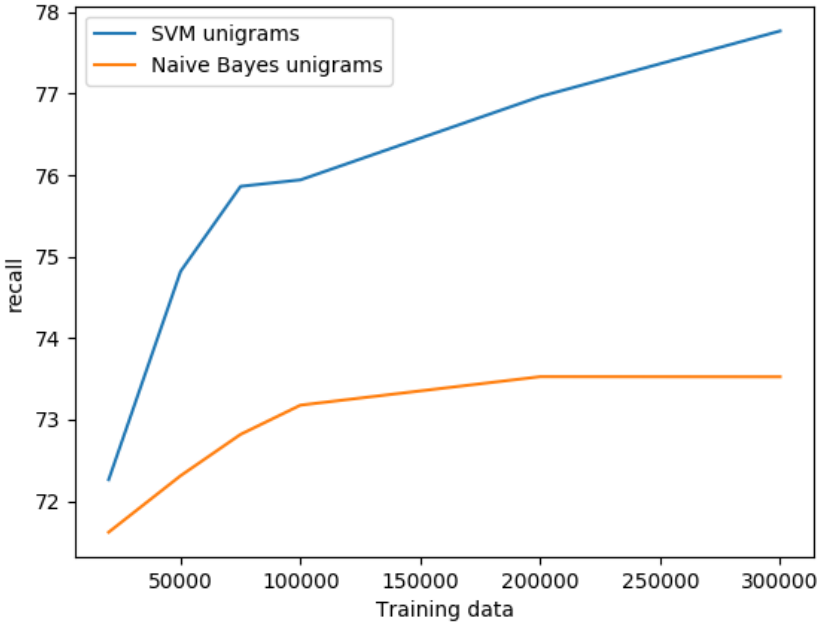


Figure 6.15: Comparison of recall results

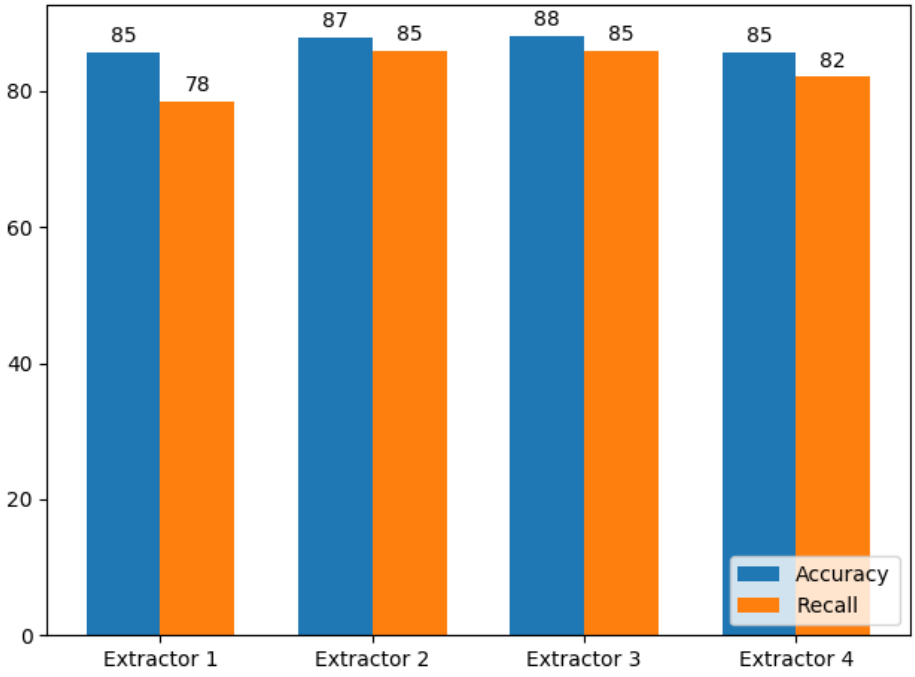


Figure 6.16: Comparison of feature extractors of the topic model

CHAPTER 7

EVALUATION

Most of the attention in this chapter will be given to evaluating sentiment analysis and topic modelling parts of the project, as well as testing results. Overall, the project has reached all its main goals. The sentiment analysis and topic models have been successfully implemented, tested, validated and packed into an application which could be run from a command line. Depending on the further requirements, the output of the program could be presented in a number of ways such as email or push notifications, as well as a simple command line output.

