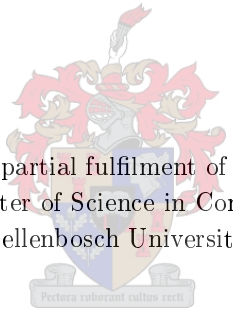
Automatic Prediction of Comment Quality

Dirk Johannes Brand



Thesis presented in partial fulfilment of the requirements of
the degree of Master of Science in Computer Science at
Stellenbosch University.

Supervisor: Prof. Brink van der Merwe
Co-supervisors: Dr. Steve Kroon & Dr. Loek Cleophas

March 2016

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2016

Abstract

Automatic Prediction of Comment Quality

Dirk Brand

Computer Science Division
Department of Mathematical Sciences
Stellenbosch University
MSc. Computer Science
December 2015

The problem of identifying and assessing the quality of short texts (e.g. comments, reviews or web searches) has been intensively studied. There are great benefits to being able to analyse short texts. As an example, advertisers might be interested in the sentiment of product reviews on e-commerce sites to more efficiently pair marketing material to content. Analysing short texts is a difficult problem, because traditional machine learning models generally perform better on data sets with larger samples, which often translates to more features. More data allow for better estimation of parameters for these models. Short texts generally do not have much content, but still carry high variability in that they may still consist of a large corpus of words.

This thesis investigates various methods for feature extraction for short texts in the context of online user comments. These methods include the leading manual feature extraction techniques for short texts, N-gram models and techniques based on word embeddings. The effect of using different kernels for a support vector classifier is also investigated. The investigation is centred around two data sets, one provided by News24 and the other extracted from Slashdot.org. It was found that N-gram models performed relatively well, mostly outperforming manual feature extraction techniques.

Uittreksel

Outomatiese voorspelling van die kwaliteit van aanlyn kommentaar

Dirk Brand

Afdeling Rekenaarwetenskap
Department van Wiskundige Wetenskappe
Universiteit van Stellenbosch
MSc. Rekenaarwetenskap
Desember 2015

Om die kwaliteit van kort tekste (bv. internet kommentaar, soektogte of resensies) te identifiseer en te analiseer, is 'n probleem wat al redelik sorgvuldig in die navorsing bestudeer is. Daar is baie te baat by die vermoë om die kwaliteit van aanlyn teks te analiseer. Byvoorbeeld, aanlyn winkels mag moontlik geinteresseerd wees in die sentiment van die verbruikers wat produkresensies gee oor hul produkte, aangesien dit kan help om meer akkurate bemarkings materiaal vir produkte te genereer. Analise van kort tekste is 'n uitdagende probleem, want tradisionele masjienleer algoritmes vaar gewoonlik beter op datastelle met meer kernmerke as wat kort tekste kan bied. Ryker datastelle laat toe vir meer akkurate skatting van model parameters.

Hierdie tesis bestudeer verskeie metodes vir kenmerkkonstruksie van kort tekste in die konteks van aanlyn kommentaar. Die metodes sluit die voorstaande handgemaakde kenmerkkonstruksie tegnieke vir kort tekste, N-gram modelle en woordinbeddinge in. Die effek van verskillende kernmetodes vir klassifikasie modelle word ook bestudeer. Die studie is gefokus rondom twee datastelle waarvan een deur News24 voorsien is en die ander vanaf Slashdot.org bekom is. Ons het gevind that N-gram modelle meestal beter presteer as die handgemaakde kenmerkkonstruksie tegnieke.

Acknowledgements

I would like to express my sincere gratitude for the following people and organizations:

- my supervisors, Prof. Brink van der Merwe, Dr. Steve Kroon and Dr. Loek Cleophas, for their support and valued inputs;
- Naspers for their financial assistance;
- the MIH media lab for their assistance and provision of an excellent work environment;
- and finally, my fiancée, friends, and family for continued and valued support.

Contents

List of Figures

List of Tables

Acronyms

ANN
artificial neural network.

CBOW
Continuous Bag-of-Words.

kNN
k-Nearest Neighbours.

LDA
Latent Dirichlet Allocation.

LnDA
Linear Discriminant Analysis.

NB
Naïve Bayes.

PCA
Principal Component Analysis.

RBF
Radial Basis Function.

SVC
Support Vector Classification.

SVM
Support Vector Machine.

TF-IDF
Term Frequency - Inverse Document Frequency.

Chapter 1

Introduction

The establishment of bulletin boards in the early days of the internet was an example of the online social interaction that eventually developed into what is now known as the social web [79]. Prominent examples of modern-day online social interaction include social media (e.g. YouTube), information sharing (e.g. Wikipedia) and online communities (e.g. Facebook). All these platforms depend on users to continuously generate, curate and annotate content.

There are now countless online platforms that permit users to generate content. These include forums, blogs, newsgroups and online news providers. One of the key features underpinning the success of these online communities is large-scale user engagement, seen in the form of rating, tagging and commenting on content [79]. User-contributed comments on web content offer a much richer, albeit unstructured, source of contextual information than ratings or tags. However, comments are often variable in quality, substance, relevance and style.

An online news provider is defined in this work as an internet entity that serves original journalistic content to users and then allows the users to engage with that content via comments and/or ratings (e.g. the New York Times). As the social web grows and people become increasingly socially aware [52], online news providers are becoming ever larger communities where users can discuss or comment on common issues in the context of news articles [48]. There are also sites that act as online news aggregators, that serve content either directly from real news sources or from users (e.g. Slashdot.org [42]).

An online news provider may fulfil many different roles, including educating people, providing timeous access to the latest news, and providing feedback to news providers about their content [91].

The importance of the role that online news plays in the media sector (especially when educating and informing people) leads news providers to strive to provide content of high quality, as well as to keep users engaged on

the site for as long as possible. Navigating through the mass of comments on articles to find useful information quickly becomes a daunting and time-consuming task for users. Therefore, to ensure high quality in user-submitted content (such as comments on articles), online news providers attempt to moderate or curate the content. Moderation on websites where discussions are fostered has been a topic of discussion in recent years [29, 50].

1.1 Problem Statement

There are great benefits to being able to analyse short texts – for example, advertisers might be interested in the sentiment of product reviews on e-commerce sites to more efficiently pair marketing material with content. However, analysing short texts is a difficult problem, because traditional machine learning models are generally developed and optimized for longer texts. Longer texts are able to produce denser feature sets for some feature construction techniques which allows for better estimation of parameters for these machine learning models. Short texts generally do not have much content, but still carry high variability in that they may consist of a large corpus of words. Thus, short texts can be characterized as having both little content and being sparse. This makes it hard to build a representative feature space for short texts [36].

There are two dominant approaches to dealing with this problem. The first is to expand the short texts with meta-information (their context, date, etc.) or external larger documents (e.g. by adding the content of the corresponding article to each comment) [148]. The second is to determine a set of topics for the text corpus and assign topics to comments [132]. Depending on the domain and the content, the first approach could be a manual and time-consuming process, but more often the problem is that external texts that fit contextually with the short text are not readily available. When automatically identifying the quality of short texts that are already sufficiently categorized, the second approach is not as well suited. In the context of online comments and forum posts, texts are often in threads (i.e. a tree structure containing texts) attached to some web object (e.g. comments on a news article) where comments may well display similar topic distributions, making classification by topic models less applicable.

The problem of automatically determining the quality of short texts in threads is a previously studied classification task. A number of learning methods have been applied to this problem, including k-Nearest Neighbours (kNN) [68], naïve Bayes [62, 163] and Support Vector Machines (SVMs) [164].

Using SVM-based methods is a particularly popular approach [180] for tasks involving short texts, because SVMs are versatile and can be modified

to work with both dense and sparse data.[1] The kernel used by the algorithm can be customized according to the format of the data (e.g. numerical, string, graph or tree data), allowing for both linear and non-linear classification. The performance of SVM-based methods depend on the choice of kernel as well as the quality of the training data provided [80]. The SVM kernels considered in this work cannot be trained as-is on textual input data. The data must first be transformed into features that can be recognized by the specific kernel function (typically numeric features). Thus, most approaches to short text classification focus on improving the quality of the features extracted from the texts [160, 36, 180], with studies comparing the approaches using SVMs with one or two kernels.

This thesis tackles the problem of automatically predicting and analysing the quality of online comments. This problem has been intensively studied since 2007 [132, 73, 89, 60, 178, 28]. We treat the task as a supervised learning problem and evaluate various approaches to feature set construction for multiple data sets, of which two were provided by News24 and a third obtained from Slashdot.org.

The leading feature extraction approaches for short texts include manual feature construction and models based on **N-grams** [180, 97, 32]. More recently, deep learning approaches to feature extraction have been proposed for various machine learning tasks, including text classification for short texts [159, 56]. This work considers these approaches in the context of predicting the quality of online comments.

1.1.1 Comment Quality Prediction

This investigation into techniques for automatic quality prediction of short texts was motivated by a data set provided by News24 (a subsidiary of Media24), a popular South African news provider that serves news articles to a mainly South African audience. They wished to replace their current article comment moderation system with a more sophisticated one, aiming at two goals, which shaped the nature of this study. First, they wanted to improve the general quality of commentary, so as to legitimise the website content and to better establish themselves as a world-class online news provider. This would have hopefully increased their user engagement. Second, they hoped to maintain the current user engagement levels of the website. The problem was that certain highly engaged users were aggressive, defamatory, and generally displayed malevolent behaviour that negatively affected the 24.com brand as well as the users of the site. After providing the data, but before the completion of this study, News24 revised their comment policy, permanently disabling all comments on all articles from the 11th of September, 2015 [172]. Their decision further illustrates the extent and severity of the

[1] The techniques for feature extraction used in this thesis produces both dense and sparse data setss.

problem – as the News24 editor-in-chief said in their official statement on the matter [172]: "Our decision to change our comments policy follows months of internal debate and discussion which has seen us consider all options practically available to us on how to wrangle the thousands of comments which are made on 24.com each day."

Before this policy revision, News24's system ranked comments by date only (oldest comments first), and allowed users to flag comments that they believed to be against News24's terms of use. A team of moderators then manually reviewed flagged comments, either unflagging the comment (and disallowing future flagging of the comment), or removing it permanently from the site. This was a labour-intensive and imperfect system, since moderators were often biased in their moderation [157]. Therefore, News24 wanted to reduce the effort and time required by editors to moderate the comments, or alternatively removed the need for editors completely.

To compare the quality prediction models for News24 comments, another set of comments was collected from a more regulated online news aggregator website called Slashdot [42]. Slashdot allows users to post links to news and other articles from other source websites, including online news providers. Users are then able to comment on these articles, either as a registered user or anonymously. Their moderation goals are similar to News24's: they wish to promote quality comments, make their content as readable and accessible as possible and do this all in an efficient way that minimizes the time required by any single moderator. They designed an automatic moderator management system to achieve these goals, which is further discussed in Chapter 2.

The case of Slashdot is fairly similar to the News24 case. News24 also has a vast volume of comments to filter and a small team of editors to it, so they have to rely on users to report malicious comments that they can then manually remove. The two data sources differ in that the Slashdot comment corpus generally contains well-formed comments with less colloquial or regional language usage than with the News24 comments. News24 is known for appealing to a broad audience, being a general news provider, whereas the Slashdot community is generally more homogeneous and often more educated, since the type of news that is posted to Slashdot is of a technical nature. The scoring mechanism used by Slashdot (as discussed in Section 1.2.3) also provides a more fine-grained and subtle measure of quality than the flagging system used by News24.

The moderation approaches used by Slashdot could have served well for a news provider like News24, but the system would still rely on an initial comment score that would need to be automatically determined, after-which a comment would be moderated over time, delaying its "stable" score. Thus, both News24 and Slashdot could benefit from a system that performs completely autonomous moderation of comments as they arrive.

1.1.2 Research Question

The quality of online comments, as investigated in this thesis, is related to the moderation schemes defined for the data sets used in the investigation. Poor quality comments could often simply be spam, which is why we use a simple spam detection model as a baseline approach to measure the performance of our approaches. The leading manual feature construction approaches, as motivated by the research literature about online comment classification, is also investigated. As a comparison to these techniques, N-gram-based models and techniques from distributed representation models are also investigated.

Thus, this thesis aims to answer the following research question:

How do feature construction techniques based on N-gram models and distributed representation models fare against leading manual feature construction approaches?

Research Objectives

To address this research question, the following research objectives were identified:

1. investigate and implement various leading approaches to manual feature construction for online comments;
2. investigate and implement N-gram-based models for building representative feature sets for comments;
3. investigate and implement word embeddings as a representative technique for distributed representation of text; and
4. evaluate the predictive performance of these approaches against a baseline pre-trained spam detection model.

1.2 The Data

Three data sets will be used to contextualise and answer the research question posed in this thesis. The first data set is extracted from a collection of databases provided by News24. The second data set was obtained by having News24 staff manually classify (according to pre-set criteria) comments extracted from the databases originally provided by News24. The last data set was automatically extracted (via web-scraping) from the Slashdot website, since a suitable existing Slashdot data set could not be found. The data sets are discussed in more detail below.

1.2.1 News24 Basic Data

News24 allowed its users to leave comments on articles. A user could either directly leave a comment on an article (referred to as a parent comment) or on a parent comment (referred to as a child comment). A parent comment together with all the child comments following it is called a comment thread. Figure 1.1 shows a fragment of a typical comment thread where one user posted a comment and another user commented on that comment. Most articles had multiple comment threads associated with them. Table 1.1 lists some summary statistics of this data set.

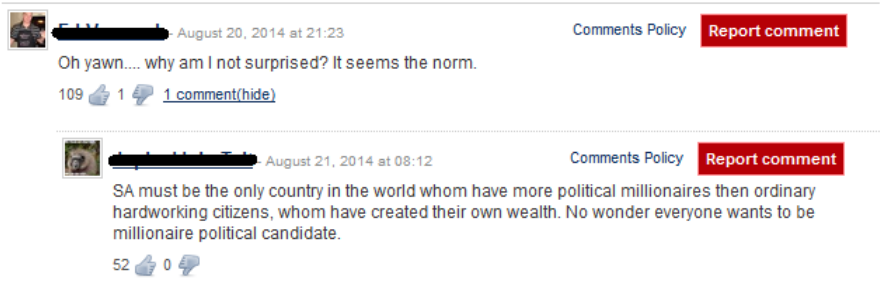


Figure 1.1: Part of a typical News24 comment thread.

Number of comments	130713
Number of parent comments	82325
Number of child comments	48388
Average number of child comments per parent	0.57
Average number of comments per article	22.20
Average number of words per comment	61.45
Percentage of 'hidden' comments	24.6%
Percentage of reported comments	6.6%

Table 1.1: News24 comment corpus statistics.

Users were also able to vote on News24 comments in the form of likes and dislikes, as well as flag comments that they felt were against the terms of use of News24. When flagged, members of the editorial team decided whether the comment should be removed from the site or not. The editorial team also actively reviewed unflagged comments to determine whether they should be removed. A simple automatic system (i.e. without the need for user reports or manual editorial effort) is also in place to detect and remove bad comments via obvious hot-word signals. Comments that are removed receive a "hidden" status, but are still stored in the database. If a comment was not removed, it received the default "visible" status.

After a comment is flagged, editors decided to remove the comment (i.e. make it "hidden") based on whether:

Unreported and visible	63.26%
Unreported and hidden	15.19%
Reported and visible	12.81%
Reported and hidden	6.06%
Automatically hidden (hot-words)	2.67%

Table 1.2: Distribution of comments according to their report, hidden and hot-word labels.

- it contained advertising or spam;
- it contained abusive language, hate speech or profanity;
- it contained completely incorrect grammar;
- it included text-speak;
- it included nicknames or insulting names for the president or the individual(s)/group the article is about; or
- it referred to racial stereotypes or contains racial slurs.

Thus, there are six implicit categories of comments in the News24 data set:

1. unreported comments unseen by editors;
2. unreported comments reviewed by editors and accepted by editors;
3. unreported comments reviewed by editors and made hidden by editors;
4. reported comments that were accepted by editors;
5. reported comments that were made hidden by editors; and
6. hot-word comments that were immediately automatically made hidden.

Table 1.2 shows the distribution of comments according to this categorisation. Since the data does not contain any information on whether comments have been seen by editors or not, categories 1 and 3 are collapsed into one category. This is a challenge when classifying the comments, since there might be comments that are being shown that fit the criteria for being made hidden, but have simply not been considered by editors. This problem motivated the necessity for a second data set that carried the guarantee of each comment having been seen by a member of the editorial team. This data set is further discussed in Section 1.2.2.

This thesis focusses on the task of predicting the "hidden vs. visible" status for unlabelled comments (typically newly posted comments) automatically, i.e. to classify a comment to reflect the opinion of the editors. A

secondary task, that is not investigated, is predicting whether a comment would eventually be reported or not, i.e. classify a comment to reflect the opinion of the users. The task of automatically finding hot-word comments has been completed by News24 and is not of interest in this thesis, thus these automatically removed comments are not included in the data sets used for this thesis.

Data Description

News24 provided two SQL databases, both containing data and meta-data about comments. One database contains unlabelled comments that were accumulated over the last six years, and another labelled comments that range over a single month. The database of unlabelled comments forms the basis for the large News24 data set, which we will henceforth refer to as `News24-large`. The second, smaller, labelled data set was generated from the large labelled data set, as described in Section 1.2.2, and will be referred to as `News24-small`. The large database of unlabelled comments was used to train the topic models and the deep learning models, as discussed in Chapter 3.

The labelled database contains the following relevant fields for each comment: a unique comment identifier, a unique author identifier, a unique thread identifier, the id of the parent comment (or `null` if a parent comment), the name of the author, the body of the comment, the title of the article, the body of the article, the number of reports, the comment's "hidden" status, and the date and time the comment was posted.

The unlabelled database contains the same fields, with the exception of the hidden status.

1.2.2 Small News24 Data Set

In an attempt to get a better labelled data set where there is no ambiguity on whether comments have been seen or not, a subset of the comment corpus was presented to the News24 editorial team for rating. For this task, an alternative rating scheme was proposed in an attempt to get more meaningful flags for comments.

The comments were each labelled into categories numbered from 1 to 3, with 1 representing a very low quality comment, 2 representing a comment that is suitable for the website, and 3 representing a remarkably sensible comment. The team decided internally that a comment was to be explicitly labelled as 1 if it:

- contained text-speak;
- consisted mainly of capital letters;
- included profanity;

Number of comments	4796
Number of parent comments	2943
Number of child comments	1853
Average number of child comments per parent	1.48
Average number of comments per article	30.94
Total number of words	±300000
Average number of words per comment	67.85
Ratings distribution	1=44%, 2=52% , 3=2.4%

Table 1.3: Statistics about the small News24 comment set.

- was not relevant to the topic;
- included the text "cANCer" [2];
- contained insulting nicknames for the president or the individual the article is about;
- attacked another commenter;
- contained racist or abusive language; or
- referred to racial stereotypes.

Table 1.3 shows some more information about the data sets and the distribution of the class labels. Since so few comments were labelled as 3, the labels of 2 and 3 were merged to split the corpus into two classes: acceptable or not acceptable for the live website. The resulting data set is thus similar to the large News24 data set, but with the added certainty that all the comments were reviewed by editors.

1.2.3 Slashdot.org Data

The Slashdot model is quite different to that of News24. Their articles are all sourced from other websites with links posted on Slashdot by the users. Users are also able to comment on these posts, as well as comment on other comments, forming a comment tree of limitless depth, unlike News24 where the comment thread depth is limited to 2.

Comments are given integer scores from -1 to 5, where 5 is the highest score. Readers of the site can then set a score threshold for the comments to be displayed to them, effectively hiding all posts below some threshold (if set). Some posts are also tagged by both moderators and users to further help readers identify posts that they wish to read, as well as allow users to

[2] This is a slur referring to the leading political party in South Africa, the African National Congress (ANC).

Figure 1.2: Part of a typical Slashdot comment thread.

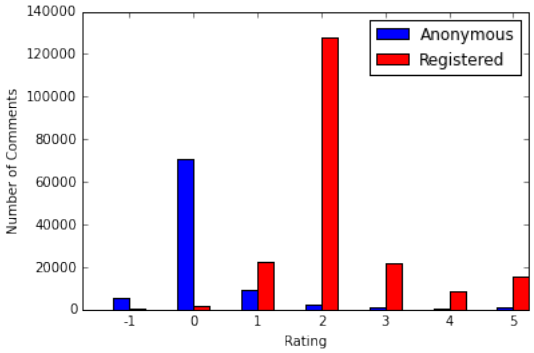


Figure 1.3: Distribution of comment scores on Slashdot.org.

categorise posts. The tags can be any word, but some common tags are typically used, such as "funny", "informative", "troll" and "flamboyant".

Registered users' comments are generally seeded with a score of 1 (although this score could also be 0 or 2, based on the ratings of their previous posts, also called the user's *karma*), while anonymous users' comments begin with a score of 0. Once a comment is seeded with a score, Slashdot automatically assigns moderation privileges to certain users (willing participants that are registered, regular users with a positive karma), allowing them to modify this comment score. Figure 1.2 shows a typical Slashdot comment with a reply. At the time, the original comment received a score of 0 (probably the default score, since the post was made by an anonymous user), but the reply received a higher score of 2 (also probably due to the poster being a registered user with a high karma). As with the News24 comments, there is no way to clearly determine whether a comment has been moderated, however Slashdot's moderation scheme assigns moderator privileges to users according to the demand of unmoderated comments. The moderation scheme is discussed further in Section 2.1.

The distribution of ratings in the data sets is shown in Figure 1.3. Table 1.4 shows some statistics about the Slashdot comment corpus that was extracted.

Number of comments	289215
Number of articles	2231
Average number of comments per article	129.63
Number of anonymous comments	91057
Number of registered comments	198158
Average number of words per comment	84.41

Table 1.4: Slashdot comment corpus statistics.

Data Extraction Methodology

The Slashdot comments were obtained by means of parsing web data (i.e. web scraping). An open source `python` library called `Jsoup` [71] was used to access a URL containing an archive of all Slashdot articles, arranged from most to least recent (`http://slashdot.org/archive.pl`). The archive page contains links to all articles published on Slashdot. Using `Jsoup`, each link was followed to its corresponding content page containing the article as well as all the comments related to that article. From each comment, the following information was extracted: the comment content, the comment's current score, the date and time the comment was posted, the username of the comment's author (or "anonymous"), the comment's immediate parent comment, as well as the comment at the root of the thread.

This process was executed on 2 August 2015 and allowed to run until the comments of the 2231 most recent articles have been captured.

1.2.4 The Unbalanced Data Problem

With the large News24 and the Slashdot data sets, the class value is unevenly distributed among the data points — specifically, one class label clearly dominates the other data points in the data set. With the News24 data sets, only about 25% of the comments have been made hidden by editors. With the Slashdot data sets, the effects are more severe, with approximately 45% of comments having a score of 2 (largest class) and only 2% of comments having a score of -1 (smallest class).

With traditional measures of accuracy, a classifier can achieve inflated accuracy by simply classifying all the data points as the majority class [173] (about 75% in the case of the News24 data). To address this, other assessment measures such as sensitivity and specificity are also used in this thesis [34].