7.1 Comparison to the original objectives

Some objectives have been changed — as a result, they have been replaced with other objectives or scratched out.

Primary objectives:

- **Extract the existing data from Twitter**: This has been achieved fully. The program can access tweets by a keyword using the Twitter API.
- **Study the existing text classification techniques, compare them and identify those that could be used for the current project**: This has been achieved as well and has been discussed in Chapter 2 (the Context Survey).
- **Train a model that recognises negative tweets about Grangemouth**: A sentiment model has been created, tested and evaluated and showed high results. This model can be used in general and works for any tweets, not just the ones about Grangemouth. The sentiment analysis program was made to be scalable and maintainable, it was created as a

platform that can be instantiated and trained in a large number of ways, depending on the command-line arguments given. For the topic modelling, the project had to divert away from the original topic (Grangemouth). As a consequence, two classifiers were implemented. One of them is rule-based but was implemented randomly, and, hence, could not be evaluated objectively. The new statistical topic model focuses on the UK General Election. The model has been created, tested and validated using k-fold validation.

The **secondary objectives**:

- **Create a mechanism to send the selected tweets to the Falkirk Council as notifications**: this objective has not been implemented because the project had to change the topic, so the output could not be delivered to the client straightaway. Instead, the program returns the output, and, as a part of future work, a simple application can be written to send the output to the client in a way they want.
- **Train another model**: This has been implemented in a number of ways. For sentiment analysis, several models can be trained easily using the same code base by providing different arguments. Any of these classifiers can be combined with the topic model to produce different outputs.
- **Analyse and compare the models and their effectiveness**: This has been done and the results were discussed in Chapter 6.
- **Create a live tool that will receive a stream of tweets and classify them 'on the go'**: This has been implemented fully and can be loaded from the `src/twitter_tool.py` file. The procedure trains the sentiment analysis and topic models, after which it starts requesting tweets from Twitter and evaluating them as they get returned by the API.
- **Look into existing studies of detecting irony and sarcasm in the text**: This has been researched and discussed in Chapter 2 (Context Survey). However, it was out of the scope of this project to attempt the implementation. An idiom filter was implemented for the rule-based version of the topic classifier, and any further work would probably also be based on maintaining a dictionary of known idioms and using it as a filter. However, that could potentially require changing the structure of the feature vector away from the Bag-of-Words approach since, to detect an idiom in a sentence, the order of words will matter more.

7.2 Testing

The main parts of the project have been tested both manually and using unit tests. Since the implementation involves classification using classifiers implemented by `NLTK` (Natural Language

Toolkit) and other APIs and libraries, it is not possible to test certain parts of the implementation using unit tests. Such parts of the program are, for example, streaming tweets using Twitter API, searching tweets using the same library, processing the tweet using `preprocessor` library and text classification using `NLTK`'s classifiers. These parts have been tested manually under a number of different conditions and different user inputs, and, in some cases, the methods `test()` remain in the code and can be called from the main methods of the corresponding modules.

The unit tests can be found in the `src/tests` folder and run individually using the command `python <filename.py>`.
These check a lot of input types for methods that perform preprocessing on data that is being used for classifier training, testing and validation. All the unit tests pass.

7.3 Relation to Existing Work

I'd like to think of this project as being a mid-point between research and industry. Most of the work related to Natural Language Processing has been done in research, and, despite the fact that it is also widely used in the industry, the application and the power of existing NLP techniques has not been fully explored by the industry. Therefore, its usage has been quite shallow. Nowadays these techniques are mostly used in marketing, media and politics. However, NLP can be an extremely powerful tool, and there is still a long way to go in order to fully use its potential.

All of this project has been based on the vast amount of research done in this area and builds up on the latest papers, too, including a lot of materials from the past two years. The techniques used at the root of the classification, such as the SVM and Naive Bayes classification, have been known for a while, but a lot of nuances involved in the feature vector extraction are new. For example, using emojis in sentiment analysis has never been discussed in any papers. The latest research, however, defined the sentiment of every known emoji, and it was possible to use the Web-scraped results in the sentiment model, which has never been done before.

Certain industry tools exist that perform Natural Language Processing. A lot of them, used in marketing and media, are internal tools of different corporations, so one of the very few publicly available tools is the Google product suite. When it comes to training statistical models, Google has the clear immense advantage of having access to tremendous amounts of data. However, as it can be seen from different papers and existing tools, almost all of them use the same techniques such as stemming, finding synonyms and patterns and using the Bag-of-Words approach to training their models. This means that this project has been done on a similar level, despite taking certain shortcuts due to the time and resource limitations. In terms of the precision and recall of the models, they are also on the same level as the models in the existing research papers.

CHAPTER 8

CONCLUSIONS

The project could be continued and a lot more Natural Language techniques could be applied to the models. The main constraint of this project has been time — a lot of fascinating topics have only been touched slightly. A lot more time and other resources could be spent on this project. For example, noun phrase extraction has been tried as a part of the sentiment analysis, and it would be very useful to have it implemented. However, the scope of this project only allowed a library (`NLTK`) method to be used for noun phrase extraction, and it proved not to be very successful - each noun extraction would take almost 3 seconds, and a lot of noun phrases would not be detected. It would be beneficial to implement a fast and light noun phrase extractor, however, it would require a lot more time.

The project leaves a lot of open questions that have not been answered by most recent research. For example, the Bag-of-Words approach is currently one of the most popular structures used in NLP. This is the simplest approach that allows for knowing almost nothing about the data that is being used. However, it would be interesting to see how the models that use the existing features of the data would perform. For example, is there a correlation between the location of the user and the sentiment of their posts? How can a tool detect idioms efficiently?

Another unanswered question of NLP is whether irony and sarcasm could ever be detected by a machine, especially when people are not that good at detecting them themselves.

Also, in the future work, comparing models created from the entire words to models created from only stems of the words and also maybe models created purely from affixes of the data could give unexpected results and be useful not only in Natural Language Processing but also in linguistics and literature. This research would show how much of emotional information different morphemes hold.

On a higher level, it might be valuable to understand how much emotional information

different parts of speech hold. Intuitively, most of the information about the sentiment is being passed by adjectives and adverbs. Is it then possible to predict the sentiment of a sentence correctly using only words of these two classes? Would a model that contains no nouns perform much worse, and, if yes, how much worse?

If this project is ever continued, the person working on it should take all the points above into consideration. Despite the fact that the direction of the project had to be changed slightly, I hope that, with all the effort that has been put into this project, it will prove useful and contribute to a very exciting project and, hopefully, still move the Falkirk Council's project closer to the goal.

It was incredibly engaging to work on this project. I learnt an immense amount of information, realised that NLP is one of the most interesting fields of Computer Science (for me, personally), and received a lot of experience in Natural Language Processing, Machine Learning, and working with a wide variety of APIs, had an amazing time working on the project. I also consider everything I have learnt throughout working on this project to be my largest achievement to date.

APPENDIX A

USER MANUAL

This project could be compiled and run in a number of different ways, both as a whole and part by part. In order to compile and run the project, certain dependencies and libraries have to be installed, such as `NLTK`, `preprocessor`, and `scikit-learn`. The project uses Python 2.7. All the necessary libraries are compiled in the `resources/requirements.txt` file of the project and are also attached in appendix C of this report.

A lot of modules of the project could be run separately and can be run from the command line. To explore the functionality of each module, one could modify the `main()` method for each of them. By default, most of them currently perform a simple test of a module by giving it a simple task.

For example, to run and test the TextBlob classifier, run

```
python textblob_classifier.py
```

This will print the accuracy of the classifier, using the training data from the provided data set.

To run the rule-based classifier, run

```
python rb_environmental_classifier.py
```

This will create a classifier and print classification outputs for a number of given sentences from the `main()` method.

Running

```
python elections.py
```

will create a Topic Model and perform ten-fold validation on it. The method will print averages for both accuracy and recall values obtained during the validation.