Attempts have been made to deal with this unbalanced data problem in other domains such as fraud detection [57, 22], where the data is often considerably more unbalanced. Two common types of solutions to this problem exists, viz. cost-based and sampling-based solutions [133]. Since we have no information about the cost of predicting certain classes above others, a cost-

based solution would not be well-suited. We use a sampling-based approach for dealing with class-imbalance in the Slashdot data set (as discussed in Section 5.3).

1.3 Thesis Overview

Chapter 2 contextualizes this thesis with previous approaches to similar problems in the research literature, and provides background on the techniques used in this thesis. Chapter 3 lists and discusses the construction of various domain-specific features based on research by other authors, as well as how features were extracted using N-grams, basic topic modelling techniques, and deep learning. Chapter 4 presents the research methodology, followed by experimental results in Chapter 5. Finally, Chapter 6 summarizes the findings of this thesis and discusses possible future work to continue this research.

Our results show that N-gram models generally perform better than manual techniques for feature construction, with character N-grams and skip-grams showing great promise. We are also able to confirm that word embedding models based on deep learning techniques are able to perform as well as the N-gram-based approaches in some cases.

Chapter 2

Background

This chapter discusses supervised learning techniques for quality prediction in short texts.[1] Background information on N-gram models, topic modelling and deep learning are also presented. Section 2.1 details some moderation schemes used by websites that make use of user-contributed content. Section 2.2 provides an overview of the research literature regarding automated quality prediction for short texts. Section 2.3 contextualises the task of quality prediction in the field of machine learning and provides background on the Naïve Bayes (NB) and SVM algorithms used to approach this task. The sections that follow discuss the background of the techniques used for feature extraction in this thesis. Section 2.4 provides a background for N-gram-based techniques for feature extraction. Section 2.5 discusses topic modelling and how it is used for short texts. Finally, Section 2.6 provides background about the proposed deep learning techniques used in this thesis.

2.1 Existing Moderation Schemes

There are various existing moderation schemes that seek to filter and moderate user-submitted contributions based on their subjective quality (i.e. what the specific moderators define as quality). This section discusses some well-known examples of content moderation systems, namely those of Digg.com, Wikipedia.org, Slashdot.org and Reddit.com, as well as two common moderation tools, namely Debate and Disqus.

In 2004, Digg.com [49] was launched. Digg is an aggregator of online news content, curated by users and presented in a concise fashion. The content consists of articles and stories from various domains. Users can "vote for" content, giving an article an up-vote if the user liked it, increasing its *Digg score*. The Digg score allows the website to rank articles and to better filter information before presenting it to a user. The site also has several internal

[1] This chapter contains portions of work from previous papers co-authored with one or more of my co-supervisors [25, 26].

moderators who attempt to determine whether an article's Digg score is fair and accurate. If not, they can adjust the score to provide a more suitable ranking for an article. This is an example of a system that uses *centralised moderation* or supervisor moderation (where the moderation is carried out by internal moderators and not users). The users are only able to vote on content, not to explicitly moderate it.

An alternative moderation scheme is *distributed moderation* (where the majority of the moderation is carried out by users). There are various examples of distributed moderation, of which the most well known is Wikipedia [184]. Users can post verifiable content (verified by means of citations) and edit or remove content posted by other users. There are also bots for automatic detection of abuse by users, such as the system developed by Santiago M. Mola-Velasco [122]. This platform of distributed content provision and moderation has led to Wikipedia becoming the world's largest online encyclopedia [18]: It has more than 100,000 active unpaid volunteers and encyclopedia entries in more than 270 languages [18]. Their status as a free and open source encyclopedia is only possible because of the volunteers and the fund raising they are able to do. This is directly influenced by their user engagement, as this is what attracts volunteers and funders. Various publications mention the efficiency of Wikipedia's moderation scheme [47, 185].

Other examples of smaller-scale distributed moderation are Slashdot [42] and Reddit [123]. Slashdot's moderation scheme first assigns a seed rating to a comment, followed by continued moderation by users with moderator privileges (see below). All users in the system have "karma" representing their reputation on the site, with a positive karma value considered to indicate good reputation. A comment's seed rating is determined by the poster's karma level. The rating changes as moderators choose to increase or decrease the rating of a comment.

Slashdot's system automatically assigns moderator status to certain users. These users are chosen based on the following factors: whether they are a registered user, whether they regularly consume Slashdot content, whether they have been active for a certain period of time, and whether they are positive contributors themselves.

When a user is given moderator status, they are provided with a number of points of influence that they can use to moderate comments. Each comment they moderate deducts a point and when their points have run out, they lose their moderator status until they are automatically asked to moderate again. Moderators are also not allowed to participate in the same discussion that they are moderating. A problem with using human moderators, is that comments are often not immediately moderated, but rather when the moderator finds time to do so. Another problem is that moderators often do not find consensus on the rating a comment deserves.

Reddit uses a very structured system of distributed moderation. Reddit consists of multiple individuals forums, called *subreddits*. Each subreddit

has one or more moderators (often the subreddit's creators) assigned to it to control the content that is posted to the subreddit. A moderator has full control of the content of the subreddit, which is motivated by their interest to maintain the vision and mission of the subreddit. Moderators are free to remove content, approve content that was erroneously removed, distinguish whether items are "safe for work" or not, and change the titles of posts. They cannot, however, edit submissions or see any personal details of a user that submitted content.

Comments on Reddit are either moderated by the community via a reporting scheme similar to that of News24, picked up by automatic spam filters, or manually assessed by moderators. Users are able to report comments for being against the rules of the particular subreddit — some subreddit rules are very strict whereas others have more of an "anything goes" policy regarding content. These comments are then shown to the moderators of the subreddit for permanent removal or reinstatement. Reddit also allows users to provide up and down votes for comments, which serves as way to filter comments that the community prefers.

On the surface, a system such as the moderation scheme used by Reddit works well, because moderation effort is distributed to users, instead of having a single site-wide point of moderation. This allows individuals to use forums that interest them, even though they may be controversial or run counter to public opinion, without the risk of having their comments removed. This does not always happen though, as moderators of subreddits may become tyrannical in their moderation practices, since they might be the sole moderator of a subreddit, which might discourage participation in the particular subreddit.

Various freely available tools exist for online news websites. Debate [21] is a website plugin for Wordpress that gives comment management functionality to blog administrators. It provides a sophisticated structure of comment threads, email-based replies, profiles for commenters and various widgets for the administrator to view statistics about the comment base. It uses a distributed reputation system based on comment quality ranking. Each registered user has a reputation score based on the average quality of their comments, which is in turn determined by the number of up-votes and down-votes their comments receive, as well as the length of the comment and the time the comment was posted. They do not do any additional language filtering of the comments beyond spam filtering [6, 107].

Disqus [165] is a free system that can be integrated into various web platforms (Wordpress, tumblr, Blogger, Drupal, etc.). The system provides a moderation tool to their customers that allow them to remove comments or mark them as spam. The actual filtering and ranking of comments happens on the Disqus servers and are mirrored to the site it is integrated with.

In summary, all these systems require manual moderation, although some systems provide initial seed ratings before manual moderation commences.

2.2 Literature Review

This section reviews various publications that parts of this thesis are based on. Some of these publications focus specifically on quality prediction in online comments, while others address more general text classification tasks.

2.2.1 Slashdot: Peer Moderation

Lampe and Resnick [97] attempted to study the efficacy of the Slashdot moderation scheme. They asked the question: "Can a system of distributed moderation quickly and consistently separate high and low quality comments in an online conversation?" Their analysis shows that the basic idea of distributed moderation works on Slashdot. After enough time, moderators seem to find some consensus, even though complete agreement is not achieved. The remaining problem they identified, was that good quality comments take too long to be identified by moderators.

Lampe and Resnick also noted that general user satisfaction diminishes as more users participate in a conversation space. Some users display disruptive and anti-social behaviour that reduce participation of other users in online conversations. They investigated various methods of limiting the disruptive effect of anti-social behaviour. These methods included analysis of usage logs (records of comments and moderations) and conducting interviews with moderators to get explanations of phenomena in the comments base. They examined the distribution of comment scores and observed a strong correlation between different levels of user participation and comment scores. They were also able to determine both the median time until a comment receives its first moderation, as well as the time until half of the comments that will eventually receive low (0 or less) or high (4 or greater) scores have been scored, viz. 83 minutes and 148 minutes respectively. Similar methods were investigated in other studies with other data sets (e.g. [64, 169]). Specifically, Szabo et al. [169] were able to show that most Digg stories reach their final stable popularity score within one day.

2.2.2 User Reputation

Interactions between users in an online social environment can be informative and productive, but also destructive. Although interaction on the internet is often characterised by anonymity, maintaining and using some form of reputation system for users has been shown to add value to both users and platform providers [140].

Resnick et al. [140] noted that a reputation system allows collection and aggregation of information about participants' past behaviour, while still allowing participants to remain anonymous. Users can base their interaction with other users of the system based on those users' reputations. This

encourages trustworthy behaviour and deters untrustworthy users. Thus, reputation systems are a way of building trust online [66].

Chen et al. [35] distinguishes between *internal* and *external* reputation scores.

External scores are made entirely public and can be used as an incentive for users to produce good content. Yahoo! Answers shows the points that a user has earned for providing good answers to questions, thus providing some measure of user reputation. Internal reputation, on the other hand, is never revealed to users and is only used in internal applications.

Reputation is often used for the following applications [35]:

- **Ranking of content** — User-contributed content can be ranked or recommended based on the poster's internal reputation;
- **Enriching existing content** — A site may want to show tweets (or other social media posts) of reputable users that relate to an article, as a means to enrich the article content;
- **Peer Moderation** — To moderate comments or other user-contributed content on a site, certain reputable users could be given moderation privileges (e.g. StackOverflow [162]). Slashdot also makes use of this in their "karma" model; or
- **Abuse Detection** — The reputation scores of users could be used as additional features in an abuse detector (since reputable users are typically less likely to abuse the system).

This thesis will consider the use of internal reputation for *ranking of content* and *peer moderation* when user-based features are constructed.

In 2007, Chen et al. [38] introduced a reputation model for users in a question and answering (QA) system. Their model combines traditional user ratings [141] (positive, neutral or negative votes towards a user) with an analysis of the QA social network.

They also proposed the idea of constructing a graph of user interaction (i.e. a sociogram where nodes are users and edges are interactions between users) and weighting the edges by the reputation of the users involved in the relation. Users with a higher reputation will affect other users' reputations to a greater extent, whether positively or negatively. This is similar to how PageRank [129] determines the relative importance of nodes in a network (e.g. web pages, articles or users).

2.2.3 Automatic Scoring and Prediction

Wanas et al. [180] attempted to extend the work done by Weimer et al. [183] and Lampe and Resnick [97]. They wanted to improve on the various techniques used by these authors.

Weimer et al. investigated methods for classifying forum posts. They proposed a set of features that addressed some known issues with classifying forum posts (e.g. the short average length of posts). The proposed features ranged from surface features (e.g. capitalized word frequency) to more complex linguistic features (e.g. relevance). They then designed and trained a classifier to classify comments as 'bad' or 'good'.

Similar to the study by Lampe and Resnick [97], Wanas et al. [180] investigated the moderation schemes used on Slashdot. The moderation scheme that Slashdot used during this study (and still uses) was somewhat dependent on human input, and as Wanas et al. noted, a significant amount of time needed to pass before users were able identify good quality comments. Additionally, earlier posts received more attention and posts that received an incorrect seed rating (or early moderated rating), often did not have their rating changed. Wanas et al. proposed a scheme of automatic post ranking based on a set of features given to a classifier (SVM classification with a Radial Basis Function (RBF) kernel). Similar work was done by Hsu et al. [79], but using support vector regression (SVR).

Wanas et al. considered the features originally proposed by Weimer et al. [183] and constructed a set of 22 features, categorised into five classes, viz. relevance (the appropriateness of posts in their respective threads), originality (the novelty of posts compared to others in their threads), forum-specific (various measures of the level of discussion a post evokes), surface (how well the contributor presents their post) and posting component (e.g. presence and quality of weblinks in posts) features. The forum-specific features were shown to contribute most to the accuracy of the post ratings, but consisted of complex linguistic features that rely on posts consisting of higher quality English, which is not necessarily the case for Slashdot. They overcame this problem by building features conscious of linguistic phenomena in online forum posts and by building a lexicon of keywords that were domain-specific.

Contrary to Weimer et al, Wanas et al. used slightly finer ratings for posts. The experiments by Wanas et al. showed their classifier to be 50% accurate when classifying posts as bad, average and good (according to their predetermined partitioning of the Slashdot posts' scores into these three classes).[2] Their experiments also showed that structural features (length, punctuation, etc.) of posts were more significant in classification than features analysing the actual text (spelling, grammatical quality, etc.). We found that certain features based on user activity (e.g. number of posts of the user or the number of comments the user has made in the past, or out degree) emerged as important features (see Section 3.5).

Hsu et al. [79] studied similar methods for predicting the quality of comments on Digg [49]. Instead of simply predicting a comment's score, they

[2] We were able to obtain similar, but slightly better, results. For a full description of the results, see Section 5.3.

attempted to rank comments based on predicted scores. They used a SVR model with an RBF kernel. They focused their efforts on finding the relative rank of a comment, as opposed to the actual value. They compared their ranking to a random ranking and a date-wise chronological ordering. They were able to achieve a much higher ranking correlation score than the random or chronological orderings.

Cheng et al. [39] investigated the communities on three international news sites, viz. CNN.com, Breitbart.com and IGN.com. They attempted to predict a banned status for users (as opposed to a score or rating of the user's comment). They categorised features into four groups, namely post features (concerned with the literal content of posts), community features (popularity in the community as shown by votes), activity features (the user's general patterns of use) and moderator features (number of posts that have been deleted, etc.). They attempted to use these features to predict which users will become banned. Unsurprisingly they showed that moderator features contribute the most to the performance of the classifier. The community features were a close second, which showed that community moderation was closely aligned to the editors' preferences.

Mishne and Glance [119] did a comprehensive study on online comments and built a binary decision tree classifier with a custom feature set similar to the features used in this thesis. They achieved a mean F1-score of 0.88 for 10-fold cross validation. Similarly, Brennan et al. [27] achieved an overall precision of 0.82 when classifying Slashdot comments with binary class labels, using a SVM classifier with 10-fold cross validation. Jamali and Rangwala [83] investigated various algorithms for comment classification, and they were able to obtain a 0.84 F1-score on binary classification using SVM-based methods with 5-fold cross validation. We were able to achieve an accuracy of 0.874 and a sensitivity (or recall) of 0.818 when evaluating a binary classifier on News24 comments, which is comparable to the results achieved in the literature. For more information, see Section 5.2.

Otterbacher [128] suggested an alternative approach to the community rating system employed by Lampe and Resnick [97]. Instead of rating the 'interestingness' of a post (i.e. how interesting users may find a post), the community rates the 'helpfulness' (i.e. the benefit a user gets from a post). The study was performed on user product reviews from Amazon.com. The study uses measures of post quality developed by Wang and Strong [181]. The framework is comprised of four measures of post quality, the first three of which are relevant in the context of this work:

1. **Intrinsic quality** — Includes believability, accuracy, objectivity and reputation;

2. **Contextual quality** — Includes relevance, timeliness, completeness and sentence/word counts;

3. **Representational quality** — Includes ease of understanding, conciseness and consistence; and

4. **Accessibility** — Concerns how easy it is for users to gain access to information.

The study looks at the correlation between these measures and the average 'helpfulness' ratings of reviews (as provided by users). A simple linear regression model was trained for this purpose and it was able to achieve an R^2 score of 0.4.

The hand-crafted features designed and mentioned in the research above, forms part of the basis for the manual feature approaches studied in this thesis. The features we implement and investigate is further discussed in Chapter 3.

2.3 Supervised Learning Methods

This section discusses the major approaches to supervised learning that we use in this thesis. The SVM techniques specifically make use of the feature extraction approaches outlined in the rest of this chapter. As mention before, this thesis addresses the problem of automatically predicting the quality of online comments, and since we have labelled comments and the goal is to predict this label for unseen comments, this problem is a supervised learning problem.

Predicting the score of a Slashdot comment is a more complex task than predicting a binary label for a News24 comment, as there are seven classes and the classes have an intrinsic order, unlike the "hidden" status of News24 comments. The ideal classifier would be able to predict the correct score among these seven scores, however, as we discuss in Chapter 5, this task proved to be very hard to accomplish, partially due to the class imbalance in the Slashdot data.

Following a series of experiments with various labelling strategies for the Slashdot comments, the most viable option was to remove the comments labelled as '2' and group the comments into two classes (i.e. less than two and greater than two). This makes comparing the results of the News24 data sets and the Slashdot data set simpler, because they follow the same binary classification scheme.

In this thesis, the main supervised learning algorithm that is used for comment quality prediction, is the SVM algorithm for classification. A Naïve Bayes model is also used in the context of a simple spam detection model [161]. These techniques are explained in more detail below.

2.3.1 Naïve Bayes

The NB classifier [145] is a simple probabilistic model. Despite its very simple form, the algorithm has shown good performance in many real world classification tasks [51, 90, 147]. Given an input feature vector $\mathbf{x} = (x_1, ..., x_n)$, the NB model attempts to predict the conditional probability of the input vector having class label C_k, i.e. $p(C_k|\mathbf{x})$. This is known as the *posterior probability* of the class, which can be formulated, using Bayes rule, as

$$p(C_k|\mathbf{x}) = \frac{p(C_k)p(\mathbf{x}|C_k)}{p(\mathbf{x})}$$

where $p(C_k)$ is known as the *prior probability* of the class, $p(\mathbf{x}|C_k)$ is known as the *conditional probability* (or likelihood) of the observation $\mathbf{x}$ given the class, and $p(\mathbf{x})$ is a normalization factor known as the *evidence*. In practice, the only interest is in the numerator of the fraction, since the denominator is the same for all classes for a specific observation $\mathbf{x}$. NB makes the naïve assumption that the features in the feature vector are conditionally independent given the class membership. Thus, the likelihood can be reformulated as follows:

$$p(\mathbf{x}|C_k) = p(x_1, ..., x_n|C_k) = \prod_{i=1}^{n} p(x_i|C_k)$$

which means the posterior probability satisfies:

$$p(C_k|\mathbf{x}) \propto p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

An NB classifier is trained to maximise this posterior probability given an observation $\mathbf{x}$ and K possible classes. This is shown in Equation 2.1.

$$\text{predicted label of } \mathbf{x} \leftarrow \underset{k=1,...,K}{\arg\max}\ p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{2.1}$$

A naïve Bayes classifier trained on bag-of-words representations (as explained in Section 2.4.1) is used in this thesis as a simple technique for finding spam comments in the News24 data set.

2.3.2 SVM

The theoretical basis of Support Vector Machines (SVMs) was laid by Vapnik and Lerner in 1963 [177], and later developed by Cortes and Vapnik to form the family of algorithms we know today [44]. In 1990 [154], the introduction of the kernel trick made SVMs much more versatile and able to model a much wider variety of data. The use of slack variables for classification of data that is not linearly separable, was introduced by Smith in 1968 [158] and

improved upon by Bennett and Mangasarian in 1992, where it was introduced to SVMs [13].

The simplest SVM implementation is a non-probabilistic binary linear classifier which attempts to find a line that can accurately separate data points into two classes. When the classes cannot be separated by a line, a non-linear function (a curve) might be able to, but viewing the data points in a higher-dimensional space and using a hyperplane is more efficient computationally [3, 23, 153]. Specialised kernels (e.g. string kernels [102]) can be used to compare data points directly, without the need for intermediate feature construction.

Support Vector Machines (SVMs) are suited for many supervised learning tasks, because of how widely applicable the algorithm is, as well as for its high fault tolerance. Specifically, SVMs have built-in overfitting protection, making them especially suited for handling high-dimensional input spaces (such as the vector space feature sets, discussed in Section 2.4) [100]. They are also well-suited to handling extremely sparse data sets (e.g. some of the N-gram models, discussed in Section 2.4.1) [92].

SVMs can be trained on discrete (Support Vector Classification (SVC)) or continuous (SVR) labels. The specifics of SVC are discussed below.

Support Vector Classification

Support vector machines for binary classification attempt to construct a hyperplane and to maximise the distance from the hyperplane to the nearest data points. A very simple 2-dimensional example is shown in Figure 2.1.

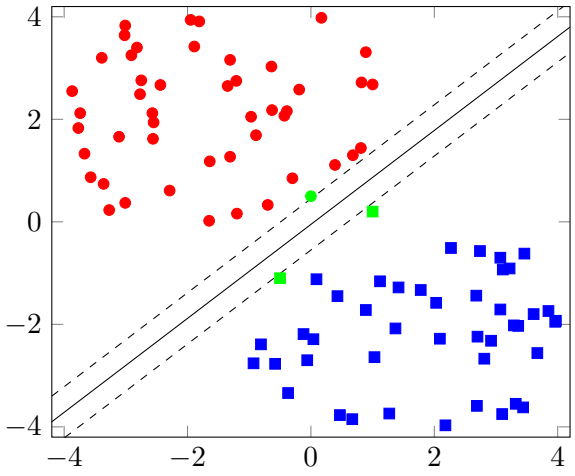


Figure 2.1: Linear SVC of binary class variables. Support vectors are shown as green markers on the margin (dashed line).

Consider a simple binary classification task with training data instances of the form $(\mathbf{x}_i, y_i), i = 1, ..., m$ where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$. The support vector classifier attempts to find an $(n-1)$-dimensional hyperplane that can perfectly separate the data points labelled as -1 from those labelled 1, as in Figure 2.1 where red circles indicate points labelled as -1 and blue squares indicate 1 labels and the distance from the solid to a dotted line is the *margin* (shown as a 2-dimensional tube in Figure 2.1, but it is an n-dimensional tube in the general case). The margin lines (i.e. the dotted lines) are parallel to the hyperplane (i.e. the solid line) and are touching the points nearest to the hyperplane on either sides.

Since a hyperplane is a set of points satisfying

$$\beta_0 + \boldsymbol{\beta}^T \mathbf{x} = 0$$

our aim is to find a hyperplane that separates the training data by finding β_0 and $\boldsymbol{\beta}^T$ such that

$$y_i \cdot (\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i) > 0 \tag{2.2}$$

for all $i = 1, ..., n$.

Continuing from Equation 2.2, to maximise the margin M, SVM optimises the values for $\beta_1, ..., \beta_n$ such that:

$$\sum_{j=i}^{n} \beta_j^2 = 1 \tag{2.3}$$

and

$$y_i \cdot (\beta_0 + \boldsymbol{\beta}^T \mathbf{x}) \geq M, \forall i \in 1, ..., n. \tag{2.4}$$

where M is the width of the margin that is to be optimised. Note that M can be made arbitrarily large by scaling β_0 and $\boldsymbol{\beta}$ if the constraints mentioned above are not enforced. The optimisation problem can be solved using quadratic programming. This procedure finds a hyperplane that maximises the distance to the margin, which is then fixed.