To test the Sentiment Model, `ngram_classifier.py` has to be run. The command also accepts five arguments:

1. **Classifier name**:
    - "Naive Bayes"
    - "MaxEntClassifier"
    - "SVM"
    - "DecisionTree"
2. **The number of tokens in an n-gram**: 1 produces the best model, 2 performs reasonably well, any other positive integer will create a model which is not likely to perform well. Any other number will not be validated.
3. **The size of training data**: the dataset contains more than a million and a half of tweets. It has been found out empirically that a model does not need more than 300 000 tweets to train on in the case of SVM. In a case of another classifier, this number will be even less.
4. **The size of testing data**: any non-negative integer that, summed with the previous parameter, will not give a sum larger than the size of the entire data set.
5. **Name of the feature extractor**:
    - **"ngram_extractor"**: Simply extracts n-grams from the tweets and performs basic pre-processing.
    - **"preprocessing_extractor"**: Parses the tweet using the `preprocessor` library, then extracts n-grams. The most advanced feature extractor.
    - **"noun_phrase_extractor"**: Extracts noun phrases using external libraries. This is the slowest extractor, and the results are not satisfying.

.

So, a command

```
python ngram_classifier.py SVM 1 10000 200 preprocessing_extractor
```

will create and test an SVM for sentiment analysis from unigrams extracted using `preprocessing_extractor` from a training set of a size 10 000 and will test the model on 200 tweets.

The `twitter_tool` module contains methods that show the implementation of the entire model. To run it, run

```
python twitter_tool.py
```

from the /src folder.

This will create two models with the best parameters obtained empirically and discussed in the report and start a stream of tweets containing the keyword “election”.

Please note that, in order to train the models, the datasets have to be placed into the /resources folder. The datasets can be downloaded from:
https://mn39.host.cs.st-andrews.ac.uk/grangemouth/training.1600000.processed.noemoticon.csv
and
https://mn39.host.cs.st-andrews.ac.uk/grangemouth/Sentiment-Analysis-Dataset.csv.

APPENDIX

B

Topic Keywords

Sentiment Triggers for Grangemouth Project

Category				
Air Quality	**Odour**	**Pests**	**Waste**	**General**
Smoke	Burning	Flies	Fly tipping	Disgusting
Black smoke	Fish	Rodents	Litter	Gross
Flaring	Eggs	Vermin	Rubbish	Dangerous
Smog	Smells	Insects	Dumped	Health
Fumes	Stinks	Pests	Dirty	Harmful
Dust	Reeks	Rats		Polluted
Asthma	Odours	Mice		Pollution
Idling	Burnt matches	Cockroaches		Oil
	Smelly			INEOS
	Stinking			Petroineos
				Calachem
				Docks
				BP
				ICI
				Falkirk Council
				SEPA
				Horrible
				Nasty
				Vile
				Sewage
				Foul
				Hydrocarbons
				Flooded
				Environment

APPENDIX C

Python Requirements

```
appdirs==1.4.2
BeautifulSoup==3.2.1
chardet==2.3.0
funcsigs==1.0.2
mock==2.0.0
nltk==3.2.2
numpy==1.12.0
oauthlib==2.0.1
packaging==16.8
pbr==2.0.0
protobuf==3.2.0
pyparsing==2.1.10
requests==2.13.0
requests-oauthlib==0.8.0
scikit-learn==0.18.1
scipy==0.19.0
six==1.10.0
sklearn==0.0
tensorflow==1.0.0
textblob==0.12.0
tweepy==3.5.0
tweet-preprocessor==0.5.0
twitter==1.17.1
```

```
Unidecode==0.4.20
```

BIBLIOGRAPHY

[1] 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, Association for Computational Linguistics, 2011, pp. 30–38.

[2] L. H. ALEC GO, RICHA BHAYANI, *Sentiment140 - a twitter sentiment analysis tool*. `http://help.sentiment140.com/for-students`.

[3] S. BERGSMA, E. PITLER, AND D. LIN, *Creating robust supervised classifiers via web-scale n-gram data*, in Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2010, pp. 865–874.

[4] D. M. BLEI, A. Y. NG, AND M. I. JORDAN, *Latent Dirichlet Allocation*, Journal of machine Learning research, 3 (2003), pp. 993–1022.

[5] K. BLOOM, N. GARG, S. ARGAMON, ET AL., *Extracting appraisal expressions.*, in HLT-NAACL, vol. 2007, 2007, pp. 308–315.

[6] B. E. BOSER, I. M. GUYON, AND V. N. VAPNIK, *A training algorithm for optimal margin classifiers*, in Proceedings of the fifth annual workshop on Computational learning theory, ACM, 1992, pp. 144–152.

[7] T. BRANTS, A. C. POPAT, P. XU, F. J. OCH, AND J. DEAN, *Large language models in machine translation*, in In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Citeseer, 2007.

[8] C. CALLISON-BURCH AND M. DREDZE, *Creating speech and language data with Amazon's Mechanical Turk*, in Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, Association for Computational Linguistics, 2010, pp. 1–12.

[9] E. CAMBRIA, B. SCHULLER, B. LIU, H. WANG, AND C. HAVASI, *Knowledge-based approaches to concept-level sentiment analysis*, IEEE Intelligent Systems, 28 (2013), pp. 12–14.

[10] ——, *Statistical approaches to concept-level sentiment analysis*, IEEE Intelligent Systems, 28 (2013), pp. 6–9.

[11] J. CHANG AND D. M. BLEI, *Relational topic models for document networks.*, in AIStats, vol. 9, 2009, pp. 81–88.

[12] F.-R. CHAUMARTIN, *UPAR7: A knowledge-based system for headline sentiment tagging*, in Proceedings of the 4th International Workshop on Semantic Evaluations, Association for Computational Linguistics, 2007, pp. 422–425.

[13] K. CRAMMER AND Y. SINGER, *On the algorithmic implementation of multiclass kernel-based vector machines*, Journal of machine learning research, 2 (2001), pp. 265–292.

[14] K. CRAWFORD AND M. FINN, *The limits of crisis data: analytical and ethical challenges of using social and mobile data to understand disasters*, GeoJournal, 80 (2015), pp. 491–502.

[15] D. DEMNER-FUSHMAN, W. W. CHAPMAN, AND C. J. MCDONALD, *What can natural language processing do for clinical decision support?*, Journal of biomedical informatics, 42 (2009), pp. 760–772.

[16] R. DESCARTES AND J.-A. EMERY, *Pensées de Descartes sur la religion et la morale*, Adrien Le Clere, 1811.

[17] L. E. DOSTERT, *The Georgetown-IBM experiment*, Machine translation of languages. John Wiley & Sons, New York, (1955), pp. 124–135.

[18] P. C. W. DR. LEANNE TOWNSEND, *Social media research: A guide to ethics*. http://www.gla.ac.uk/media/media_487729_en.pdf.

[19] V. D. W. B. . K. C. DUGDALE, J., *Social media and SMS in the Haiti earthquake*, https://www.researchgate.net/publication/241624291_Social_media_and_SMS_in_the_Haiti_Earthquake, (2012).

[20] J. ELLIS, *The new york times built a robot to help make article tagging easier*, Nieman Lab, (2015).

[21] FALKIRK COUNCIL, *Environmental baseline report*. https://www.falkirk.gov.uk/services/planning-building/planning-policy/local-development-plan/docs/strategic-environmental-assessment/revised-environmental-report/1%20REP%20Appendix%201%20-%20Environmental%20Baseline%20Report.pdf?v=201406231453, 04 2013.

[22] ——, *Settlement population estimates*.

https://www.falkirk.gov.uk/services/council-democracy/statistics-census/docs/census/2011/6%20Settlement%20Population%20and%20Household%20Estimates.pdf?v=201406020914, 11 2013.

[23] FALKIRK HERALD, *Good weather leads to complaints about Polmont landfill site*. http://www.falkirkherald.co.uk/news/environment/good-weather-leads-to-complaints-about-poln 04 2012.