This works for linearly separable data sets, but that is not always the case with real world data. To address this, the strict separability constraint (Equation 2.4) is relaxed to allow for some samples to fall within the margin. A set of non-negative slack variables, $\{\xi_i, i = 1, ..., n\}$ is introduced (one for each training point), together with a penalty allowance C. The optimization problem above now comes with $n + 1$ additional constraints:

$$\xi_i \geq 0$$

$$\sum_{i=1}^{n} \xi_i \leq C$$

which essentially states that the sum of the slack variables may not exceed C.

This problem is still solvable by quadratic programming. Once we have the corresponding weights, a new training sample $\mathbf{x}^*$ can be classified using the hyperplane by calculating the sign of:

$$f(\mathbf{x}^*) = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}^*$$

The magnitude of the value gives an indication of the confidence of the classification (i.e. the further the sample is away from the separating hyperplane, the more confident one can be in the classification).

One key feature of using the maximal margin approach, is that the equation for the hyperplane only depends on the data points that lie directly on or over the margin. These points are known as *support vectors* and are shown as green data points in Figure 2.1. While training the model on a set of training data points of size p, the algorithm only needs to make use of the inner product between data points and not the points themselves, so it can be shown that $f(\mathbf{x})$ can be written as linear combination of inner products:

$$f(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^T \mathbf{x} = \beta_0 + \sum_{i=1}^{p} \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle$$

where α_i is a coefficient corresponding to the i^{th} training sample. When x_i is not an SV, $\alpha_i = 0$, so the algorithm only needs to look at the set of support vectors, say Ω, and not all the data points, so the formula can be rewritten as follows:

$$f(\mathbf{x}) = \beta_0 + \sum_{i \in \Omega} \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle \tag{2.5}$$

This is a major computational advantage, since significantly fewer computations are necessary during evaluation of a trained model.

The above formulae work for linear decision surfaces; however some problems might require a non-linear separating boundary (i.e. not a hyperplane). This is tackled with the kernel trick.

The Non-Linear Case

Linear SVM might get poor accuracy when trying to classify data that is not almost linearly separable. A separating boundary might, however, be found by transforming the set of n features into, say, $2n$ features $x_1, x_1^2, ..., x_n, x_n^2$ and then constructing a hyperplane in this $2n$-dimensional space.[3] This means the non-linear problem in n-dimensions can be transformed to a $2n$-dimensional linear problem, which is easier to solve using the maximal margin approach.

[3] This transformation is a simple example. In general, the data samples just need to be transformed into a space where they are nearly linearly separable.

The kernel trick involves replacing the inner product in Equation 2.5 with a more general kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ which calculates the "similarity" between two vectors (as the inner product does in the linear case). The kernel trick can also be employed during training, since the quadratic programming task can be reformulated in terms of inner products using its dual form. Some popular kernels are:

- Linear Kernel — $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$;
- Polynomial Kernel — $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_k)^d$; and
- Radial Basis Function (RBF) — $K(\mathbf{x}_i, \mathbf{x}_k) = \exp\left(-\gamma ||x_i - x_k||^2\right)$.

The linear and RBF kernels are used in this thesis and their respective performance results are shown in Section 5.2. Illustrations of decision surfaces from the d-degree polynomial and radial basis kernels are shown in Figure 2.2.

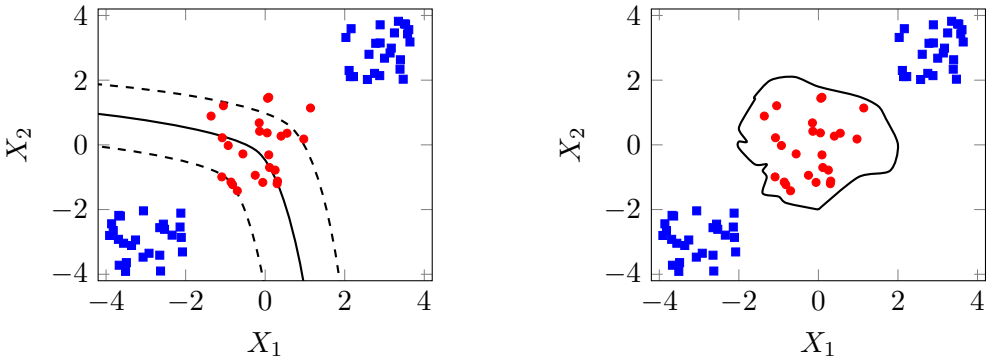


Figure 2.2: Example d-degree polynomial and radial basis function decision surfaces for SVC.

2.4 N-Gram-Based Approaches

One of the approaches to feature extraction that this thesis investigates, is a model based on classic bag-of-words approaches (also called N-gram models). No studies were found that specifically use N-gram-based approaches for quality prediction of internet comments; however N-grams have been used for other related text classification tasks. Some relevant papers are [32, 118, 163]. N-gram models are widely applicable to a variety of problems, not only in information retrieval. These applications include probabilistic language modelling (where words are predicted using N-grams, often useful in translation) [10], DNA sequencing [171] and improved compression algorithms [76].

The models designed by Lampe and Resnick [97], Wanas et al. [180] and other authors interested in automatic comment filtering, used various custom-designed comment features that often incorporate context around the comments and the users that post the comments. Designing and creating a manual feature set is time-consuming, and the features are often highly subjective, meaning that a user could manipulate the system if they knew the features that were being used. It would thus be valuable to be able to extract features in a natural or even automatic way. Also, manual feature construction requires specific domain knowledge, making some of the techniques hard to generalise. Thus, techniques that do not depend on specific domain knowledge are valuable.

This thesis considers alternative approaches to feature construction for representing text data. This section investigates approaches that are taken from common practices in information retrieval [134]. It should be noted that traditionally, information retrieval techniques are designed for working with longer texts, but the techniques are applicable to short texts as well, although the techniques typically result in more sparse data sets.

In information retrieval, a piece of text is often represented by certain keywords or terms. A set of weights can also be associated with these terms to show their relative importance to the text. It is thus sensible to represent the texts in question as a set of term vectors [150]. This idea of text representation is often called the N-gram model [70], a specific type of vector space model for texts [151]. Traditionally, the term *bag-of-words* model implies that features represent single words (or unigrams) whereas the term *N-gram* models generally refer to models where features are N-grams with $N > 1$. In this thesis the term N-gram and N-gram model will be used to refer to models where features are N-grams with $N \geq 1$.

N-gram models have been hugely successful in other natural language processing tasks [118, 55], but their performance on short texts such as internet comments, where the representations will likely be very sparse, is not well-studied. This thesis investigates the use of N-grams for quality prediction of short texts, which has been shown to have some success in the research literature [32]. Various N-gram model representations used in this thesis are discussed in Chapter 3.

2.4.1 N-Grams

Joachims showed that the way text data is represented has a strong influence on the accuracy of a classifier [84]. The assumption that Joachims makes is that using small sequences of words as features, instead of longer sequences of text, allows for greater generalization, since taking the total order of words into account (i.e. not the order within the small word sequence) is often not necessary [5]. Also, better models can often be created since less data is needed for training.

Formally, an N-gram is a series of contiguous objects (letters, words, syllables, or other linguistic units) from a longer piece of sample text. The simplest N-gram representation is the *unigram*, which only considers one object at a time. More interesting models with higher order N-grams (e.g. bigrams and trigrams) are also used.

As an example, consider the sentence "The blue bird flew away", which is composed of the following word N-grams:

unigrams — "The", "blue", "bird", "flew" and "away";

bigrams: "The blue", "blue bird", "bird flew" and "flew away"; and

trigrams: "The blue bird", "blue bird flew", "bird flew away".

Similarly, the word "medal" consists of the following character N-grams:

unigrams: "m", "e", "d", "a" and "l";

bigrams: "me", "ed", "da" and "al";

trigrams: "med", "eda" and "dal";

The general pattern is that a sequence of k words (or characters) will consist of k unigrams, $k-1$ bigrams, $k-2$ trigrams, etc. This thesis investigates both word and character N-grams.

2.4.2 Skip-grams

An N-gram is often taken to be a contiguous sequence, but other co-occurring sets of objects are also used (e.g. the first and last character of words, i.e. skip-grams [40]). Character skip-grams are investigated in this thesis as an alternative to the normal character N-gram model. Skip-grams in the context of the investigations of this thesis, are sequences of letters made from words were certain characters are left out (or "skipped"). This approach will be referred to as *character skip-grams*.

As an example, consider the following character skip-grams for the word "south":

skip-grams: "outh", "suth", "soth", "souh" and "sout".

The motivation for using character skip-grams in this thesis, is that it could help deal with the case of misspelled or obfuscated words being regarded as different entities in the normal character N-gram model. For instance, if someone were to write "loser" as "lo$er", the character skip-gram model would pick up on the fact that the words are similar.

2.4.3 Literature Study

Manevitz and Yousef [106] used similar representations (to those investigated in this thesis) for one-class classification tasks and were able to show that their results were sensitive to the chosen data representation and the kernel they used for their support vector machines. They show that using the simplest binary representation worked better than the more sophisticated Term Frequency - Inverse Document Frequency (TF-IDF) representation [2] (as discussed in Section 3.2.4), which is known to be better for other problems, where certain words appear in most of the sample documents and other, perhaps more discriminative words, occur infrequently (e.g. spam detection models [152]).

Cavnar and Trenkle [32] used N-gram-based techniques for building word frequency profiles for long documents and using these profiles to categorise these documents. They achieved a very high accuracy using the statistical properties of simple character N-grams of lengths 1 to 5, all combined for one model.

Mishne [118] investigated the accuracy of using N-gram approaches for multi-label classification of the mood or sentiment of blog posts. The posts were obtained from Livejournal.com, a free blog service with several million users. The posts were tagged by users with moods from a predefined list of 132 common moods, including "angry", "happy" and "amused". Mishne used various text-based features to augment the N-gram vectors and obtained modest results. Sriram et al. also augmented traditional N-gram approaches (specifically $N = 1$) with their own domain-specific features to categorise texts [163]. They were able to marginally improve the classification accuracy of the bag-of-words approach with features specific to the Twitter domain, including the presence of shortened words, currency, Twitter-like directives (e.g. "@username"), etc.

2.5 Topic Modelling

A common challenge of the N-gram approach to text representation is that the resulting feature vectors are often quite sparse. Dimensionality reduction techniques exist that prune very frequent or infrequent words or N-grams from the model's vocabulary, or even limit the vocabulary to a certain size. This, however, might remove some of the words that characterise a text from the vocabulary, making the representation poorer.

An alternative approach is to group words in the vocabulary together in classes of words that are semantically related, forming "ad-hoc" topics. This is referred to as topic modelling. Topic modelling essentially attempts to determine which recurring patterns of co-occurring words a text corpus consists of. A good topic model would naturally produce clusters of words that make intuitive sense (e.g. "boat", "sea" and "fish" might be in the same

cluster). The topics that occur in a corpus of documents can be used to show similarity between words or documents. When two words are in the same topic, they show a certain degree of similarity, and similarly, two documents that contain the same topics also show some degree of similarity.

Early topic modelling techniques were introduced by Papadimitriou et al [130], but then refined by Thomas Hofmann [77] and called Probabilistic Latent Semantic Indexing (PLSI). A common generalization of PLSI is Latent Dirichlet Allocation (LDA), first introduced by Blei et al [20]. LDA will be used for topic modelling of the comments used in this thesis.

LDA is a generative probabilistic model for discrete data sets, including text corpora. LDA represents a document as a distribution of latent topics [20] where a topic is defined as a distribution over a vocabulary of words. Fundamental to LDA is the assumption that documents consist of multiple words that can be attributed to multiple topics. This is illustrated in Figure 2.4.

Figure 2.3 shows a probabilistic graphical model [94] representation of LDA. The shaded node $W_{d,n}$ represents words in the documents (i.e. the individual comments) that are observed, while the unshaded nodes represent latent variables. Given a vocabulary of size V, each of the K topics is modelled as a multinomial distribution over the V words in the vocabulary (with parameter vector Φ_k). These V-dimensional vectors are calculated using a Dirichlet prior with hyper-parameter β. Similarly, each of the D documents is modelled as a multinomial distribution over the K topics (with parameter vector θ_d) calculated using a Dirichlet prior with hyper-parameter α. Each word in each document is then generated by sampling a topic $Z_{d,n}$ from the topic distribution θ_d of the d^{th} document, followed by drawing a word $W_{d,n}$ from that topic's V-dimensional $\Phi_{Z_{d,n}}$.

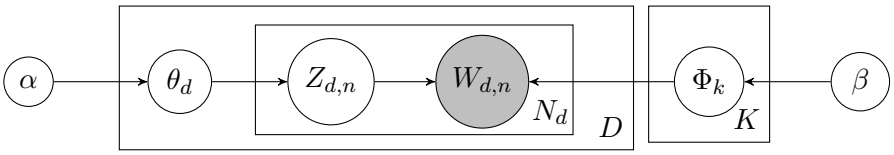


Figure 2.3: A graphical representation of LDA. Nodes represent random variables, edges represent dependencies, plates denote replicated nodes and shaded nodes are observed variables.

The distribution of topics for each document can then be used as a K-dimensional representation vector for a document. We train an LDA model on the News24 and Slashdot corpora. This model is then used to infer a topic distribution vector for each individual comment. These vectors are then used as training data for the supervised learning methods that follow. The details of how the models are trained, are discussed in Section 3.3.

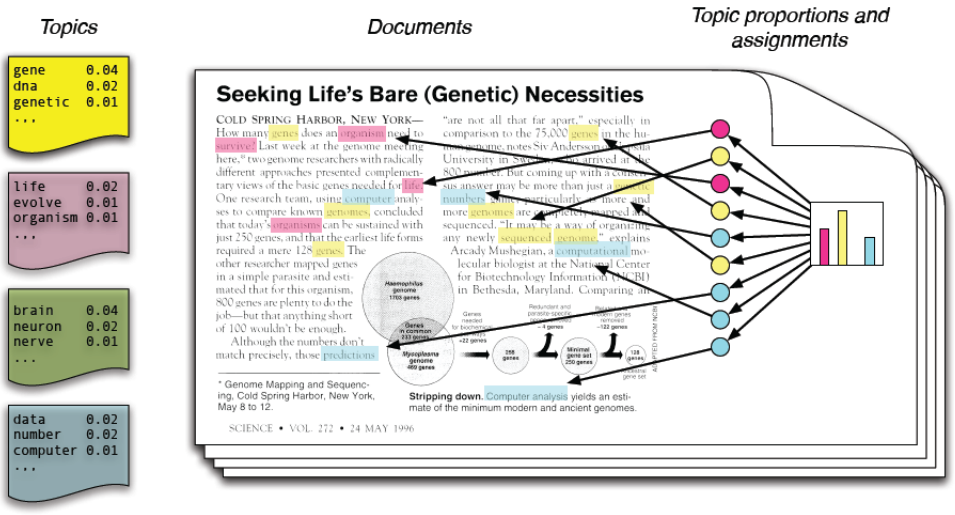


Figure 2.4: An illustration of topic modelling for a piece of text. The various topics that are extracted are coloured separately and on the right a probability distribution shows the topics that the document consists of. Image by Blei et al. [19]

2.6 Deep Learning Networks

Deep learning networks are a specific type of artificial neural networks (ANNs) (only a cursory overview if ANNs are presented here, with a more detailed background given in Appendix B. For further information, see [16].). This section provides background on deep learning techniques and motivation for the use of deep learning for text processing.

In 1989, Hornik et al. [7, 78] famously proved that any continuous function can be approximated arbitrarily well by some feed-forward neural network with a single hidden layer. Realistically, however, a single-layer network is not an efficient approximator of practical functions. Hornik's universal approximation construction essentially allocates a single neuron to every unit (or pattern) of the input space and learns to correctly label each such unit. Unfortunately, the size of possible inputs from the input space grows exponentially as the dimensionality of the input space grows, making Hornik's construction inefficient and not scalable.

Adding depth rather than width to a network can help deal with this explosion in input size. A deep neural network can represent several steps of computation in a function. In fact, a network with three or more hidden layers with trainable weights can classify any arbitrary input space given a reasonable time frame [17]. This line of thinking led to the shift of research into deeper networks.

Deep learning [75] is a modern reincarnation of ANNs, which were first suggested in 1950 by Karl Lashley [98]. Early advances in ANNs were limited by the long time it took to train the networks. This was due to a variety of factors, most notably the lack of hardware able to efficiently train these large networks. Since then, more refined learning methods have been discovered and advances in computer technology have allowed the training of huge data sets in much shorter time frames [124, 135]. Thus, due to the great increases in algorithmic efficiency and computing power, deep learning has become a viable technique and shows impressive performance on classic natural language processing tasks. Research into deep learning became popular due to Geoffrey Hinton who published a paper [75] in 2007 showing how feed-forward neural networks can be pre-trained as individual layers, then linked together, resulting in what he called a "deep neural network".

A big motivation for deep learning is that it enables the learning of features and representations on *unlabelled* data. Most data available today are unlabelled for the tasks that they are used for. Video, audio, images, web pages and documents are generally unlabelled, but often carry rich contextual information that can be useful for various machine learning tasks. Deep learning helps to discover broadly useful representations for such unlabelled data forms. This type of learning is a combination of unsupervised and supervised learning.

A deep learning network has the ability to recognise patterns of increasing complexity, making it an appropriate learning method for any hierarchical data (e.g. text, images, graphs). These types of learning methods are biologically motivated by the way humans learn simple concepts and combine them to form more complex ideas [166]. This feature of deep learning is very valuable in natural language processing tasks, because of the hierarchical structure of language. Language is composed of sentences, which are in turn composed of words, which are simply sequences of letters. Sentences can also be parsed into its phrase structure. This is illustrated in Figure 2.5, where a sentence is parsed to form a syntactic parse tree.

This hierarchy of feature complexity is what makes deep learning so powerful. Deep networks are capable of handling highly complex data sets and automatically identifying distinguishing patterns (or features). In particular, deep learning allows the learning of distributed representations for text [11, 114]. A specialised type of neural network called an autoencoder is often used for feature learning in language models. These language models have been shown to provide significant improvements on a multitude of natural language processing tasks, including dependency parsing [95], named entity recognition [58], sentiment analysis [159], information retrieval [155], machine translation [167] and contextual entity linking [63].

The deep learning methods that are used in this thesis are discussed in the following subsections.

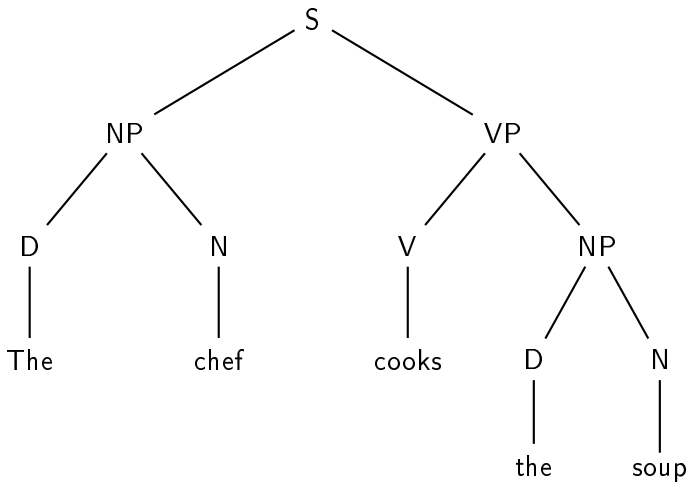


Figure 2.5: Example of the hierarchical nature of natural language. The sentence is parsed into several parts of speech that are grouped to form constructs, such as noun phrases (NP).

2.6.1 Representation Learning

Another alternative to manual feature construction and the way comments are represented in Section 2.4.1, is the use of distributed representation models [115]. This section discusses distributed representations for individual words, with the sections that follow discussing distributed representations for entire comments.

The N-gram models discussed in Section 3.2.4 essentially represent words as *one-hot* vectors, while a distributed representation represents each word as a fixed-size vector of real values. With one-hot representations, each word is represented by a vector of length $|V|$, where V is the vocabulary. So, for example, the representation of a single word would be $[0, 0, ..., 1, ...0, 0]$, and would be orthogonal to all other words, which says nothing of the relations between words. An example of a one-hot representation would be the following:

```
Vocabulary = (Monday, Tuesday, fruit, apple, today)
Monday   =      [1, 0, 0, 0, 0]
Tuesday  =      [0, 1, 0, 0, 0]
fruit    =      [0, 0, 1, 0, 0]
apple    =      [0, 0, 0, 1, 0]
today    =      [0, 0, 0, 0, 1]
```

Note that the inner product of any two words in the vocabulary is zero, regardless of their semantic similarity. Below are the same words, but represented with real-valued vectors:

```
Monday  =  [0.4, 0.1, 0.2]
Tuesday =  [0.2, 0.05, 0.3]
fruit   =  [0.9, 0.8, 0.03]
apple   =  [0.72, 0.4, 0.1]
today   =  [0.3, 0.2, 0.4]
```

Here, the words in the vocabulary are all represented by 3-dimensional real-valued vectors. The cosine similarity, given as $\frac{A \cdot B}{||A|| \, ||B||}$ (where A and B are document vectors), between two obviously related terms such as `Monday` and `Tuesday` is 0.869, but the cosine similarity between seemingly unrelated words like `today` and `fruit` is 0.681.

Distributed representations (or encodings) are used to map (or embed) words into a high-dimensional semantic word space, where this idea of similarity is maintained, after which these representations can be combined to represent sequences of words (e.g. phrases, sentences, documents) [174]. Such a representation can also encode richer contextual information, like similarities between words, in this high-dimensional space. These word vectors are usually constructed using neural network techniques. Distributed representations learned with neural networks have been shown to significantly outperform N-gram models in the task of statistical language modelling [11]. The N-gram model approach disregards the context of N-grams (beyond the words within N-grams) and the semantics of words. Using distributed representations can also significantly reduce the dimensionality of a feature set (i.e. one feature per word in the input space when using one-hot representations vs. a fixed-size distributed representation).

The idea of representing individual words as dense fixed-size vectors, was first proposed by Geoffrey Hinton [74] and popularised by Google's Mikolov et al. [115, 114], who were involved with making one of the leading libraries for learning distributed representations for text, which we discuss next.

Word2Vec

Google released a popular neural network implementation for calculating a distributed representation for words in a corpus, called `word2vec`. The basic algorithm takes a text corpus (e.g. all the News24 or Slashdot comments) and a dimension as input and produces a real vector of the given dimension for each unique word in the corpus (e.g. [bird] $= [0.1, -0.5, 0.02, 0.10]$ for dimension 4). The word vectors can then be used as features for various machine learning tasks.

The algorithm also outputs the model that can be used for retrieving representation vectors for input words. If a word has not been seen by the model, the model can be further trained on new contexts that contain the word.

Mikolov et al [115] introduced the training algorithms used in `word2vec`, which are the Continuous Bag-of-Words (CBOW) and the continuous skip-

gram models, both based on neural network language models (NNLMs) [11]. The task of learning distributed word representations is an unsupervised learning task, but to train the model, some type of interim "label" needs to be created (making it an implicit supervised learning task) — this is what the CBOW and continuous skip-gram models achieve. Essentially, the skip-gram model aims to predict the words surrounding a given word, whereas the CBOW model aims to predict a word given its surrounding words (or context).