[24] J. FERRERO, F. AGNES, L. BESACIER, AND D. SCHWAB, *Compilig at semeval-2017 task 1: Cross-language plagiarism detection methods for semantic textual similarity*, arXiv preprint arXiv:1704.01346, (2017).

[25] FORTH PORTS, *Port of Grangemouth*. https://forthports.co.uk/grangemouth/, 04 2017.

[26] 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, Association for Computational Linguistics, 2004, p. 841.

[27] P. L. GARVIN, *The Georgetown-IBM experiment of 1954: an evaluation in retrospect*, Mouton, 1968.

[28] M. GHIASSI, J. SKINNER, AND D. ZIMBRA, *Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network*, Expert Systems with applications, 40 (2013), pp. 6266–6282.

[29] A. GO, R. BHAYANI, AND L. HUANG, *Twitter sentiment classification using distant supervision*, CS224N Project Report, Stanford, 1 (2009).

[30] A. GODE AND H. BLAIR, *Interlingua grammar*, New York: Storm, (1951).

[31] R. G. GOLDBERG AND R. R. ROSINSKI, *Automated natural language understanding customer service system*, Apr. 20 1999.
US Patent 5,895,466.

[32] B. HAN, P. COOK, AND T. BALDWIN, *Text-based Twitter user geolocation prediction*, Journal of Artificial Intelligence Research, 49 (2014), pp. 451–500.

[33] J. HUTCHINS, *"the whisky was invisible" or persistent myths of MT*, MT News International, 11 (1995), pp. 17–18.

[34] ——, *Two precursors of machine translation: Artsrouni and Trojanskij*, International Journal of Translation, 16 (2004), pp. 11–31.

[35] W. J. HUTCHINS, *The Georgetown-IBM experiment demonstrated in january 1954*, in Conference of the Association for Machine Translation in the Americas, Springer, 2004, pp. 102–114.

[36] C. J. HUTTO AND E. GILBERT, *Vader: A parsimonious rule-based model for sentiment analysis of social media text*, in Eighth International AAAI Conference on Weblogs and Social Media, 2014.

[37] K. C. JACOB METCALF, *Where are human subjects in big data research? the emerging ethics divided*, `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2779647`, (2016).

[38] E. KOULOUMPIS, T. WILSON, AND J. D. MOORE, *Twitter sentiment analysis: The good the bad and the omg!*, Icwsm, 11 (2011), p. 164.

[39] P. KRALJ NOVAK, J. SMAILOVIĆ, B. SLUBAN, AND I. MOZETIČ, *Sentiment of emojis*, PLoS ONE, 10 (2015), p. e0144296.

[40] M. KULSTAD AND L. CARLIN, *Leibniz's philosophy of mind*, in The Stanford Encyclopedia of Philosophy, E. N. Zalta, ed., Metaphysics Research Lab, Stanford University, winter 2013 ed., 2013.

[41] M. LAPATA AND F. KELLER, *Web-based models for natural language processing*, ACM Transactions on Speech and Language Processing (TSLP), 2 (2005), p. 3.

[42] J. H. LAU, T. BALDWIN, AND D. NEWMAN, *On collocations and topic models*, ACM Transactions on Speech and Language Processing (TSLP), 10 (2013), p. 10.

[43] K. LEETARU, S. WANG, G. CAO, A. PADMANABHAN, AND E. SHOOK, *Mapping the global twitter heartbeat: The geography of Twitter*, First Monday, 18 (2013).

[44] P. LIANG, *Semi-supervised learning for natural language*, PhD thesis, Massachusetts Institute of Technology, 2005.

[45] C. LIN AND Y. HE, *Joint sentiment/topic model for sentiment analysis*, in Proceedings of the 18th ACM conference on Information and knowledge management, ACM, 2009, pp. 375–384.

[46] D. LIU AND T. CHEN, *Unsupervised image categorization and object localization using topic models and correspondences between images*, in Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, IEEE, 2007, pp. 1–7.

[47] I. LUNDEN, *TwitterâĂŹs woes continue on Q1 sales of $595M, a sluggish 310M MAUs and weak guidance*, TechCruch, (2016).

[48] R. J. McQueen, S. R. Garner, C. G. Nevill-Manning, and I. H. Witten, *Applying machine learning to agricultural data*, Computers and electronics in agriculture, 12 (1995), pp. 275–293.

[49] S. M. Mohammad, S. Kiritchenko, and X. Zhu, *NRC-canada: Building the state-of-the-art in sentiment analysis of tweets*, arXiv preprint arXiv:1308.6242, (2013).

[50] T. Mullen and N. Collier, *Sentiment analysis using support vector machines with diverse information sources.*, in EMNLP, vol. 4, 2004, pp. 412–418.

[51] National Research Council and Automatic Language Processing Advisory Committee and others, *Language and Machines: Computers in Translation and Linguistics; A Report*, National Academy of Sciences, National Research Council, 1966.

[52] G. Neubig and K. Duh, *How much is said in a tweet? a multilingual, information-theoretic perspective.*, in AAAI Spring Symposium: Analyzing Microtext, 2013.

[53] P. K. Novak, J. Smailović, B. Sluban, and I. Mozetič, *Sentiment of emojis*, PloS one, 10 (2015), p. e0144296.

[54] M. G. Paul S. Earle, Daniel C. Bowden, *Twitter earthquake detection: earthquake monitoring in a social world*, `http://www.annalsofgeophysics.eu/index.php/annals/article/view/5364`, (2014).

[55] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, *Large margin dags for multiclass classification*, in Proceedings of the 12th International Conference on Neural Information Processing Systems, MIT press, 1999, pp. 547–553.

[56] R. K. Plumb, *Russian is turned into English by a fast electronic translator*, New York Times, (1954).

[57] H. Poursepanj, J. Weissbock, and D. Inkpen, *uottawa: System description for semeval 2013 task 2 sentiment analysis in twitter*, Atlanta, Georgia, USA, (2013), p. 380.

[58] M. Sabou, K. Bontcheva, and A. Scharl, *Crowdsourcing research opportunities: lessons from natural language processing*, in Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies, ACM, 2012, p. 17.

[59] G. Shih, *Over 20 million tweets sent as sandy struck*. `http://in.reuters.com/article/storm-sandy-twitter-idINDEE8A10AX20121102`, 11 2012.

[60] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, *Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks*, in Proceedings of the

conference on empirical methods in natural language processing, Association for Computational Linguistics, 2008, pp. 254–263.

[61] S. TAN, X. CHENG, Y. WANG, AND H. XU, *Adapting naive bayes to domain adaptation for sentiment analysis*, in European Conference on Information Retrieval, Springer, 2009, pp. 337–349.

[62] H. THAKKAR AND D. PATEL, *Approaches for sentiment analysis on twitter: A state-of-art study*, arXiv preprint arXiv:1512.01043, (2015).

[63] J. TURIAN, L. RATINOV, AND Y. BENGIO, *Word representations: a simple and general method for semi-supervised learning*, in Proceedings of the 48th annual meeting of the association for computational linguistics, Association for Computational Linguistics, 2010, pp. 384–394.

[64] A. M. TURING, *Computing machinery and intelligence*, Mind, 59 (1950), pp. 433–460.

[65] TWITTER, *REST APIs*.
`https://dev.twitter.com/rest/public`, 2017.

[66] ——, *Twitter milestones*.
`https://about.twitter.com/company/press/milestones`, 2017.

[67] TWITTER BLOG, *Location, Location, Location*.
`https://blog.twitter.com/2009/location-location-location`, 2009.

[68] V. VAPNIK, *Pattern recognition using generalized portrait method*, Automation and remote control, 24 (1963), pp. 774–780.

[69] A. WAGEMAKERS, *Twitter is struggling to reengage its 1 billion inactive users*, Business Insider, (2015).

[70] B. WHITMAN AND S. LAWRENCE, *Inferring descriptions and similarity for music from community metadata.*, in ICMC, 2002.

[71] J. YI, T. NASUKAWA, R. BUNESCU, AND W. NIBLACK, *Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques*, in Data Mining, 2003. ICDM 2003. Third IEEE International Conference on, IEEE, 2003, pp. 427–434.

[72] L. L. ZAMENHOF, *An international language*, 1911.