Mikolov et al. state that the CBOW model is several times faster than the skip-gram model, but the skip-gram model works better with smaller data sets. Therefore, since the task at hand deals with a large set of comments (in the order of a few million words, which is at a similar scale to Mikolov et al.'s experimental work), only the CBOW model will be investigated. If fully optimised, `word2vec` is able to process 100,000 words per second on a 2.3Ghz Macbook Pro [54].

The developers of `word2vec` showed that the resulting embedding has very interesting properties and captures many linguistic regularities. Mikolov et al. [115, 116] showed that doing basic vector arithmetic with the word vectors, produced results indicating the representation captures various semantic properties. For example:

$$[\text{King}] - [\text{Man}] + [\text{Woman}] \simeq [\text{Queen}]$$

and

$$[\text{Paris}] - [\text{France}] + [\text{Italy}] \simeq [\text{Rome}] .$$

Section 3.4.2 shows related results from models trained on News24 and Slashdot comments respectively.

Other examples of relationships between words are shown in Figure 2.6 where the words in a column are the closest (in terms of similarity) to the word in the column heading.

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

Figure 2.6: A list of example relationships between terms in an embedding. Table taken from [43].

This notion of 'closeness' between word vectors can be used to form a word embedding, which can be visualised as a two-dimensional concept plane, using t-SNE [175]. An example region from a word embedding is shown in Figure 2.7. These clearly show how semantically similar words are grouped together (e.g. words related to finance are grouped together on the left). Using the word embedding, a very high-dimensional word space can be collapsed to equivalence classes by forming clusters of words that are "close" (to a certain degree) in terms of their vector representations, which addresses the inherent sparsity associated with N-gram models and significantly improves generalization. This is conceptually similar to how topic vectors are generated using LDA (as discussed in Section 2.5).

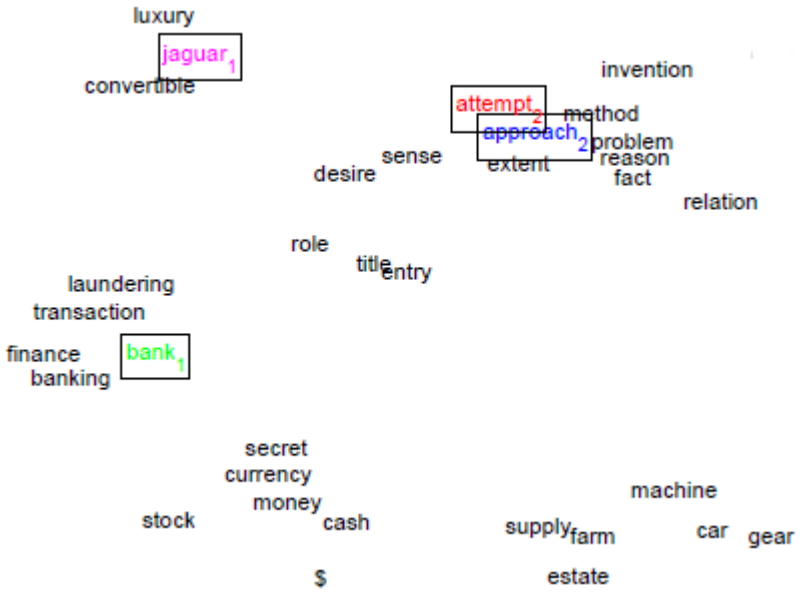


Figure 2.7: A small part of a word embedding. Image from [81].

Bilingual word embeddings are also possible and similar results as above were obtained when combining German and English [105]. This is illustrated in Figure 2.8. Here, English and German words are embedded into the same space with additional prior knowledge of which English and German word-pairs are related and should be "close" in the embedding. The model is then able to generalise to unknown English and German word-pairs that are placed close together in the embedding and are in reality related.

A known concern with models like word2vec, is that homonyms (words that are spelled the same but have different meanings) will distort the representation of words in the model, as they will be used in different contexts. This is, however, a relatively rare occurrence and if a word-vector were to represent multiple meanings, it would be some weighted combinations of the meaning, so the vector should still carry the semantics of the contexts to

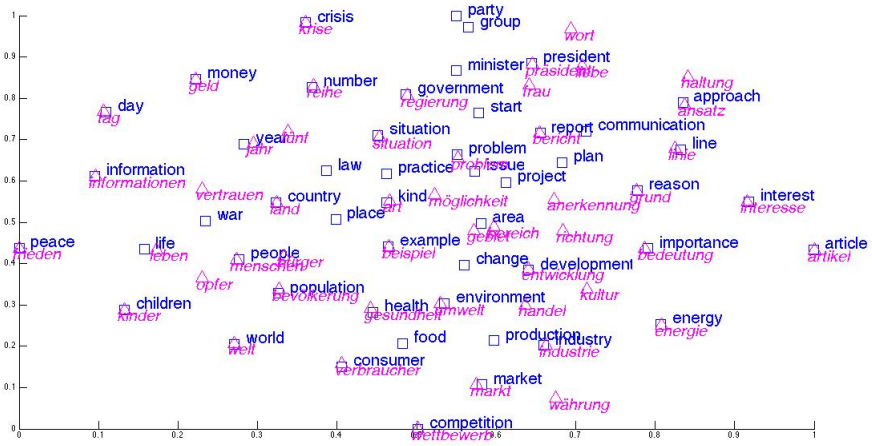


Figure 2.8: Visualization of a bilingual word embedding between German and English. German words are shown in pink with English words shown in blue. Image from [105].

some degree. The problem could be tackled with word-sense disambiguation, but this is often a computationally intensive task to perform on large corpora.

The creators of `word2vec` published a pre-trained model that was trained using a data set from Google News. The model contains 300-dimensional word vectors for about 3 million words. This model will be compared to similar models trained on News24 and Slashdot data, as proposed in Chapter 3. Next the underlying model we use in `word2vec` is discussed.

NNLM and the Continuous Bag-of-Words Model

The CBOW model is based on Neural Network Language Models (NNLMs), as proposed by Bengio et al. [11, 12]. The NNLMs use neural networks to map a sequence of words (with associated feature vectors) to a probability distribution over all the words in the corpus. In speech and text recognition, probabilistic models are used to find the most likely translation of a series of words given their context. NNLMS are also used in predictive text systems to predict the next word a user would type based on the words already entered. A predictive text system is often a specific type of NNLM called the N-gram language model where the probability of a word is calculated given only the N-1 words in its prior context. If the context is a single word, the model is a bigram model.

These models work under a *Markov* assumption in the sense that it assumes a future word can be predicted entirely by words in a certain context surrounding it. In particular, the NNLM is a distributed representation

learning method that learns a vector associated with each word by building a neural network with trainable weights that learns to predict the word given the context words preceding it. This context window can be arbitrarily large, but its size influences the computation time in the hidden layers, thus increasing the training time of the network. The outcome of training such a network, is that word vectors are mapped into a semantic vector space where semantically similar words have similar vectors (e.g. "weak" and "timid").

The training set for such a model, is a sequence of words $(w_1, ..., w_T)$ (where T is the size of the training set) from a vocabulary V. Associated with each word is a 1-of-V vector that represents the index of the word in the vocabulary. NNLM tries to learn a function that determines the conditional probability of a word w_t given the previous $n-1$ words, given as $P(w_t|w_{t-1}, w_{t-2}, ..., w_{t-n+1})$. This conditional probability can be expressed as two nested functions [10]:

- A mapping C that maps a word index to a real vector (i.e. the word's distributed representation). Practically, C is a $|V| \times m$ lookup matrix, where m is the predetermined size of the distributed representation.
- A probability function g over the distributed representations of the words. Function g maps a sequence of vectors (the history of word w_t in this case) to a conditional probability over the entire vocabulary V. g produces a $|V|$-dimensional vector where the i^{th} value indicates the probability that the i^{th} word in the corpus follows the words in the history window.

Thus, $P(w_t|w_{t-1}, w_{t-2}, ..., w_{t-n+1}) = g(C(w_{t-1}), ..., C(w_{t-n+1}))$ and is often modelled by a neural network, called a NNLM, such as the example in Figure 2.9. Here, the representation for a word w_t is learned from a context window (or history) of 4 words $w_{t-1}, ..., w_{t-4}$. The indices associated with the words in the context window are used to retrieve the respective distributed representations, $C_{w_{t-1}}, ..., C_{w_{t-4}}$, associated with each of the words, and are then concatenated to form a vector x which forms the input for the hidden layer. The C matrix is essentially a weight matrix for the input neurons. The hidden layer is a simple hyperbolic tangent layer which in turn forms the input for the $|V|$ nodes in the output layer where a softmax function is applied to all the output nodes to form the conditional probability distribution over all the words in the vocabulary for word w_t [16]. The output of training this network is the set of continuous vectors C that are the distributed representations of the words in the vocabulary.

Training is achieved by maximising the *log-likelihood* (penalized by the size of the training corpus T) of the input training set. For a context window of size $n-1$, the log-likelihood L is given as:

$$L = \frac{1}{T} \sum_t \log P(w_t|w_{t-1}, w_{t-2}, ..., w_{t-n+1})$$

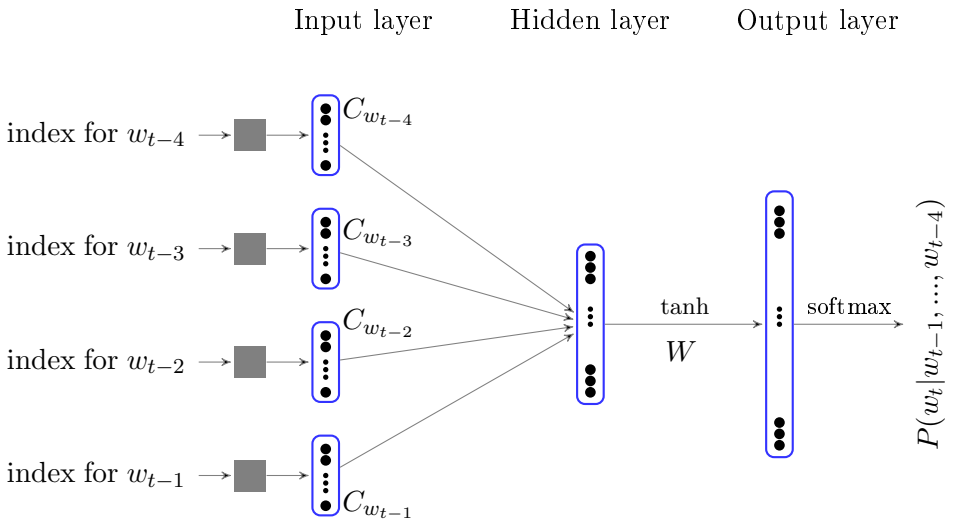


Figure 2.9: A NNLM where the input layer consists of *one-hot* encoded input vectors (representing words) for a word window of size 4 and a vocabulary of size V. The network outputs a probability distribution of size V.

The probability of a word w_t given its context, as computed with a softmax function, is:

$$P(w_t|w_{t-1}, w_{t-2}, ..., w_{t-n+1}) = \frac{e^{a_{w_t}}}{\sum_{l=1}^{|V|} e^{a_l}}$$

where a_l represents the unnormalized log-probability for each output word l, computed as follows:

$$a = b + W \tanh(c + Hx)$$

In the formula above, b is the bias of the output layer, c is the bias of the hidden layer, W is the set of weights of the connections between the hidden and output layers, and H is the set of weights between the input and hidden layers. These variables form the free parameters of the neural network. Stochastic gradient descent [144] can then be used to train these variables. The neural network size is controlled by the number of words in the context window and the dimensionality of each of the word feature vectors. The computational complexity of the network lies with the hidden layer where the hyperbolic tangent is applied to each node.

CBOW is a generalization of the main NNLM procedure, but also an improvement on NNLM in terms of speed and the size of the network. NNLM assumes a word can be adequately predicted based on only the words before it. The CBOW procedure takes a more general approach and considers

a word's neighbourhood to be the words before and after the given word, within sentence boundaries. Also, the network's hidden layer, that previously concatenated the input vectors, is replaced by a layer that performs a simple element-wise linear combination on the input vectors (e.g. a weighted average or a sum of the vectors). This means the size of the hidden layer is only determined by the dimension of the dense vectors attributed to the words in the vocabulary. This also means that arbitrarily large context windows can be used, since the size of the network is only affected by the size of the vocabulary. An illustration of the CBOW model is shown in Figure 2.10. Mikolov et al. [113] showed that NNLM can train 100-dimensional vectors for 6 billion words with their contexts in 14 days with 180 CPU cores, but that CBOW could train 1000-dimensional vectors for the same words in 2 days with 140 CPU cores.

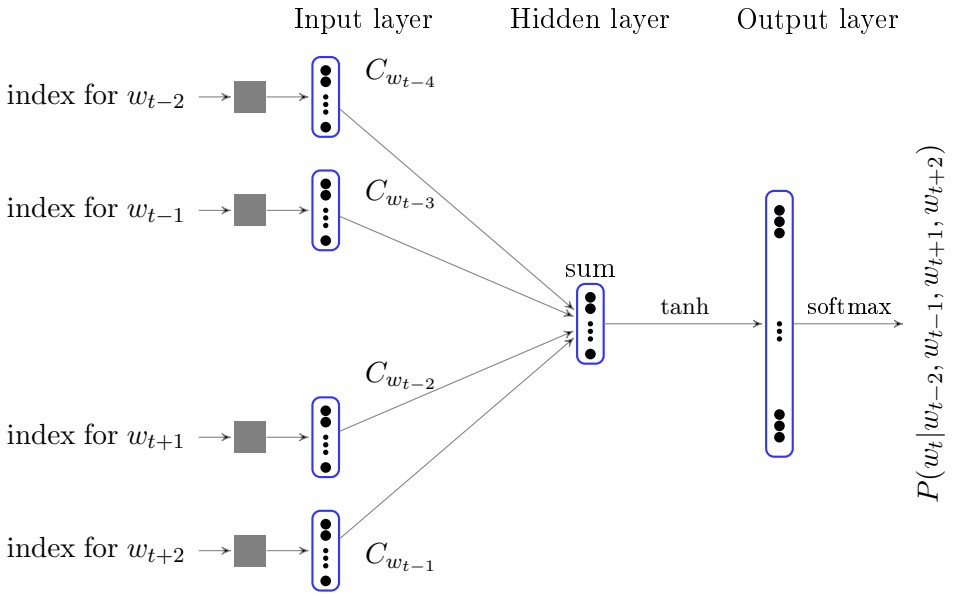


Figure 2.10: An example of a CBOW neural network that predicts the word w_t given its past and future contexts. The input vectors are put through a linear transformation layer (usually a weighted sum or weighted average of the input vectors) followed by an output layer with a softmax function.

Essentially, CBOW seeks to predict a word w_t with a context window size k, by taking the preceding k words and the following k words as input (e.g. with $k = 2$, the context is $w_{t-2}, w_{t-1}, w_{t+1}$ and w_{t+2}) with a multiclass classifier such as softmax. As a minimal example, consider the sentence "Hi Fred, how was the pizza?" and a context window of size one, then the contexts of the words would be {"Hi **Fred** how", "Fred **how** was", "how **was** the", "was **the** pizza"} where the predicted word is shown in bold. In the traditional NNLM, the input vectors are concatenated, whereas in CBOW

the vectors in the context window are summed or averaged, reducing the number of hidden nodes necessary.

Additional optimizations are necessary with very large vocabularies, where the size of the output layer of a traditional NNLM would be in the order of millions of nodes, which is inefficient to evaluate for softmax, since the complexity of calculating softmax is in the order of the size of the vocabulary. To avoid this, the CBOW network is trained using the hierarchical softmax (HS) algorithm and negative sampling [121, 114].

Hierarchical softmax is a variant of softmax that uses a binary Huffman tree to encode the words in the training corpus, where the $|V|$ words form the leaves of the tree. Each node in the tree explicitly represents the relative probabilities of its children. The softmax calculation is then done by descending the Huffman tree to the leaves and multiplying the probabilities of each step together. The final probability at the leaf node is then the conditional probability of the word at that leaf. This optimization essentially decreases the time complexity of the softmax stage from linear (in the size of the vocabulary) to logarithmic. Negative sampling evaluates a training sample by only evaluating the output neuron that represents the desired word in the training sample plus some randomly chosen additional neurons.

Mikolov et al. [115] introduced both the CBOW and continuous skip-gram models, but reported that CBOW is several times faster to train on large data sets. In contrast to the CBOW model, the training objective of the continuous skip-gram model is to predict the context vectors of a word given the distributed representation of the word. Simply put, the input of the skip-gram model is the index of w_t, which is used to retrieve C_{w_t}, which is then directly used as input for a softmax function. The softmax produces a $|V|$-dimensional vector where the j^{th} element shows the probability that the j^{th} word in vocabulary is in the context of word w_t.

The name "skip-gram" is used because contexts are often built not by the immediate neighbouring words, but by skipping a constant number of words to obtain a wider context. This leads to the continuous skip-gram model often obtaining better representations for rare words than CBOW [115].

2.7 Summary

This chapter discussed well-known techniques from the research literature for automatic quality prediction of online user-contributed comments, as well as techniques for representing short texts (or comments). Further, existing filtering and moderation schemes used on popular websites where users contribute content, as well as techniques used in the research literature to approach this and other similar tasks, were mentioned. The supervised learning techniques used to train models on the feature sets, viz. NB and SVMs, were discussed.

Background on the techniques for feature construction that this thesis investigates was also discussed. Three main feature construction techniques were introduced in this chapter: a custom-designed feature set based on techniques in the research literature, N-gram-based approaches for comment representation and distributed representation techniques based on popular approaches in deep learning.

Chapter 3

Feature Identification

This chapter discusses in more detail our application of the proposed approaches to feature construction considered in this thesis.[1] First we detail a suite of techniques for hand-crafted domain-specific features. Next, we discuss our N-gram-based representations of short texts [126, 143]. The third approach we discuss is one based on topic modelling. Finally, distributed representations inspired by techniques from deep learning are discussed. All of these features will be evaluated with SVMs in Chapter 5.

For simplicity, all formulae in this chapter will be shown for n sample comments, and when calculating features, the formulae will be given in terms of a sample input comment c_j and the comment c_{j-1} that precedes it in a thread (from a corpus of comments $C = \{c_j : j \in 1, ..., |C|\}$). Comment c_j consists of a set of words W (W$'$ will denote the unique words). $|c_j|$ will denote the number of words in the comment. The set of all comments that precede a comment in a thread will be denoted by C_H.[2]

3.1 Custom Feature Construction

This section discusses the various hand-crafted feature extraction methods used on the training data. Most of these features are based on or taken directly from Wanas et al. [180], Hsu et al. [79], Weimer et al. [183] or Cheng et al. [39], and are indicated as such where relevant. The features we construct are grouped into two categories, namely *post features* and *user features*, where post features focus on identifying characteristics of the text and user features incorporate information about the user's reputation and

[1] This chapter contains portions of work from previous papers co-authored with one or more of my co-supervisors [25, 26].

[2] If a feature uses the comments in a thread for a calculation, it would always use only the comments preceding the comment in question, to preserve the order in which comments are added to the thread.

commenting patterns. The features are detailed below and a list of the features used is given in Table 3.1.

Post Features	Timeliness, relative length, absolute length, part-of-speech counts (verbs, nouns, pronouns), uppercase word frequency, question and exclamation frequency, capitalized sentence frequency, entropy, lexical diversity, spelling, profanity, informativeness, readability, relevance, polarity, subjectivity, comment-article sentiment overlap.
User Features	In-degree, out-degree, user account age, post count, post rate, PageRank, authority and hub scores.

Table 3.1: List of the custom feature extraction methods.

3.1.1 Post Features

Timeliness

Timeliness reflects the response time of a user's comment relative to the posting time of other comments in its thread [180]. It takes into account the rate at which comments are posted by users and affects the probability that a comment will be seen by other users and subsequently commented on [169]. Timeliness is calculated as follows:

$$\text{Timeliness}(c_j) = \frac{T_{c_j} - T_{c_{j-1}}}{\frac{1}{|C_H|}\sum_{i=1}^{|C_H|}(T_{c_i} - T_{c_{i-1}})} \quad \forall j = 1, ..., |C|$$

where T_{c_j} is the time that the j^{th} comment in the thread was posted. The post time of the first comment can not be calculated.

The absolute time in seconds between the first comment and every other comment in the thread is also stored as a feature.

Relative Length

This feature simply measures the length of a comment relative to previous comments in its thread [180]. The value is normalized by the mean length of previous comments in the thread. The value is calculated as follows:

$$\text{RelativeLength}(c_j) = \frac{|c_j|}{\frac{1}{|C_H|}\sum_{i=1}^{|C_H|}|c_i|} \quad \forall j = 1, ..., |C|.$$

Absolute Length

The number of characters in the comment is used as a feature.

Part-of-speech Count

The number of verbs, nouns and pronouns in the comment are obtained by using a part-of-speech tagger from `NLTK` [15] and are all used as features.

Uppercase Frequency

This feature is the proportion of the number of words in a comment that are completely uppercase [79].

Question and Exclamation Frequency

These features are the proportions of sentences in the comment that end in question and exclamation marks respectively [183].

Capitalized Sentence Frequency

This feature determines the proportion of sentences in the comment starting with a capital letter. Although we have not encountered this feature in the literature, we thought it would be suitable, since we hypothesise that using capitalised sentences could be an indicator of quality in comments. In Section 3.5 we show that, for the small News24 data set, this feature is suitable for classification.

Entropy

The complexity of a comment is measured by the entropy of the words in the comment, as defined by Hsu et al [79]:

$$\text{Entropy}(c_j) = \frac{1}{|\text{W}'|} \sum_{w \in \text{W}'} f_w[\log_{10}(|\text{W}'|) - \log_{10}(f_w)]$$

where $|\text{W}'|$ is the number of unique words in the comment and f_w the frequency of word w in the set of unique words W'.

Lexical Diversity

We designed another, simpler, measure of complexity, namely the lexical diversity of the comment. The feature was added to model a comment author's range of vocabulary. We hypothesise that high lexical diversity values could be an indicator of quality.

The lexical diversity of a comment is calculated as the number of unique words in a comment divided by the total number of words in the comment.

This function approaches 0 as more words are repeated in a comment and will be 1 if all the words in the comment are unique. Thus, we expect large values to indicate good quality comments.

Spelling

The number as well as the frequency of incorrectly spelled words in a comment are used as features. The feature is calculated by looking for each word in a dictionary and recording how many lookups are unsuccessful. The dictionary is comprised of words extracted from Peter Norvig's spell checker data sources [125] and the NLTK [15] sources for male and female names.

Profanity

These features record both the number of curse words and the proportion of words that are curse words (and other profanities) in a comment [27]. Similar to the spelling feature, the feature value is calculated by looking up each word in a profanity list and recording the number of words found in the list. The list of profane words was obtained from material published by Alejandro U. Alvarez [4].

Informativeness

The informativeness feature captures how distinctive a comment is within the set of all comments in its thread [79]. The measure that was used, is a modification of the standard TF-IDF measure for individual words [2]. The basic formula for TF-IDF is presented in Section 3.2.4: this is modified by dividing the term frequency by the maximum term frequency in the document (to avoid bias towards longer documents). The informativeness of a comment is then calculated by taking the sum of the informativeness of the individual words in the comment:

$$\mathrm{I}(c_j) = \sum_{w \in W} \frac{\mathrm{tf}_w}{\mathrm{mtf}_{\mathrm{W}'}} \times \mathrm{idf}_w \tag{3.1}$$

where tf_w is the term frequency and $\mathrm{mtf}_{\mathrm{W}'} = \max_{w \in \mathrm{W}'} \{\mathrm{tf}_w\}$ of comment c_j. The idf_w component is the inverse document frequency of term w in all the comments in the thread (preceding the comment in question) and is the standard IDF measure.

The average term frequency of words in the comment is also used as a feature.

Readability

The readability of a comment (as determined by the Flesch Reading Ease Score (FRES) [61]) is a measure of the ease with which a reader should be

able to read the comment [79]. The test uses is a linear model that penalizes long sentences and long words. The formula for the FRES is:

$$FRES(c_j) = 206.835 - 1.015\left(\frac{|c_j|}{s_j}\right) - 84.6\left(\frac{b_j}{|c_j|}\right)$$

where s_j is the number of sentences in comment c_j and b_j is the number of syllables in comment c_j. The number of syllables is obtained by looking up each word in the Carnegie Mellon Pronouncing Dictionary [182].[3]

A high score (above 90) indicates that the text can be understood by an average 11-year old, whereas a low score (between 0 and 30) indicates that the text will probably only be understood by university graduates.

Relevance

The relevance of a comment is measured relative to the article and relative to the comment thread that the comment belongs to [180].

To calculate the relevance within the set of comments, the overlap is quantified between the words in the comment and the words in the thread. A vocabulary V is generated from all the comments in the thread preceding comment c_j and the relevance of the comment is then measured according to the following definition:

$$\text{Relevance}(c_j) = \frac{\mid W \cap V \mid}{\mid W \mid} \tag{3.2}$$

Similarly, to calculate a comment's relevance to the article, a bag of words is generated from the body of the article and Formula 3.2 is used.

Polarity

The polarity of a comment is quantified as the proportion of the comment's words that are negative or positive in terms of sentiment. To determine the polarity of a comment, a 2-class classifier (Naïve Bayes) was trained to predict the sentiment of a comment [103]. The classifier was trained and tested using a corpus of 50,000 positive and 50,000 negative tweets (posts) that were hand-labelled as either positive or negative.[4] The classifier produces per-word probabilities to determine the sentiment of the words in the comment. The classifier achieved a prediction accuracy of 84.7% on a holdout test set of 25% of the tweets.

[3] If the word is not in the dictionary, a default syllable count of 1 is used.

[4] This was chosen as a training set, as it is the closest sentiment-labelled training set that relates to comments that could be found.

Comment-Article Polarity Overlap

To capture the commenter's alignment with the article's author, the sentiment of the article (as determined by the aforementioned Naïve Bayes classifier) is compared to that of the comment [79]. If both the article and the comment shows the same sentiment (i.e. both positive or both negative) a 1 is recorded, otherwise a 0.

Subjectivity

The subjectivity of comments are also captured as a feature [79]. The positive polarity probability of a comment is determined by the NB classifier in the polarity feature and if the probability is below 0.4 or above 0.6, the comment is labelled as subjective; otherwise it is labelled as objective.

3.1.2 User Features

Users that contribute content to an online community are valuable to that community. User reputation helps to bring trust and legitimacy between the different parties in online transactions. An example would be the use of reputation in reducing fraud on an online auction site, as was done by Gregg and Scott [66] when examining eBay.

There are various approaches to quantifying a user's reputation and some of the measures require knowledge of the user's network effect [35, 66]. Sabater and Sierra [146] noticed that direct interaction between users, albeit a good source of user reputation, is scarce in large multi-agent systems. They designed a system that takes advantage of the social relations between users in a network, resulting in a reputation model based on social network analysis.

Social network analysis [146] is the study of relationships between individuals in a community, where relationships are represented using directed, weighted graphs called **sociograms**, where the nodes represent the users and the edges relationships between users.

Many examples of sociograms being used for analysis of social networks exist, such as [1, 35, 146]. Different types of relations are often defined for the graph, because participants in the social network can perform different types of actions. Different types of actions can also carry different weights, which signify the relative importance of actions to a user's reputation.

For quality prediction in comments, certain social features can be extracted from a sociogram of the commenters in the system. Consider the graph $G = \langle V, E \rangle$, where V is the set of all users that post comments and E is the directed edge set representing replies to parent comments. The graph is constructed by adding a connection for each child comment, where the connection is a directed edge from the user that left the comment to the user that posted the original parent comment. If a user comments multiple

times on a single parent comment, multiple edges are added. Also, if a user comments on parent comments in multiple threads, edges will be created for each of the individual comments. Thus, nodes can be connected with multiple parallel edges. It is also possible for isolated nodes to exist if a user has only left comments on articles and nobody has replied to their comments.

Below is a list of features that will be extracted as "social features" from the resulting graph [143, 1]. The `python` library `NetworkX` is used to construct the user graph and for calculating the features mentioned below.

- **In-degree** — The number of comments that a user has received across the comment base;
- **Out-degree** — The number of other comments the user has commented on;
- **PageRank** — The PageRank [129] value of the user (see below); and
- **HITS** — The Hyperlink-Induced Topic Search (HITS) authority and hub values of the user [93].

In addition, the following information about the user is also used as features:

- **User "age"** — The difference between the user's first comment ("join date") and the user's last comment (in days); and
- **Post count** — The number of comments the user has posted;
- **Post rate** — $\frac{\text{Post Count}}{\text{User Age}}$.

PageRank algorithm

Google search is based on an algorithm called PageRank (PR) [129, 37] which measures the relative importance of websites when delivering search results. If the internet is represented as a huge graph where web pages are nodes and links are edges, a traditional measure of importance would be the in-degree of a node (number of links pointing *to* a page).

PageRank is essentially a link analysis algorithm that weights pages according to their relevance. The original paper [129] defines a page A's PageRank as:

$$PR(A) = (1-d) + d(PR_{T1}/C_{T1} + ... + PR_{Tn}/C_{Tn}) \tag{3.3}$$

where d is a damping factor (which leverages the influence of "neighbouring" pages on a page), $T1$ to Tn are other pages with links pointing to page A, and C_T is the number of outgoing links of page T. The damping factor is usually set to 0.85 in practice.

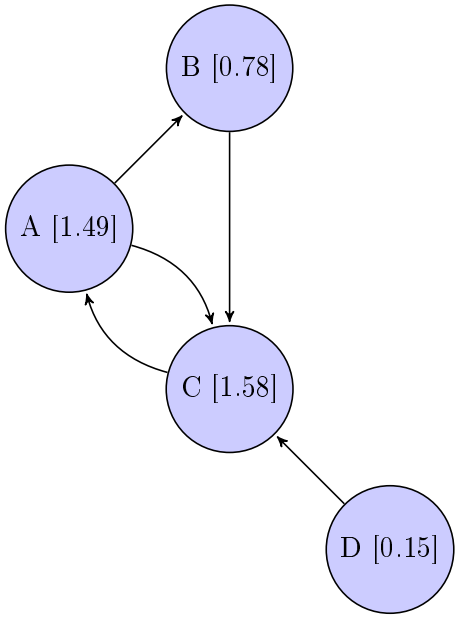


Figure 3.1: Example of PageRank scores with $d = 0.85$.

An example showing PageRank values for a small sociogram is presented in Figure 3.1.

The correct PR values are indicated on the nodes. The algorithm initially randomly selects a positive PR value for all nodes and then iterates applying Equation 3.3 until the PR values converge.

HITS algorithm

Similar to PageRank, Hyperlink-Induced Topic Search (HITS) [93] is a link analysis algorithm. It distinguishes two roles for pages, namely acting as *hubs* and *authorities*. Hubs serve as directories to other pages and do not necessarily provide authoritative information themselves, whereas authorities are pointed to by many hubs, making them good sources of authoritative information. Most pages are a combination of the two, linking to other pages and being linked to. Intuitively, a page's authority score is the sum of the hub scores of all the pages that link to it, and a page's hub score is the sum of the authority scores of all the pages it links to.

This can be translated to any graph to determine characteristics of the nodes in the graph. The HITS algorithm determines hub and authority scores (u_i and v_i respectively, initially 1) for each of the nodes in the sociogram. A minimal example is shown in Figure 3.2.

Calculating the authority vector $\mathbf{v}$ and hub vector $\mathbf{u}$, giving all the authority and hub scores for the nodes, only requires a few simple matrix multiplications. The hub values are updated by multiplying the adjacency

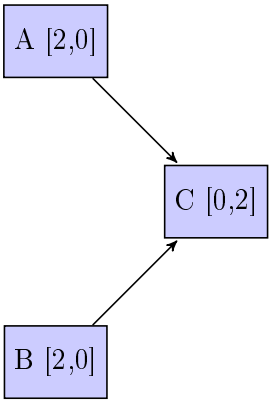


Figure 3.2: Example of the HITS algorithm.

matrix by the authority values ($u = Av$) and the authority values are updated by multiplying the transposed adjacency matrix by the hub values ($v = A^T u$). This process is repeated until convergence.

The adjacency matrix of the graph in Figure 3.2 is $A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ with transpose $A^t = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$ and the initial hub values $u = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

Then the authority values are updated as follows:

$$v = A^T u = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

and the updated hub values are:

$$u = Av = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix}$$

.

Then nodes A and B have a hub value of 2, and authority levels of 0 each, and vice versa for node C. Further iterations of the algorithm only produce scalar multiples of the vectors, so the relative weights remain the same.

3.2 N-Gram-Based Feature Construction

As discussed in Section 2.4, both single-word representations and higher-order N-gram representations (with $N > 1$) will be investigated and their

suitability for representing comments will be evaluated. This thesis investigates both word and character N-grams.

3.2.1 Word N-grams

For word N-grams, the effects of three main choices in the construction of the N-gram models is investigated.

The first is the order of the N-grams used — unigrams ($N = 1$), bigrams ($N = 2$) and trigrams ($N = 3$) are considered (detailed in Section 3.2.4). Other N-gram representations with $N > 4$ could be used, but the resulting feature vectors are extremely sparse, making them unsuitable for training a classifier. For each choice of N, either N-gram features alone (denoted by "="), or a feature vector using all N-grams of that order or lower (denoted by "$\leq$") are considered. Finally, three different vectorization methods for determining the value of each component in the resulting comment vector are considered: the binary count (denoted by "B"), the frequency count (denoted by "F")[5] and the TF-IDF-normalized frequency (denoted by "T"). These three methods are explained in detail in Section 3.2.4. The selection of N, whether lower-order N-grams are included or not, and the vectorization method are summarized in the feature set notation by concatenating the three symbols denoting the three choices. For example, "$\leq$3T" represents the use of unigrams, bigrams and trigrams with TF-IDF-normalized vectorization.

3.2.2 Character N-grams

Character N-grams are used in addition to word N-grams because they could potentially handle inconsistent spelling in words (i.e. users that use the same word but spell it slightly differently). This is useful for users trying to change the spelling of bad words or derogatory terms to prevent them from being filtered. For this, character skip-grams are even better suited, since users might replace characters in the middle of words to obfuscate them. Thus character skip-grams are also investigated.

For character N-grams, all the N-grams for values of $2 \leq N \leq 8$ are included in a single representation (denoted by "$C28$"). These N-grams are taken across whole sentences (i.e. with spaces included). As with the word N-grams, the binary and TF-IDF-normalized frequency vectorization methods are used. This produces two feature sets, namely $C28B$ and $C28T$.

For character skip-grams, each complete word, as well as all the variations of that word with characters left out, are included (i.e. the character skip-grams for the word "bird" are "bird", "ird", "brd", "bid" and "bir"). This means

[5] The frequency count is only investigated for unigrams, since it was experimentally determined that higher order N-grams rarely occur multiple times within a comment, so there is very little difference between the binary and frequency count representations.

that character skip-grams are not calculated across words and they do not include whitespace. This is done to inhibit the growth in dimensionality from using the skip-gram approach. Again, the binary and TF-IDF-normalized frequency vectorization methods are used. This produces two feature sets, namely CSB and CST.

3.2.3 Pre-Processing

For the data sets mentioned below, two pre-processing steps are done before the feature vectors are generated: stop-word removal [136] and lemmatization.

The list of stop-words are taken from the `NLTK` corpora [15]. Thereafter, words are transformed into their base dictionary form (called the lemma) through a process called lemmatization. As an example, consider the words *car*, *cars*, *car's* and *cars'*. Through lemmatization, all these words are mapped to *car*. This is done to group plurals and other word variations into a single representative term. For lemmatization, an implementation from `NLTK` [15], that makes use of the WordNet [117] lexical database, is used.

3.2.4 Constructing N-gram Representations

Let V be the set of all terms (or N-grams) in the corpus. The values in this set are called term-features. Depending on the representation, this set either contains distinct words (unigrams), sequences of words (bigrams or trigrams), character N-grams (for $N = \{2, 3, 4, 5, 6, 7, 8\}$), or character skip-grams. Each data sample (comment) is then represented by a vector of term-feature weights $\langle e_{j1}, ...e_{j|V|}\rangle$. Here e_{ji} is the weight of term i in comment j and $|V|$ is the size of the set of all terms.

All the representations are implemented using row-wise sparse data representations from `scipy` [87]. This representation stores only the non-zero entries in the matrix, saving on memory when the matrices are extremely sparse, as is the case with N-gram representations.

For vectorization (or occurrence counting), three different schemes are considered:

- **Binary Count** — For each comment c_j, the i^{th} term-feature will be 1 if the word appears in the comment and 0 otherwise;
- **Frequency Count**— Instead of simply considering the existence of a term in a comment, the frequency of occurrence is a common weighting [104]. For each comment c_j, the i^{th} term-feature will contain the frequency of the N-gram in the comment; and
- **TF-IDF Normalization**— TF-IDF is a per-word measure to reflect the relative importance of a word within a comment (first introduced

by Salton and Buckley [149]). TF is simply the frequency of a word in a comment and IDF the frequency of the word in the corpus of comments. Longer comments have higher frequencies of terms and are hard to compare to shorter texts for similarity. Also, terms that occur very frequently in all texts, provide little discriminatory information for a classifier. Therefore, approaches using inverse-document frequency and length normalization have become popular in vector space models [149]. For each comment c_j, the i^{th} term-feature is given by:

$$\text{tfidf}(c_j, t_i) = \frac{tf_i}{|c_j|} \times \ln\left(\frac{|C|}{f(t_i, C) + 1}\right)$$

where tf_i is the frequency of term t_i in comment c_j, normalised by the number of words in the comment, C is the set of all comments, and $f(t_i, C)$ is the number of comments t_i appears in.[6] This is the implementation used by the `python` library `scikit-learn` [131] for TF-IDF.

3.3 Topic Modelling for Feature Construction

This thesis uses LDA to construct topic models for both News24 and Slashdot comments. The models are trained using `Gensim` [139], an open source `python` library designed for topic analysis of text documents. For the News24 model, an archive of unlabelled comments (approximately 1 million comments) was used to train the model, where the model was trained to identify 100 topics in the corpus. Similarly, the second model was trained on a corpus of 350000 Slashdot comments with the same parameters. As an example, 5 random topics are taken from the two models (with the top 5 words and their probabilities in each topic shown) and shown below:

- News24 Model:

topic #1 : 0.084 * href + 0.070 * url + 0.058 * urland + 0.040 * condition + 0.034 * ppl

topic #2 : 0.101 * corrupt + 0.077 * official + 0.047 * afraid + 0.044 * threat + 0.042 * madiba

topic #3 : 0.266 * good + 0.071 * bad + 0.024 * idea + 0.021 * thing + 0.020 * enough

topic #4 : 0.127 * country + 0.076 * sa + 0.073 * government + 0.036 * people + 0.019 * world

topic #5 : 0.093 * year + 0.076 * time + 0.043 * day + 0.036 * old + 0.027 * one

[6] The frequency is artificially increased by 1 to avoid zero division errors.

- Slashdot Model:

topic #1 : 0.021 * slashdot + 0.020 * see + 0.018 * subject + 0.018 * reply + 0.015 * org

topic #2 : 0.036 * market + 0.031 * business + 0.028 * company + 0.022 * product + 0.019 * people

topic #3 : 0.086 * u + 0.027 * war + 0.026 * american + 0.022 * speech + 0.019 * country

topic #4 : 0.058 * law + 0.017 * legal + 0.017 * rule + 0.017 * right + 0.016 * court

topic #5 : 0.048 * post + 0.045 * read + 0.035 * article + 0.035 * slashdot + 0.030 * comment

It is evident that some topics highlight certain concepts, like politics, whereas others simply group semantically similar words. In the Slashdot model, topic #1 and topic #5 both have the word "Slashdot", illustrating that words can feature in more than one topic.

These models are used to infer topic distributions of comments, which will then be used as input feature vectors for the supervised learning algorithms.

3.4 Deep Learning for Feature Construction

Thus far, the feature sets that were constructed for representing comments consisted either of hand-crafted features or features that represent the components of the input space (i.e. words, characters, topics, etc.). The manual approach requires knowledge of the domain and the N-gram-based approaches are unaware of the context and semantics of the components they represent. LDA captures some notion of semantic word context, but topics are not necessarily the best way to represent the underlying themes in the comments (i.e. a word-embedding approach might be better suited). Thus, approaches motivated by deep learning techniques are investigated.

A pipeline outlining the basic procedure for training the word-embedding model is illustrated in Figure 3.3. First, the data is pre-processed to transform the data for the `word2vec` model. Second, a model is trained to learn representations of the pre-processed data. Thereafter, the model is used to extract a feature vector for each comment, after which these feature vectors can be used for classification, as is the case with the other feature sets constructed in Sections 3.1 and 3.2.4. The details of each step will be discussed in the following sections. This pipeline was used to train two `word2vec` models, one on News24 comments and one on Slashdot comments, which were then used to construct feature sets for the News24-large, News24-small and the Slashdot data sets. A third model that was pre-trained on a Google news corpus (that contains larger documents) was also used.

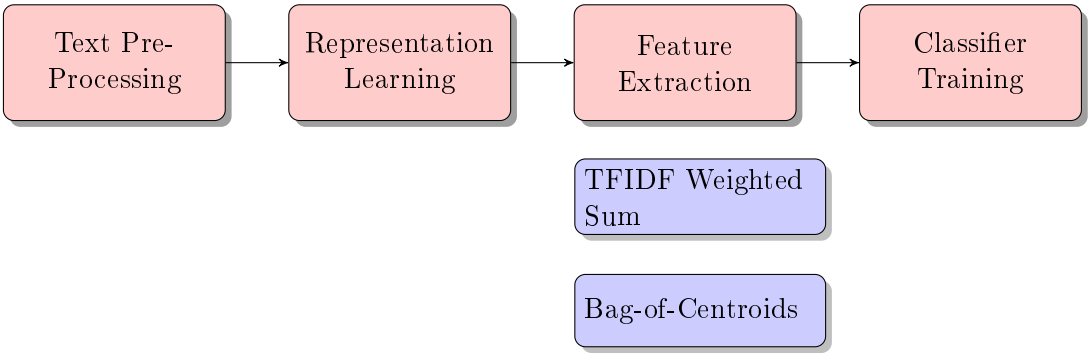


Figure 3.3: A pipeline for extracting word-embedding features.

3.4.1 Text Pre-Processing

One of the strengths of the `word2vec` model is that it can be trained on raw text that has been split into individual words. There is, however, some pre-processing of the input texts that can lead to a stronger representation model. First, all HTML tags are stripped from the comment (to remove noise in the text). Then, the comments are split into individual sentences which are then split further into individual words (all lowercase, to remove distinctions between words based on case). As with the N-gram models in Section 3.2, all punctuation is ignored and all stop-words are removed. This leads to a list of individual words that is used as input data for a phrase detector.

The original `word2vec` library released by Mikolov et al. [114] includes methods for detecting pairs of words (referred to as "phrases") that are frequently seen together, but infrequently on their own. For example, "south africa" is used relatively often in the News24 corpus, when compared to the usage of the individual words, but "this is" won't be recognised as a frequently used phrase, because the individual words are more often used apart than together. The phrase detector is used to combine the most frequently used phrases in the input text with an underscore character (e.g. "cape_town"), leading to input data consisting of words and phrases for the `word2vec` model.

3.4.2 Model Construction

Two `word2vec` models were trained, one on a corpus of 3 million unlabelled News24 comments and the other on about 300 000 Slashdot comments. The models were trained with a 10-word context window to generate 200-dimensional representations for the words in the comments.

Appendix C shows examples of language regularities discovered in these models. One such example is of the phrase 'south_africa': in the News24 `word2vec` model the "closest" (in terms of cosine distance) to the phrase is

'sa', which is its acronym, but in the Slashdot `word2vec` model, the closest other word or phrase is 'switzerland'. This example clearly illustrates the difference in content between the News24 and Slashdot models. The News24 model identifies a synonym for 'south_africa', whereas the Slashdot model fails to capture the local context of the phrase. It only recognizes 'south_africa' as the name of a country, so it finds the names of other countries as related phrases.

3.4.3 Feature Construction

To represent longer pieces of text than single words, such as phrases, sentences and documents, as fixed-length vectors, two approaches to extend word representation models are introduced [120, 114]. The simplest of these approaches is to combine the word vectors as either mean or a sum (depending on the number of words in the texts), which can be made more sophisticated by adding TF-IDF-weighting of vectors. A second approach is to use the nature of the word vectors and the notion of similarity between them, to create clusters of words. This also helps to deal with the massive size of the vocabulary and the sparseness in the N-gram model approaches.

Both these approaches to multi-word compositionality are investigated and discussed below.

TF-IDF-weighted Sum of Word Vectors

Each comment can be represented as the sum of the vectors assigned to the words in the comment, but each vector is weighted by the inverse document frequency of the word in the corpus of comments. Thus, the vector representing comment C_j, is calculated as follows:

$$\mathbf{C_j} = \sum_{t_i \in C_j} \text{tf}_{i,j} \times \text{idf}_i \times \mathbf{w_i}$$

where $\text{tf}_{i,j}$ is the term frequency of term i in comment j, idf_i is the inverse document frequency of term i and $\mathbf{w_i}$ is the word vector of term i as provided by the distributed representation model. The term frequency and inverse document frequency are calculated the same way as in Section 3.1.1.

Bag-of-Centroids

Another way to deal with the high dimensionality of N-gram model approaches, is to cluster the words in the vocabulary, so as to collapse the vocabulary into word classes. Since a natural artefact of `word2vec` is that the resulting word vectors carry a notion of similarity, a common approach is to cluster word vectors to produce these word classes [112, 33]. This is related to the topic modelling approaches introduced in Section 2.5.

Using k-means clustering [46], the word vectors are clustered into semantically related groups of words. Appendix D shows examples of clusters that are obtained from the News24 and Slashdot `word2vec` models.

The clusters extracted from the News24 model clearly shows how semantically related words are grouped together and how it efficiently deals with the issue of words being spelt differently and with words being in multiple forms (e.g. plurals). The Slashdot model seems to identify less useful clusters, which is understandable, since it is trained on a corpus one tenth the size of the News24 corpus.

It has been suggested that relatively small clusters give better results [33], so the number of clusters are chosen so that clusters have roughly 50 words each, which produces about 2800 clusters in the News24 model and 800 clusters in the Slashdot model. After clustering, a standard frequency vectorization method is run over the comments and the clusters, producing a bag-of-clusters vector for each comment C_j, where the i^{th} feature is the frequency of words in the comment belonging to cluster i.

3.5 Most Relevant Features

Identifying the most relevant features (in terms of contribution to predictive accuracy) is valuable. Two common approaches to dimensionality reduction are often used: Principal Component Analysis (PCA) and Linear Discriminant Analysis (LnDA).[7]

PCA aims to find the components (or features) that best explain the variance in the data set, whereas LnDA can additionally find the components that best separate the class labels. PCA ignores class labels, making it an unsupervised learning method and well suited for unlabelled data sets. In contrast, LnDA is a supervised learning method and since the feature sets in this thesis are labelled, LnDA is better suited for feature identification. Additionally, LnDA outperforms PCA when the number of samples per class is relatively large [108], as is the case for the feature sets used in this thesis.

LnDA was developed by Ronald A. Fisher [59] and is often used as a linear classifier. The two-class LnDA algorithm projects an n-dimensional feature space onto a line maintaining maximum discriminatory information about the class labels. The coefficients of the features (also known as the loadings or scores) are used as a measure of feature importance for classification.

Using LnDA, the following 10 most relevant features (in order of relevance) for various feature sets were determined:

In both the large News24 and Slashdot data sets, a user-feature is identified as the most important feature. As expected, with both the News24 data sets, spelling and profanity features are identified as important features,

[7] Since the usual acronym for Linear Discriminant Analysis (LDA) conflicts with that used for Latent Dirichlet Allocation, LnDA is rather used.

News24-large	
Manual Features	Authority score, percentage of profane words, lexical diversity, number of profane words, user PageRank, number of incorrectly spelled words, in degree of user, thread relevance, noun frequency, verb frequency.
Unigram	card, customer, data, double, enjoy, gb, get, message, mtn, note.
Bigram	buy mtn, dear customer, card go, enjoy enjoy, go message, new promo, note wait, receive message, serial number, say dear.
Trigram	card go message, go message box, last month note, mtn user u, message box type, message say dear, promo mtn user, receive message say, say dear customer, send dis mtn.
News24-small	
Manual Features	Noun frequency, absolute length of comment, percentage of incorrectly spelled words, number of pronouns, comment entropy, lexical diversity, number of incorrectly spelled words, capitalised sentence frequency, user PageRank, out degree of user.
Unigram	accurate, follow, gill, marcus, numerology prediction, racism, recruitment, sale, sh*t.
Bigram	accurate prediction, custom service, follow know, gill marcus, nigeria custom, numerology make, physical psychic, prince william, psychic twitter, step pravin.
Trigram	money grab industrialist, neglect industry even, nigeria custom service, over bunch hole, physical psychic twitter, point end game, prediction follow know, private enterprise either, run egostistical greedy, run ugly people.
Slashdot	
Manual Features	Out degree, hub score, noun frequency, absolute lengths, number of posts of user, comment entropy, lexical diversity, verb frequency, thread relevance, authority score.
Unigram	addons, apk, botnets, cow, dns, gaywad, moooooooo, packer, usermode, wigger.
Bigram	botnets stop, cow say, gay wigger, host file, host less, moo cow, moo say, packer mover, reliability protect, speed security.
Trigram	admin malwarebytes employee, apk host file, cow cow say, cow say moo, gay wigger association, host file engine, lol oh lol, moo say cow, say cow cow, wigger association dice.

Table 3.2: The most relevant manual features, unigrams, bigrams and trigrams.

while in the Slashdot data set, a few other post-features that have more to do with the comment's structure are identified.

3.6 Summary

This chapter discussed four main methods of feature extraction for internet comments used in this thesis. The first was a collection of features that are manually constructed to show various characteristics of the comments. Most of these features are closely related to features used by other researchers [180, 126, 143, 79], but some were specifically constructed for the task at hand. Second, various N-gram-based methods were discussed. These included word N-grams, character N-grams and character skip-grams. Thereafter, a feature construction technique based on topic modelling (LDA) was discussed. Finally, deep learning was used to create word and sentence embeddings that are used to construct features for comments. The chapter concluded with the most relevant features of some of the technique, as determined by linear discriminant analysis.

Chapter 4

Methodology

The purpose of this chapter is to discuss the methodology followed in this thesis in terms of the SVM techniques.[1] Next, the feature pre-processing techniques for training a predictor will be introduced. Finally, the details of the hyper-parameter training for the SVM models are discussed.

4.1 Quality Prediction Pipeline

All of the constructed feature sets (the hand-crafted features, N-gram features, LDA features and distributed representations), as discussed in Chapter 3, are used to train SVM models [176, 45], which are then evaluated and compared by using standard metrics.

The training data consist of labelled data points which are transformed into feature vectors with an associated label. The choice of features to use from the training data depends on the domain of the data, as well as the relevance of the features [88]. In this case, the training data sets will consist of N comments, denoted as $\{c_1, c_2, ..., c_N\}$. For each comment c_i, a set of m features $F_i = \{f_1(c_i), f_2(c_i), ..., f_m(c_i)\}$ is extracted. Thus, a candidate feature set consists of rows of the form $\{(F_1, r_1), ..., (F_N, r_N)\}$, where a tuple (F_i, r_i) indicates a feature set F_i for comment c_i, and the associated class value r_i.

The training data must be transformed in various ways to be suitable as training data for the SVC predictor. This is shown in Figure 4.1 and detailed in the sections that follow.

The source data was either the raw database files provided by News24, or the raw data obtained via scraping of the Slashdot website. Then, usable data sets (in the form of `pandas` data frames [110]) are extracted from the raw data. Thereafter the data is split into training and testing sets using stratified sampling (i.e. the class distribution is maintained through the

[1] This chapter contains portions of work from a previous paper co-authored with one or more of my co-supervisors [26].

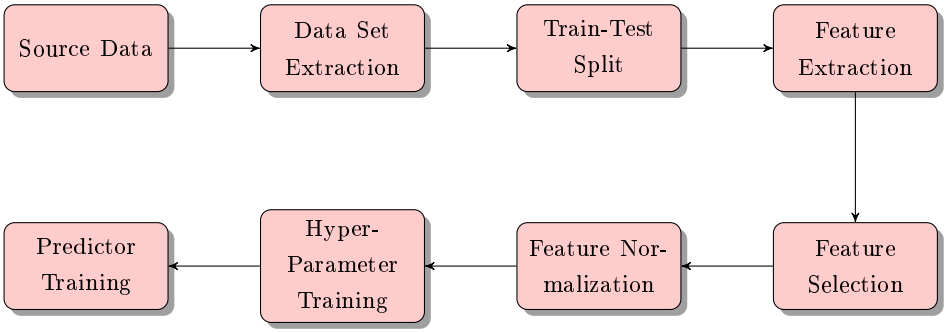


Figure 4.1: The pipeline that a comment goes through to be classified.

split). The training and test sets make up 60% and 40% of the data set, respectively. After the partitioning, feature extraction is performed according to the various feature extraction methods discussed in Chapter 3. For the N-gram models, the training set is used for training the model which is then used to transform the test set.

After the feature extraction step, feature selection is done to reduce the dimension of the feature set. This is necessary due to constraints on the time required to train the SVM models. The features are then normalized which has been shown to lead to improved performance in SVMs with a linear or RBF kernel [72, 80, 67]. The normalized training features are then used for tuning the SVM hyper-parameters through cross-validation. Finally, an SVC model is trained (with the parameters determined through cross-validation) to predict the class value of a training feature vector for each data set. The classifier is then evaluated with various metrics using the test set. Finally the trained classifier can be used to predict values for unlabelled feature vectors, representing new comments.

The RBF and linear kernels [85] for SVC will be considered by using the SVM classifier from the `Scikit-Learn` Python library [131]. These are very common kernels, and previous studies on comment classification have also made use of them [79]. These kernels are also attractive for N-gram techniques, since they can efficiently deal with the sparse matrices induced by our N-gram representations.

4.2 Feature Sets

Various feature sets are constructed to be used for training SVM models. The following list summarizes the feature set types with their respective notation, some of which are described in Section 3.2:

- manual features (M),
- N-gram models:

  - binary word features ($= 1B$),
  - frequency word features ($= 1F$),
  - TF-IDF-normalized word features ($= 1T$),
  - bigram term features ($= 2B$, $= 2T$, $\leq 2B$, $\leq 2T$),
  - trigram term features ($= 3B$, $= 3T$, $\leq 3B$, $\leq 3T$),
  - character N-gram features ($C28B$, $C28T$), and
  - character skip-gram features (CSB, CST).
- LDA topic model (LDA),
- distributed representations based on deep neural networks:
  - `word2vec` models:
    - TF-IDF weighted sum representation ($W2V\text{-}TWS$), and
    - bag-of-centroids representation ($W2V\text{-}BOC$).
  - Google News model:
    - TF-IDF weighted sum representation ($Go\text{-}TWS$).

Preliminary experiments on the large News24 data set showed that comments with fewer than 20 words are hard to classify with the methods described in this thesis and are, as such, not included in the large News24 data set used here. They are, however included in the small data set as the data set was initially compiled from comments that were both shorter and longer than 20 words. The Slashdot data set was modified by removing anonymous comments and comments with a score of '2' (as discussed in Section 2.3). This modified data set contains 70393 comments.

Thus, to summarise, the large News24 data set consists of 79017 comments (each of length 20 or more), the small News24 data set of 4796 comments, and the Slashdot data set of 70393 comments by registered users with scores not equal to '2' (all before train-test splitting). Table 4.1 shows the number of features per training set for each feature representation (before dimensionality reduction). Here it is clearly shown how the dimensionality of the feature sets explode with higher-order N-gram representations. The LDA and `word2vec` feature sets are all fixed in size.

4.3 Feature Preprocessing

The aim of the preprocessing steps is to transform the input to conform more to the assumptions made when training SVMs. Normalizing the data leads to improved performance and, for computational efficiency, it is sensible to select only a certain subset of the features before training a classifier. These steps are done on both the training and test feature sets.

Table 4.1: The number of features of each feature set.

	Number of Features		
Feature Representation	News24 Large	News24 Small	Slashdot
manual	33	33	33
$= 1B, = 1F, = 1T$	43541	10807	24871
$= 2B, = 2T$	583779	64495	1064278
$= 3B, = 3T$	834832	71376	1638838
$\leq 2B, \leq 2T$	627320	75302	1114020
$\leq 3B, \leq 3T$	1462152	146678	2752858
$C28B$, $C28T$	4627585	955447	6620597
CSB, CST	484006	104611	832488
LDA	100	100	100
W2V-TWS	200	200	200
W2V-BOC	2082	2082	838
Go-TWS	300	300	300

4.3.1 Dimensionality Reduction

Reducing the dimensionality of the feature space often helps to reduce noise in the data by removing irrelevant and redundant features [170]. This can result in better performance for many regression and classification models [69], as well as improve the generalizability of a model. The computational complexity of some regression and classification models are also dependent on the dimensionality of the training data, so dimensionality reduction also leads to faster model training.

Dimensionality reduction comes in two forms, namely auxiliary feature construction and feature selection. The former involves constructing alternative features by combining old features in order to lower the dimensionality of the feature set [170]. Two popular approaches are Principal Component Analysis (PCA) [86] and Linear Discriminant Analysis (LDA) [137]. A major disadvantage of auxiliary feature construction is that the combined features are hard to interpret.

Feature selection [30] involves selecting a subset of the original features to maximize relevance to the class variable. The advantage of using feature selection as opposed to constructing additional features, is that results are much easier to interpret [170]. This is an important consideration in this project as it can help us understand which features (in the case of the custom features) or which terms (in the case of the word features) are the most relevant to our classification task.

To identify the most relevant features, a popular type of feature selection, called univariate feature selection, is applied [101]. This involves looking at each feature individually (and at its relationship to the class variable) and

scoring its contribution to the classification strength of the model. The scoring mechanism used is the χ^2 statistical test [65].

The top 50% of features are selected for the baseline manual feature set. For the N-gram representations, all the features are kept, up to a maximum of 200000 features. This is done due to constraints on the time required to train the classifiers, as the training complexity of an SVM is directly correlated to both the number of samples and the number of features (for both linear and RBF kernels) [24]. It was determined experimentally that this arbitrary maximum does not significantly impact the results, but provides a substantial decrease in running time of experiments.

4.3.2 Feature Normalization

The range of values determined by the features in Chapter 3 varies widely. Features in greater numerical ranges can dominate other features in smaller ranges, but more importantly, noise in larger features can dominate signal in smaller features [80, 127].

Feature scaling, in most cases, involves manipulating each component of each observation in the feature set as follows:

$$X^* = \frac{X - \mu}{\sigma}$$

where X is the value of the component, and μ and σ are the empirical mean and standard deviation of that component for all observations in the feature set. This can be computationally problematic for sparse feature sets such as the N-gram representations, since subtracting the mean in such a case will typically shift many values away from zero, thus losing the sparsity in the original representation. For this reason, an alternative approach, called feature normalization, is used. In feature normalization, each *feature* is individually scaled so that its l^2 norm (i.e. the sum of the squared values for that feature over all the samples) equals 1. Thus, feature values of zero are unaffected, which is advantageous for maintaining sparseness.

4.4 Hyper-Parameter Estimation

Techniques such as SVMs are often highly sensitive to the choice of parameters [41]. A grid search is used to tune the parameters for each feature set being used with the RBF or linear kernel. Grid search has been shown to be a very efficient and accurate technique for hyper-parameter tuning [14].

The cost parameter C, as well as the bandwidth parameter γ for the RBF kernel, was tuned. The search was done using values spread over a logarithmic scale (10^{-1} to 10^7 for C and 10^{-4} to 10^1 for γ). For the large News24 data set, a random subset of 10265 samples from the training set (25% of the training samples) is used as a validation set for cross-validation.

Since the experiments are very time-consuming, parameter tuning with cross-validation on a larger data set was not feasible. The entire small News24 training set is used, since it is small enough. Similar to the large News24 data set, a random selection of 14999 samples are selected from the Slashdot data set. For all of the parameter tuning experiments, a 3-fold cross-validation is done and the results compared to select the best parameters. The resulting parameters for each model are given in Table 4.2.

4.5 Spam Detection

An alternative to the classification approaches discussed for the News24 data sets, is a simple naïve filtering approach where a spam detection classifier is built to filter comments that are labelled as spam, since one of the criteria for labelling News24 comments as hidden, is that the comment contains advertising or spam.

For this task, a spam classifier was trained on public email data sets containing both spam and non-spam emails. The data sets that were used were the Enron-Spam data set (as described in a paper by Metsis et al. [111]) and the SpamAssassin public corpus [82]. These data sets consist of various rows of text (obtained from emails) that are labelled as either spam or non-spam. Each row of text is converted into a bag-of-N-grams (unigrams and bigrams) representation (as discussed in Chapter 3) and then TF-IDF-normalized. A Naïve Bayes classifier is then trained on this input data and then used to classify texts as either spam or non-spam. This methodology is derived from an approach by Radim Rehurek [138]. The classifier achieves an accuracy of 0.992 on the combined data set of 55326 emails (where roughly 40% are non-spam and 60% are spam).

This classifier is used in the investigations in Chapter 5 to determine whether a comment is spam or not, as an alternative to SVM with the other feature extraction techniques.

4.6 Summary

This chapter examined the methodology used for the experiments in this thesis. A pipeline was presented that shows the processes and transformations the data sets undergo to become training and testing sets for the prediction models. The techniques for feature selection, feature normalization and hyper-parameter tuning were also introduced. A baseline spam detection approach was also introduced as a naïve method for predicting comment quality. The next chapter introduces the experiments that follow the methodology discussed in this chapter.

	News24 Large			News24 Small			Slashdot		
Feature Set	SVC (RBF)		SVC (Lin)	SVC (RBF)		SVC (Lin)	SVC (RBF)		SVC (Lin)
Parameter	C	γ	C	C	γ	C	C	γ	C
manual	10^4	10^{-3}	10^4	10^{-1}	10^{-1}	10	10^6	10^{-4}	10^3
$= 1B$	10	10^{-2}	10^{-1}	10^2	10^{-2}	1	10^3	10^{-1}	10^3
$= 1F$	10	10^{-3}	10^{-1}	10^2	10^{-2}	1	10^3	10^{-1}	1
$= 1T$	10	10^{-1}	1	10^{-1}	10	1	10^4	1	10^2
$= 2B$	10^3	10^{-3}	1	10^3	10^{-1}	1	10^2	10^{-2}	10
$= 2T$	10	10^{-1}	10^2	10^4	10^{-1}	10^4	10^{-1}	1	10^2
$= 3B$	10^{-2}	10^{-1}	1	10	10^{-1}	1	10	10^{-1}	1
$= 3T$	10^{-1}	10^2	10^3	10	10	10^3	10^2	1	10^3
$\leq 2B$	10^6	10^{-3}	10^{-1}	10^3	10^{-1}	1	10^2	10^{-3}	10^3
$\leq 2T$	10^2	10^{-1}	10	10^4	10^{-1}	10^4	1	1	10^4
$\leq 3B$	10^2	10^{-3}	10^{-1}	10	10^{-1}	1	10^2	10^{-3}	10^4
$\leq 3T$	10	10^{-1}	10^4	10	10	10^3	1	10^{-1}	10^4
$C28B$	10	10^{-4}	10^{-3}	10	10^{-3}	10^{-1}	10^3	10^{-3}	10^{-2}
$C28T$	10	1	10	10^2	10^{-1}	10^2	10	1	10^2
CSB	10	10^{-3}	10^{-2}	10^2	10^{-3}	10^{-1}	10	10^{-3}	10^{-1}
CST	10^4	10^{-4}	1	10^2	10	10^2	10	1	10
LDA	10^6	10	10^{-3}	10^7	1	1	10^6	10^{-2}	10
$W2V - TWS$	10^4	10^{-4}	10^4	10	10^{-2}	10^{-2}	10	10^{-2}	10^2
$W2V - BOC$	10^7	10^{-4}	10^4	10^3	10^{-4}	10^{-1}	10^6	10^{-4}	10^3
$Go - TWS$	10	10^{-3}	10^{-2}	10	10^{-3}	1	10^7	10^{-4}	10^4

Table 4.2: Parameters obtained through parameter tuning.

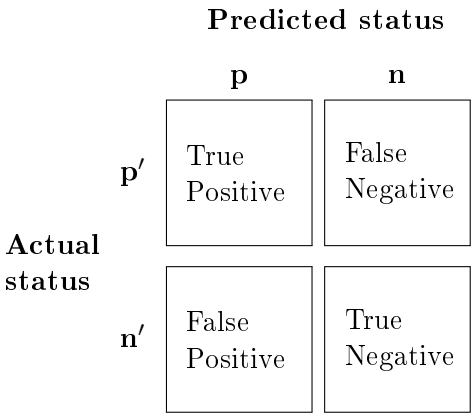
Chapter 5

Results

This chapter describes the methods of evaluating the various classification models and the feature construction methods, as introduced in this thesis, as well as evaluate these methods and discuss the results.[1] The predictive performance of the classification models is evaluated using standard performance metrics.

5.1 Evaluation Metrics

With regards to the classification task, the main goal is to identify "bad" comments, so "bad" comments are assumed to be a positive observation in the schemes discussed below (i.e. p and p′) for all three data sets. The labelling scheme is summarised in the following diagram:



The *accuracy* of a classifier is defined as the proportion of correct predictions it makes (Equation 5.1). This is a simple measure for estimating the quality of a classification. However, when class membership is imbalanced, a classifier could still achieve high accuracy if it succeeds at classifying one

[1] This chapter also contains results previously presented in [26].

class, but completely fails to classify the other (e.g. if 80% of the comments are good, classifying all comments as good would lead to a relatively high accuracy of 80%).

Thus, accuracy alone is not enough to determine the quality of a classifier, so two other metrics are used: sensitivity and specificity. *Sensitivity* is the proportion of bad comments that are identified as such (Equation 5.2). Conversely, *specificity* is the proportion of good comments identified as good comments (Equation 5.3). Accuracy can further be formulated as a weighted average between sensitivity and specificity, as is shown in Equation 5.1 where α is equal to the proportion of samples that have positive labels ($\frac{p}{p+n}$). α could also be the relative weight associated with sensitivity/specificity, based on the cost associated with making positive and negative predictions.[2] We unfortunately do not have any information about the cost of predictions for any of the data sets, so we simply take *alpha* as the ratio of positive to negative samples. This means that the accuracy score would favour the majority class in our experiments. Another simpler metric is thus used: the reweighted average between sensitivity and specificity (i.e. $\frac{\text{Sensitivity}+\text{Specificity}}{2}$). This captures the probability of a given comment (irrespective of class imbalance) being classified as either "good" or "bad". This metric is easily interpretable in that a value lower than 0.5 would indicate that a naïve guess would achieve better predictions and a high value would indicate a general measure of quality.

Two other standard metrics used to evaluate binary classifiers, are precision and the F1 score [176]. These metrics, however, are sensitive to which class is used as the positive class (i.e. they are asymmetric), so they are not used.

$$\begin{aligned}\text{Accuracy} &= \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{TN}+\text{FN}} \\ &= \alpha \times \text{Sensitivity} + (1-\alpha) \times \text{Specificity}\end{aligned} \tag{5.1}$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{5.2}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}} \tag{5.3}$$

For any classification task, there is usually a trade-off between sensitivity and specificity. A perfect classifier would achieve 100% sensitivity and 100% specificity. For the task at hand (identifying bad comments), high sensitivity would typically be more valuable than high specificity because the original goal was to flag bad comments, thus making false negatives more costly.

[2] If α represents some cost associated with making predictions, the formula no longer represents the accuracy of predictions.

Data Set	Baseline Accuracy
News24 Large	0.679
News24 Small	0.561
Slashdot	0.644

Table 5.1: Baseline accuracy for classifying all samples as the majority class for the various data sets.

Data Set	Accuracy	Sensitivity	Specificity	Average
News24 Large	0.38	0.85	0.15	0.50
News24 Small	0.46	0.79	0.20	0.50
Slashdot	0.46	0.64	0.36	0.50

Table 5.2: Results for the spam detection classifier.

In addition, for each of the data sets, a baseline accuracy is reported in Table 5.1 which is based on classifying all the samples as the majority class (i.e. without any intelligent prediction scheme). In the case of the News24 data sets, the majority class is good comments, and in the case of the Slashdot data set, the majority class is low ranking comments. All reported accuracy scores should be viewed relative to the baseline accuracy.

5.2 News24 Results

Table 5.2 presents the evaluation scores of using the simple spam detection classifier (as discussed in Section 4.5) on the three data sets. Table 5.3 and 5.5 present the performance scores of the RBF kernel SVM classifier trained on various feature sets generated from the large and small News24 data sets respectively. Similarly, Tables 5.4 and 5.6 present the performance scores for the linear kernel SVM classifier.

The results of the spam classifier on the News24 and Slashdot data sets, as shown in Table 5.2, confirm that a naïve spam classifier is insufficient for classifying the comments. It seems to be overly aggressive in labelling comments as spam, which yields a high sensitivity, but at the cost of specificity and subsequently accuracy.

Table 5.3 shows that nearly all the N-gram representations outperform the manual feature set on all metrics. There seems to be little difference between the results for the binary and the TFIDF vectorization methods, although it seems that the TFIDF models perform slightly worse for unigram,

Feature Set	Accuracy	Sensitivity	Specificity	Average
manual	0.836	0.641	0.928	0.784
$= 1B$	0.870	0.707	0.946	0.827
$= 1F$	0.863	0.671	0.953	0.812
$= 1T$	0.869	0.692	0.953	0.812
$= 2B$	0.855	0.676	0.939	0.808
$= 2T$	0.855	0.681	0.937	0.809
$= 3B$	0.845	0.666	0.929	0.829
$= 3T$	0.818	0.519	0.960	0.739
$\leq 2B$	0.861	0.702	0.936	0.819
$\leq 2T$	0.845	0.739	0.896	0.817
$\leq 3B$	0.856	0.677	0.941	0.809
$\leq 3T$	0.638	**0.818**	0.552	0.685
$C28B$	0.871	0.679	0.961	0.820
$C28T$	0.865	0.711	0.938	0.825
CSB	**0.874**	0.696	0.959	**0.828**
CST	0.873	0.686	0.962	0.824
LDA	0.561	0.329	0.671	0.500
W2V-TWS	0.791	0.746	0.813	0.780
W2V-BOC	0.597	0.227	0.772	0.500
Go-TWS	0.671	0.016	**0.981**	0.499

Table 5.3: Results for RBF SVM classification on the large News24 data set.

bigram and trigram models. We get a reasonable gain in sensitivity by adding higher-order N-grams to the unigram models, with the model that combines unigrams, bigrams and trigrams obtaining the highest overall sensitivity, albeit at the cost of accuracy and specificity. The model that combines only unigrams and bigrams shows a significant gain in sensitivity without too much loss in specificity. This indicates that, for the large News24 data set, adding bigrams to unigrams increases the strength of the model.

The naïve character N-gram models, for the most part, did as well as or better than the word N-gram models. The binary character skip-gram model had the highest overall accuracy. The `word2vec` bag-of-centroids and Google TF-IDF models struggled to classify 'bad' comments. The `word2vec` models might be useful, but they need refinement.

Table 5.4 again shows that the N-gram models, for the most part, outperform the manual feature sets. The results further show an increase in sensitivity when higher-order N-grams are used to augment the unigram feature sets. The feature set that combines unigrams and bigrams obtained the highest overall accuracy and sensitivity-specificity average.

Again, the character N-gram models show promise, with the binary character N-gram model achieving the highest specificity score. For the distributed models, the `word2vec` model trained on the News24 corpus performed slightly better (in terms of accuracy and specificity) than the pre-trained Google corpus model. The `word2vec` model features also performed relatively well in comparison to the N-gram model features, unlike with the RBF kernel.

Tables 5.5 and 5.6 show the results for classification on the small News24 data set. The results are worse overall than that of the large News24 data set, which is to be expected considering how much fewer data points there are. Nonetheless, we highlight a few of the interesting results.

Again, the unigram feature sets achieve higher accuracy than the manual feature set. The deteriorating performance of higher order N-gram representations (as observed with the large News24 feature sets) is much greater in the results for the small News24 data set. The highest accuracy in Table 5.6 is achieved by the N-gram feature set where unigrams and bigrams are combined, indicating that augmenting the unigram feature set with higher order N-grams sometimes results in better performance (also confirmed by the gain in accuracy achieved in Table 5.5), but the results also indicate that higher order N-grams on their own struggle to classify accurately.

In Table 5.5, the `word2vec` model trained by Google with TFIDF shows the highest overall sensitivity. For the linear SVM classifier, the `word2vec` models all performed relatively well, with the `word2vec` model with the TFIDF-weighted sum showing the highest sensitivity-specificity average.

Feature Set	Accuracy	Sensitivity	Specificity	Average
manual	0.830	0.647	0.917	0.782
$= 1B$	0.869	0.690	0.954	0.821
$= 1F$	0.869	0.694	0.952	0.823
$= 1T$	0.865	0.677	0.954	0.816
$= 2B$	0.855	0.678	0.938	0.808
$= 2T$	0.850	0.686	0.927	0.807
$= 3B$	0.855	0.668	0.943	0.806
$= 3T$	0.855	0.671	0.941	0.806
$\leq 2B$	**0.876**	0.709	0.942	**0.826**
$\leq 2T$	0.866	0.715	0.937	**0.826**
$\leq 3B$	0.867	0.705	0.944	0.825
$\leq 3T$	0.742	**0.775**	0.726	0.751
$C28B$	0.866	0.654	**0.966**	0.810
$C28T$	0.869	0.667	0.964	0.816
CSB	0.873	0.679	0.965	0.822
CST	0.846	0.714	0.908	0.811
LDA	0.839	0.596	0.954	0.775
W2V-TWS	0.848	0.631	0.950	0.791
W2V-BOC	0.553	0.349	0.650	0.500
Go-TWS	0.837	0.644	0.928	0.786

Table 5.4: Results for linear SVM classification on the large News24 data set.

Feature Set	Accuracy	Sensitivity	Specificity	Average
manual	0.618	0.228	0.923	0.576
$= 1B$	0.621	0.511	0.707	0.609
$= 1F$	0.626	0.528	0.703	0.616
$= 1T$	0.665	0.559	0.748	0.631
$= 2B$	0.579	0.050	0.993	0.521
$= 2T$	0.609	0.326	0.830	0.578
$= 3B$	0.578	0.045	0.996	0.521
$= 3T$	0.577	0.039	**0.998**	0.519
$\leq 2B$	0.633	0.250	0.933	0.592
$\leq 2T$	0.671	0.452	0.842	0.647
$\leq 3B$	0.601	0.114	0.983	0.549
$\leq 3T$	0.667	0.388	0.886	0.637
$C28B$	**0.684**	0.549	0.789	**0.669**
$C28T$	0.677	0.548	0.777	0.663
CSB	0.648	0.562	0.716	0.639
CST	0.680	0.568	0.768	0.668
LDA	0.573	0.513	0.620	0.567
$W2V\text{-}TWS$	0.593	0.092	0.985	0.539
$W2V\text{-}BOC$	0.638	0.546	0.711	0.629
$Go\text{-}TWS$	0.662	**0.578**	0.727	0.653

Table 5.5: Results for RBF SVM classification on the small News24 data set.

Feature Set	Accuracy	Sensitivity	Specificity	Average
manual	0.638	0.513	0.736	0.625
$= 1B$	0.657	0.519	0.767	0.643
$= 1B$	0.623	0.541	0.688	0.615
$= 1F$	0.613	0.533	0.676	0.605
$= 1T$	0.643	0.569	0.701	0.635
$= 2B$	0.600	0.162	0.942	0.552
$= 2T$	0.592	0.659	0.539	0.599
$= 3B$	0.581	0.050	**0.997**	0.524
$= 3T$	0.580	0.052	0.993	0.523
$\leq 2B$	**0.677**	0.489	0.824	0.657
$\leq 2T$	0.662	0.503	0.786	0.645
$\leq 3B$	0.663	0.387	0.879	0.633
$\leq 3T$	0.663	0.438	0.840	0.639
$C28B$	0.657	0.568	0.730	0.649
$C28T$	0.681	0.543	0.789	0.666
CSB	0.637	0.559	0.698	0.629
CST	0.656	0.582	0.715	0.649
LDA	0.607	0.589	0.621	0.605
$W2V\text{-}TWS$	0.672	0.649	0.689	**0.669**
$W2V\text{-}BOC$	0.646	0.625	0.662	0.644
$Go\text{-}TWS$	0.622	**0.654**	0.596	0.625

Table 5.6: Results for linear SVM classification on the small News24 data set.

5.3 Slashdot Results

The investigation of automatic quality prediction for Slashdot comments can either be framed as a multi-class classification task or an ordinal regression task, since there are 7 distinct scores assigned to comments, so treating these scores as continuous values and solving an ordinal regression problem also seems appropriate. Preliminary experiments showed that we are unable to train a linear or non-linear regression model to adequately predict these continuous labels better than by simply predicting the mean score. This is most likely due to the skewness in the Slashdot data, as around 63% of the comments are labelled as '2'. Through cross-validation and hyper-parameter tuning, the highest R^2 value [53] we were able to obtain was 0.04 with a non-linear SVM regression model, which is barely better than simply assigning a score of '2' to each test sample.

To address the class imbalance issue, a second experiment was designed where the classes were sub-sampled to form a data set with equal number of data points in each of the classes. Again, through cross-validation and hyper-parameter tuning the highest R^2 value we obtained was 0.11 with a linear regressor. This R^2 score is still unsatisfactory.

Finally, we decided to group scores together to transform the task into a classification task. Table 5.7 shows the accuracy, sensitivity and specificity for the three comment labelling strategies, as well as the baseline accuracy (labelling everything as the majority class) for each strategy. Each strategy uses an SVM classifier with the RBF kernel. Also, based on the results of the News24 classification experiments, the most viable feature extraction method (in terms of quality of results and how relatively fast training and testing are) was shown to be the binary unigram model, so this was used as an input feature set to evaluate the three labelling strategies.

The first labelling strategy is to label all comments with a score higher than 2 as 'high' and the rest as 'low', effectively making the problem a two-class classification problem, similar to the News24 comment classification problem addressed in this thesis. This technique still suffers from some class imbalance, since the majority class in this case would contain 75% of the samples.

Second, comments are grouped into three classes, where comments with a score lower than 2 are labelled as 'low', comments with a score of exactly 2 as 'satisfactory', and comments with a score higher than 2 as 'high'. This labelling is similar to the labelling scheme used by Brennan et al. [27].

The final labelling strategy involves taking the two-class labelling approach, but leaving out comments that are labelled as 2, to reduce the class imbalance in the data. Thus, comments with a score lower than 2 are labelled as 'low' and comments with a score higher than 2 are labelled as 'high'.

Table 5.7 shows that the two-class strategy achieves the highest accuracy, however, the two-class strategy where samples labelled as '2' are left out

Labelling Strategy	Baseline Accuracy	Accuracy	Sensitivity	Specificity
Two-class	0.832	0.750	0.860	0.190
Three-class	0.449	0.570	0.410	0.710
Two-class without 2	0.645	0.690	0.900	0.190

Table 5.7: Results for SVM classification on the Slashdot data set with three different comment labelling strategies.

achieved a higher sensitivity without decreased specificity, with the added benefit of not being influenced by gross class imbalance. Also, it was able to achieve an accuracy closer to the baseline accuracy. Thus, we decided to evaluate the rest of the feature extraction techniques on a modified Slashdot data set where comments labelled as '2' are removed. Although this means that the data set is not representative of the real Slashdot comment base, the evaluation should give insight into the strength of our various feature construction techniques without the problem of gross class imbalance.

Tables 5.8 and 5.9 show the results for both RBF and linear kernel SVMs, respectively, trained with the various feature construction techniques performed on the modified Slashdot data set.

Considering the RBF SVM results in Table 5.8, the models all struggled to identify 'low-scoring' comments, with the highest overall sensitivity obtained being 0.493 when using the manually constructed feature set. What is surprising, is that all the RBF-kernel models failed to achieve a higher accuracy or sensitivity than the manual feature set, which achieved the highest sensitivity-specificity average by a large margin. Most of the features that we used for the manual feature set were formed from techniques found in papers by other authors, some of which studied Slashdot comments specifically, indicating perhaps that the borrowed techniques work especially well for Slashdot comments, especially when compared to the other techniques we proposed.

The character N-gram and skip-gram models were able to obtain slightly higher sensitivity than the word N-gram models, which again shows that the character N-gram and skip-gram models provide stronger representations for comments.

The linear SVM classification results (Table 5.9) clearly show the diminishing returns (in terms of sensitivity) when higher-order N-grams are used. What is interesting, and different from the other results, is that we obtain higher sensitivity when TFIDF frequency is used over binary vectorization. This indicates that the models benefit from normalization, which could indi-

Feature Set	Accuracy	Sensitivity	Specificity	Average
manual	**0.757**	**0.493**	0.903	**0.698**
$= 1B$	0.639	0.284	0.834	0.559
$= 1F$	0.643	0.247	0.861	0.554
$= 1T$	0.621	0.391	0.747	0.569
$= 2B$	0.594	0.129	0.850	0.490
$= 2T$	0.630	0.159	0.890	0.525
$= 3B$	0.595	0.131	0.851	0.491
$= 3T$	0.617	0.098	0.902	0.500
$\leq 2B$	0.642	0.057	0.965	0.511
$\leq 2T$	0.617	0.224	0.833	0.529
$\leq 3B$	0.645	0.048	0.973	0.511
$\leq 3T$	0.636	0.122	0.918	0.520
$C28B$	0.640	0.362	0.793	0.578
$C28T$	0.631	0.296	0.815	0.556
CSB	0.645	0.309	0.830	0.570
CST	0.643	0.331	0.815	0.573
LDA	0.586	0.383	0.699	0.541
$W2V\text{-}TWS$	0.654	0.168	0.921	0.545
$W2V\text{-}BOC$	0.645	0.000	**1.000**	0.500
$Go\text{-}TWS$	0.546	0.363	0.647	0.505

Table 5.8: Results for RBF SVM classification on the Slashdot data set.

Feature Set	Accuracy	Sensitivity	Specificity	Average
manual	0.627	0.208	0.857	0.533
$= 1B$	0.541	0.235	0.709	0.472
$= 1F$	0.575	0.579	0.573	0.576
$= 1T$	0.590	0.512	0.633	0.573
$= 2B$	0.624	0.073	0.927	0.500
$= 2T$	0.632	0.112	0.918	0.515
$= 3B$	0.642	0.016	0.987	0.502
$= 3T$	0.635	0.042	0.961	0.502
$\leq 2B$	0.604	0.195	0.830	0.513
$\leq 2T$	0.619	0.311	0.789	0.550
$\leq 3B$	0.532	0.236	0.696	0.466
$\leq 3T$	0.635	0.042	0.961	0.502
$C28B$	0.590	0.578	0.597	0.588
$C28T$	0.612	0.311	0.777	0.544
CSB	0.596	**0.585**	0.602	**0.594**
CST	0.643	0.297	0.834	0.566
LDA	**0.645**	0.000	**1.000**	0.500
$W2V\text{-}TWS$	0.613	0.036	0.931	0.484
$W2V\text{-}BOC$	**0.645**	0.000	**1.000**	0.500
$Go\text{-}TWS$	0.603	0.096	0.883	0.484

Table 5.9: Results for Linear SVM classification on the Slashdot data set.

cate that certain features (or N-grams in this case) tend to dominate others more so than with the News24 data.

The character N-grams were again able to achieve slightly higher sensitivity, but at the cost of specificity, which in turn negatively impacted their accuracy.

5.4 Summary

The spam detection classifier performed poorly for all three data sets and all the proposed techniques were able to achieve a higher accuracy. With the two News24 data sets and the Slashdot data set with the RBF kernel, we where able to significantly outperform the baseline accuracies, but the linear kernel SVM trained with the Slashdot data set failed to achieve a higher accuracy. This could be due to the fact that the Slashdot data set has a more subtle scoring mechanism than the News24 data sets, making it a hard classification task that a linear kernel is simply ill-suited for.

When comparing the manual features, N-gram approaches, LDA and word embedding approaches, it is clear that N-gram approaches stood out as fairly reliable technique with good performance. On the large News24 data set, the N-gram approaches mostly fared better than other techniques, but it should be noted that we got diminishing returns with greater orders of N. The character N-grams fared especially well and managed to achieve some of the highest accuracy values among all the experiments. The LDA and bag-of-centroids (*W2V-BOC*) techniques often showed poor results and warrant further investigation.

In general, on all three data sets, the experiments where the RBF kernel was used either showed better results or similar results than where the linear kernel was used. An interesting trend with the large News24 data set, is that the linear kernel seems to do slightly better on some of the lower-dimensional feature sets (i.e. *LDA*, *W2V-TWS* and *Go-TWS*). This could be due to the fact that the RBF kernel has a greater capacity for overfitting than the linear kernel when the dimensionality of the feature set is low, but not when the feature set is higher-dimensional and sparse (e.g. the N-gram approaches), where the two kernels show comparable performance.

The manually constructed features performed relatively well on all data sets and all kernels. This indicates that manual features might be an adequate technique for feature extraction for comment classification, but still at the cost of manual design effort.

Chapter 6

Conclusion

The main problem considered in this thesis, is how to automatically predict the quality of short texts in the context of online comments. Support vector machines were chosen for both classification and regression, because of their versatility, high fault tolerance and generalizability from other problem domains.

The investigation was contextualised by three data sets, two provided by News24, a leading online news provider in Southern Africa, and one extracted from the website of Slashdot, an online news aggregator. The data sets are comprised of user-generated comments that have been made after an article was posted. The News24 comments are labelled by editors, since editors are able to label comments that contain certain banned words, are off-topic, or are considered destructive in nature, assigning a "hidden" status to them. The Slashdot comments are rated by a scheme that includes both user reputation scores and ratings by other users. This thesis therefore investigates methods for classifying comments based on a predicted quality label.

6.1 Research Question

In Chapter 1, the research question that forms the basis for this thesis was identified. The question was: **How do feature construction techniques based on N-gram models and distributed representation models fare against the leading manual feature construction approaches?**

To address this question, four tasks were identified. Each of these tasks have been completed and the resulting feature sets were used for training SVM models which were then evaluated in Chapter 5.

The results show that the simple spam classifier is inadequate for classifying News24 comments, which is to be expected, since only some of the News24 comments are known to be spam, but other non-spam comments are also labelled as "bad", for which the SVM classifier trained with manual

features is better suited. Tables 5.3 to 5.9 show the results for the SVM models trained with both RBF and linear kernels. The SVM model with RBF kernel, trained on the large News24 data set, was able to obtain a classification accuracy of 0.874, which is much greater than the 0.375 obtained by the spam detection model on the same data set. We also found that the manually constructed features are better-suited for the Slashdot data set than other approaches. This indicating that the manual features can be a viable approach to comment classification, but come at the cost of manual design effort.

The results further showed that the N-gram-based approaches, for the most part, outperformed the manual features on the accuracy, sensitivity and specificity metrics. The character N-grams and skip-grams showed great promise, with the character N-grams achieving the highest specificity with RBF SVM classification on the large News24 data set and the highest accuracy on the same data set when using the linear kernel SVM classifier. Thus, it can be concluded that character N-grams and skip-grams are well suited for representing short texts, even more so than word N-grams.

A problem with the N-gram-based approaches is that the resulting feature sets are quite large and sparse, which results in long training times and models that often overfit. Thus, we investigated Latent Dirichlet Allocation for topic extraction as a means to group words together into representative topics, thereby reducing the size of the feature set. In our results, unfortunately, the LDA method did not fare especially well for any of the three data sources.

Another alternative to the N-gram-based approaches was taken from techniques commonly used in language modelling and deep learning where words are represented as real-valued vectors of fixed length. Using these vectors, we could create dense distributed representations for entire comments, resulting in lower-dimensional feature sets which could be used for training a classification model. We further used these fixed-size word vectors to create clusters of words, which also serves as a technique to produce lower-dimensional feature sets. The experiments with the News24 data sets showed some cases where the distributed representation models performed as well or better than the N-gram approaches. The approaches did not fare as well on the Slashdot data set.

6.2 Future work

The techniques investigated in this thesis were customized for the problem of classifying comments. In the case of the N-gram model and deep learning techniques, the effect of using these techniques on other short text classification problems would be a valuable insight (especially with the character N-gram models, which showed promise).

The feature extraction models that were investigated, can be improved. Various smoothing techniques and contextual language models [121] could be considered to improve on the N-gram representation techniques. Another approach is to use more sophisticated bigrams and trigrams that identify certain lexical classes of terms (e.g. *adjective_ noun* or *adv_ verb* bigrams) in the hopes that these classes will reduce the noise in the bigram and trigram feature sets. Also, investigating syntactic N-grams could help to identify equivalent N-grams by considering different neighbourhoods for the elements contained in the N-grams.[1] These more sophisticated approaches to building N-grams are loosely inspired by the research of Sidorov et al. [156]. The LDA techniques for topic modelling could also be extended to use higher-order N-grams for topics.

When we considered the distributed representation models, we used fairly rudimentary techniques for combining word vectors for comments and in future would like to investigate more sophisticated techniques. In particular, an approach called *paragraph vector* has recently been discussed in the research literature [99]. We would also like to investigate using the approaches for calculating distributed representations for words in the context of higher-order N-grams, particularly for the bag-of-centroids approach.

Other future work is considering alternative approaches for short text classification. These approaches may include both alternative SVM kernels (e.g. string kernels [102]), as well as alternative classification techniques.

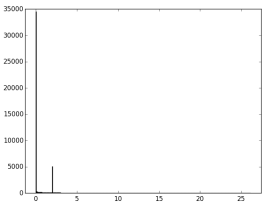
The discussion on deep learning techniques in Chapter 2 ends with the construction of features for comments based on the individual word vectors provided by the embedding models. Other techniques for end-to-end feature construction (e.g. autoencoders [9]) could also interesting, as they remove the need for a model training phase. An autoencoder could also be used to find a lower-dimensional representation for entire comments by using individual word representations as input/output for the network.

The word embedding approaches fail to express the semantic compositionality of entire comments. New approaches have been investigated [159, 186] to capture the recursive nature and natural phenomena of natural language. An investigation into the effects of using recursive neural networks for feature extraction could provide interesting insights, as it may be better able to take word order and composition of text into account.

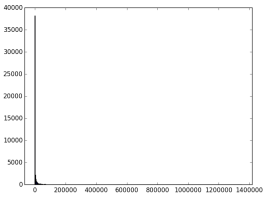
[1] Syntactic N-grams are N-grams where a word's neighbours are not defined as the words immediately before or after it, but rather the words before or after it in a syntactic tree, such as a parse tree.

Appendix A

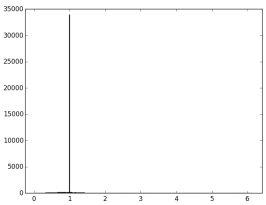
Manual Feature Graphs



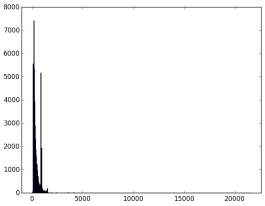
(i) Timeliness



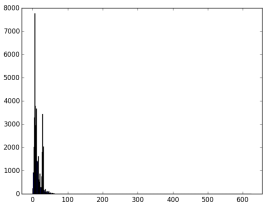
(ii) Time passed



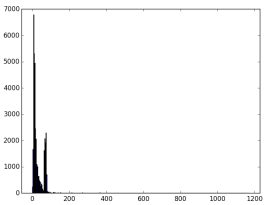
(iii) Lengthiness



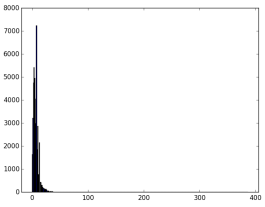
(iv) Number of characters



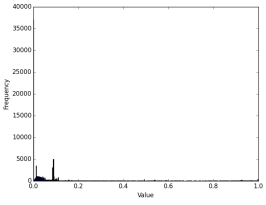
(v) Verb frequency



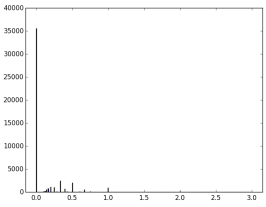
(vi) Noun frequency



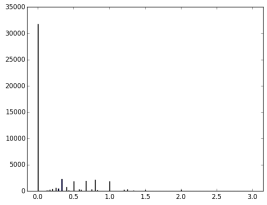
(vii) Pronoun frequency



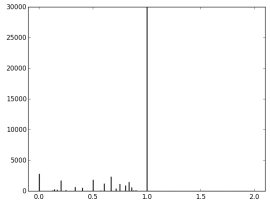
(viii) Capitalized word frequency



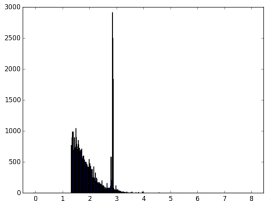
(ix) Question sentence frequency



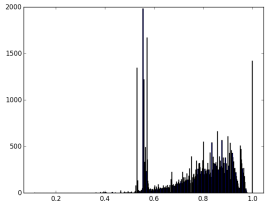
(x) Exclamation sentence frequency



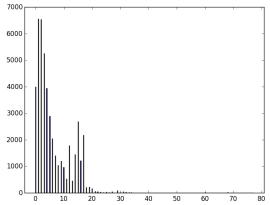
(xi) Capitalized sentence frequency



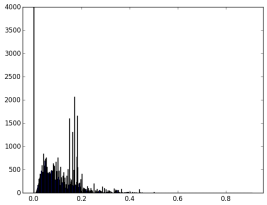
(xii) Comment complexity



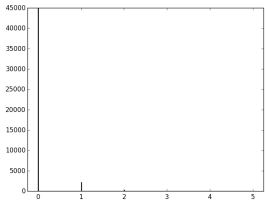
(xiii) Lexical diversity



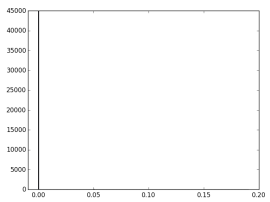
(xiv) Number of incorrectly spelled words



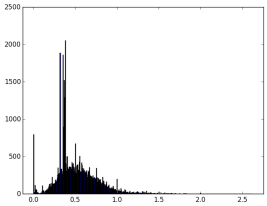
(xv) Percentage incorrectly spelled words



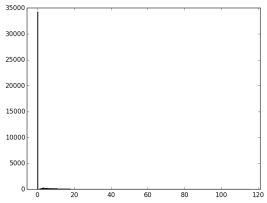
(xvi) Number of profane words



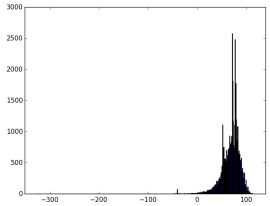
(xvii) Percentage profane words



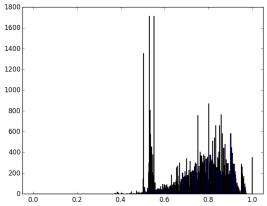
(xviii) Mean term frequency



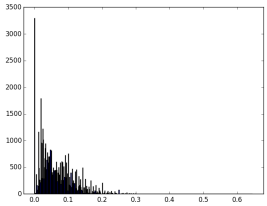
(xix) Informativeness



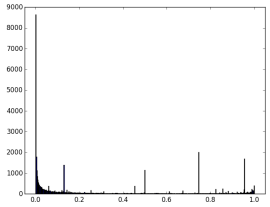
(xx) Readability



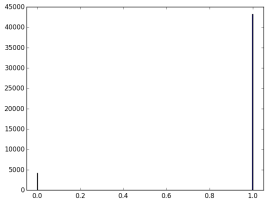
(xxi) Thread relevance



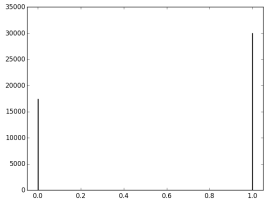
(xxii) Article relevance



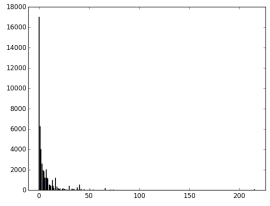
(xxiii) Percentage positive sentiment



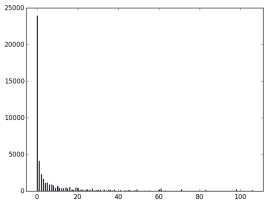
(xxiv) Subjectivity



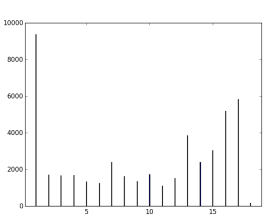
(xxv) Polarity overlap between comment and article



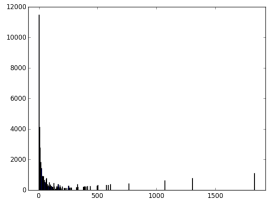
(xxvi) In degree of user



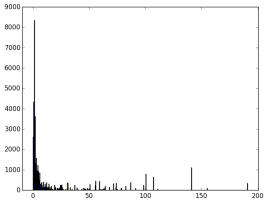
(xxvii) Out degree of user



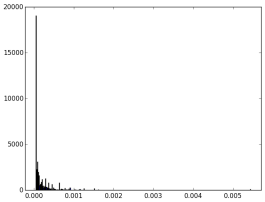
(xxviii) User age



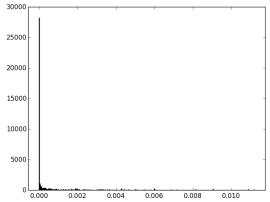
(xxix) Number of posts for user



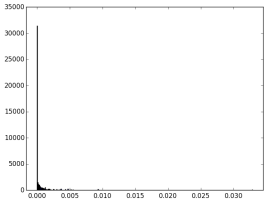
(xxx) User post rate



(xxxi) User pagerank

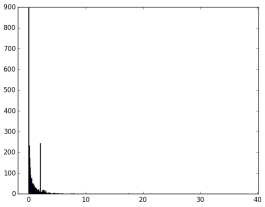


(xxxii) User hub score

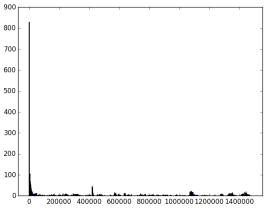


(xxxiii) User authority score

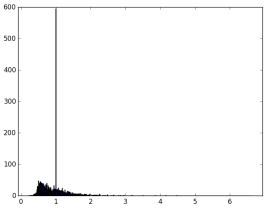
Figure A.1: Distribution of custom features for the large News24 data set.



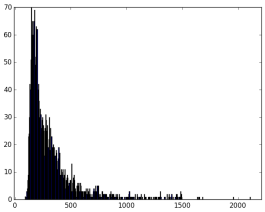
(i) Timeliness



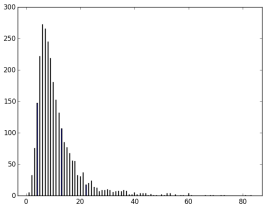
(ii) Time passed



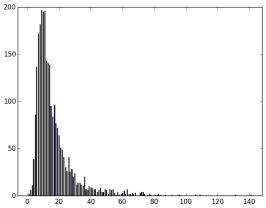
(iii) Lengthiness



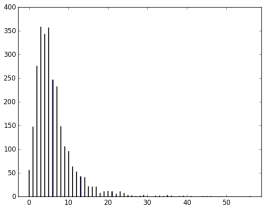
(iv) Number of characters



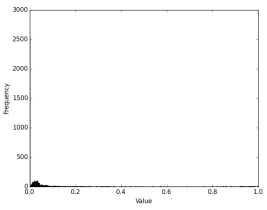
(v) Verb frequency



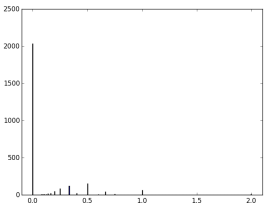
(vi) Noun frequency



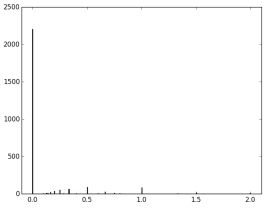
(vii) Pronoun frequency



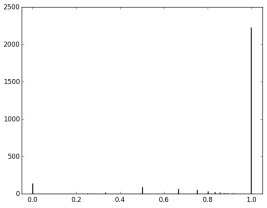
(viii) Capitalized word frequency



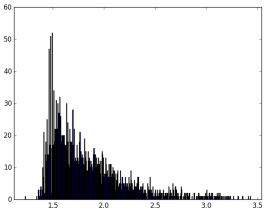
(ix) Question sentence frequency



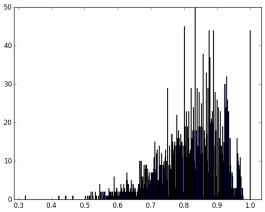
(x) Exclamation sentence frequency



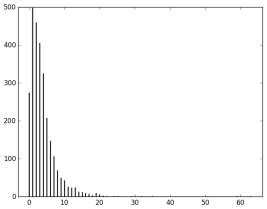
(xi) Capitalized sentence frequency



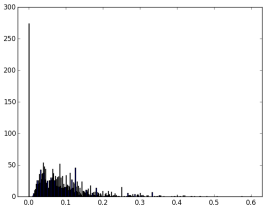
(xii) Comment complexity



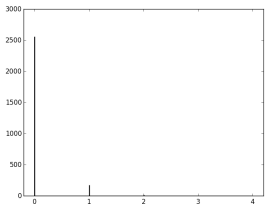
(xiii) Lexical diversity



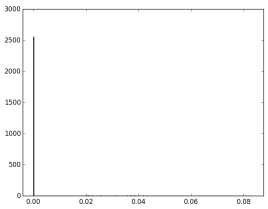
(xiv) Number of incorrectly spelled words



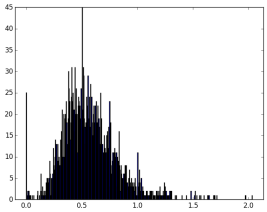
(xv) Percentage incorrectly spelled words



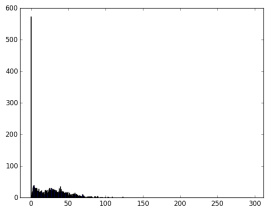
(xvi) Number of profane words



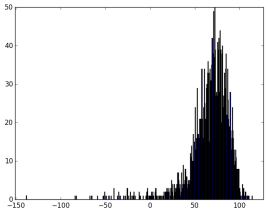
(xvii) Percentage profane words



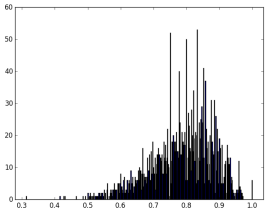
(xviii) Mean term frequency



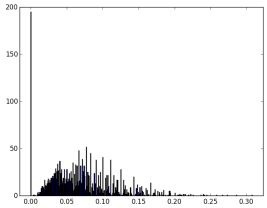
(xix) Informativeness



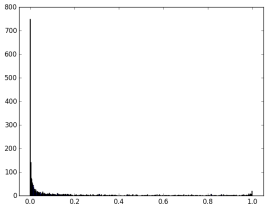
(xx) Readability



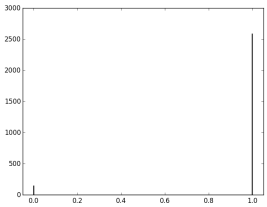
(xxi) Thread relevance



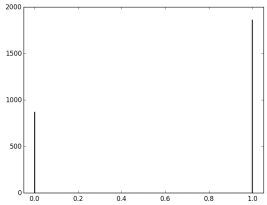
(xxii) Article relevance



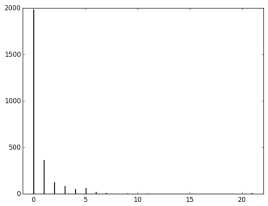
(xxiii) Percentage positive sentiment



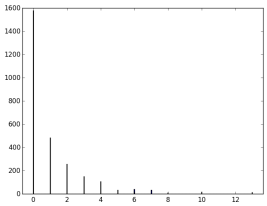
(xxiv) Subjectivity



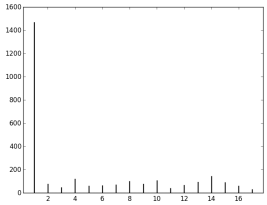
(xxv) Polarity overlap between comment and article



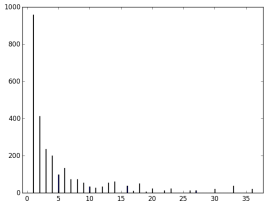
(xxvi) In degree of user



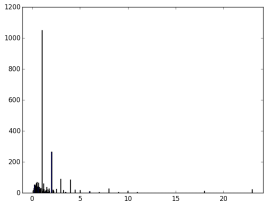
(xxvii) Out degree of user



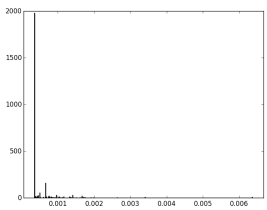
(xxviii) User age



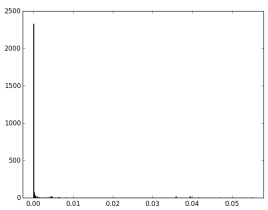
(xxix) Number of posts for user



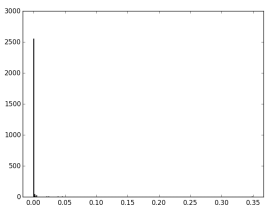
(xxx) User post rate



(xxxi) User pagerank

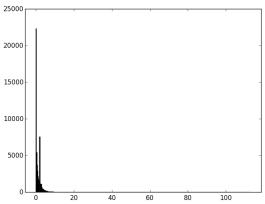


(xxxii) User hub score

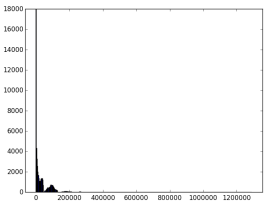


(xxxiii) User authority score

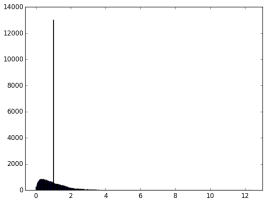
Figure A.2: Distribution of custom features for the small News24 data set.



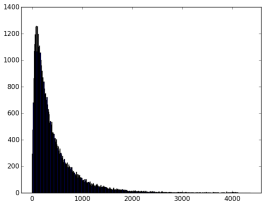
(i) Timeliness



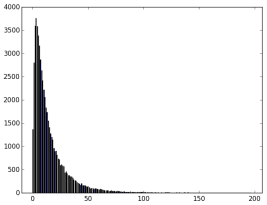
(ii) Time passed



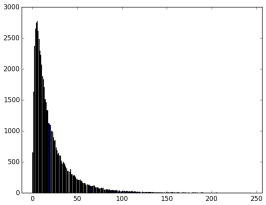
(iii) Comment lengthiness



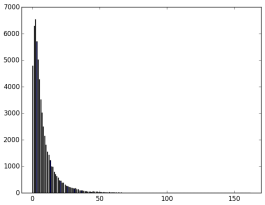
(iv) Number of characters



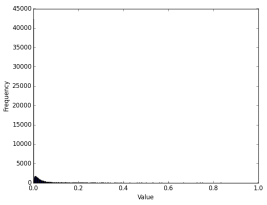
(v) Verb frequency



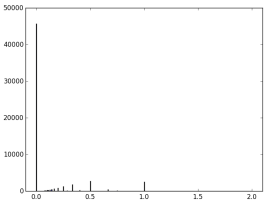
(vi) Noun frequency



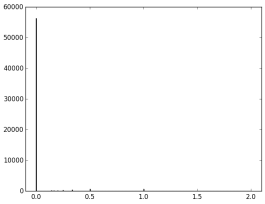
(vii) Pronoun frequency



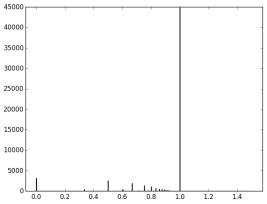
(viii) Capitalized word frequency



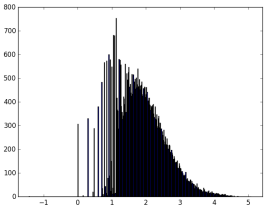
(ix) Question sentence frequency



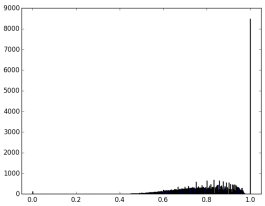
(x) Exclamation sentence frequency



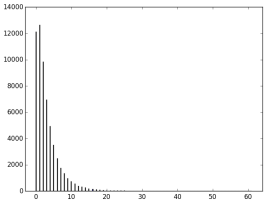
(xi) Capitalized sentence frequency



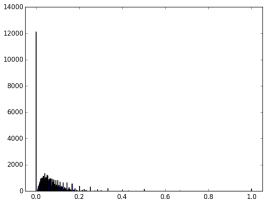
(xii) Comment complexity



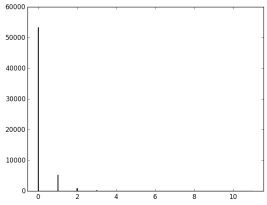
(xiii) Lexical diversity



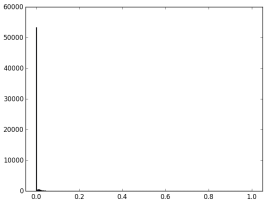
(xiv) Number of incorrectly spelled words



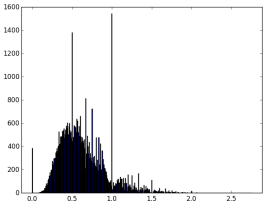
(xv) Percentage incorrectly spelled words



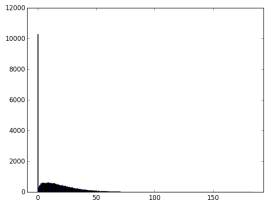
(xvi) Number of profane words



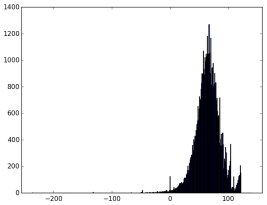
(xvii) Percentage profane words



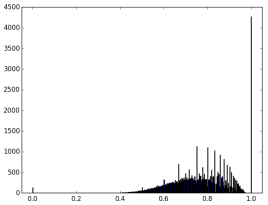
(xviii) Mean term frequency



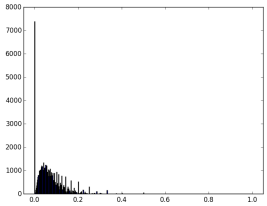
(xix) Informativeness



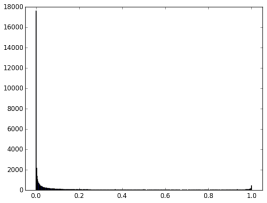
(xx) Readability



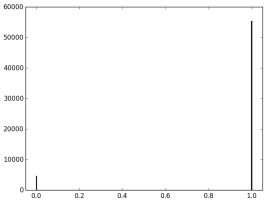
(xxi) Thread relevance



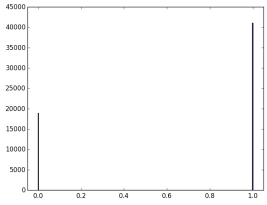
(xxii) Article relevance



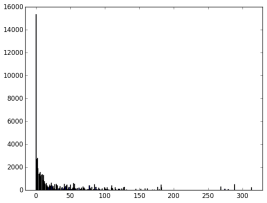
(xxiii) Percentage positive sentiment



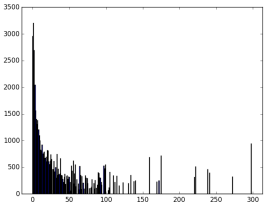
(xxiv) Subjectivity



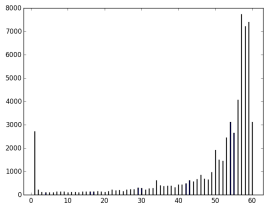
(xxv) Polarity overlap between comment and article



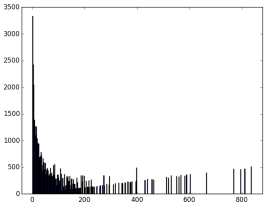
(xxvi) In degree of user



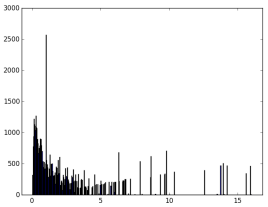
(xxvii) Out degree of user



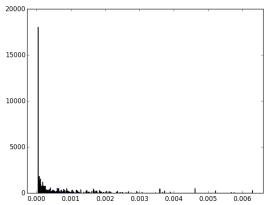
(xxviii) User age



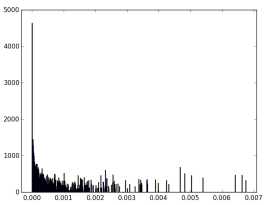
(xxix) Number of posts for user



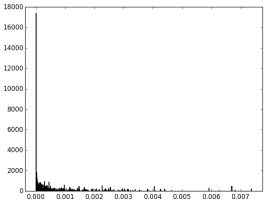
(xxx) User post rate



(xxxi) User pagerank



(xxxii) User hub score



(xxxiii) User authority score

Figure A.3: Distribution of custom features for Slashdot comments.

Appendix B

Artificial Neural Networks

This appendix provides background on artificial neural networks (ANNs) for readers that are not familiar with the techniques presented in Chapter 2.

There are many problems that are difficult for a computer to solve, but typically significantly easier for a human to solve. A human can instantly identify a picture of a dog as being a picture of an animal. In fact, if a human is shown a picture of an animal it has never seen, it can still identify the picture as an image of an animal. Conversely, a computer struggles with identifying simple images (e.g. a simple CAPTCHA [179]).

ANNs are learning models motivated by the techniques the brain uses to store, manipulate and learn from data. Just as the brain can combine concepts into new realizations (e.g. learning sentences from words), ANNs can combine simple components to produce a complex system [8]. The main characteristics of human cognition that ANNs try to emulate, are: adaptive learning capability, generalization, and high fault tolerance.

B.1 Classical Artificial Neural Networks

ANNs are weighted graphs of multiple *neurons* that work in parallel and have the ability to adapt based on feedback from a fixed set of training samples (supervised learning), or from experience (reinforcement learning [168]). Neurons are typically connected by weighted, directed connections. Formally, an ANN can be thought of as a weighted graph containing these neurons as nodes and the connections between them as edges, represented as follows: $G = (N, V)$ where N is the set of neurons and $V \subseteq \{(i, j) \mid i, j \in [1, ..., |N|\,]\}$ the set of connections. Additionally, $W = w_{ij}\ \forall (i, j) \in V$ is a set of weights for each of the connections in V.

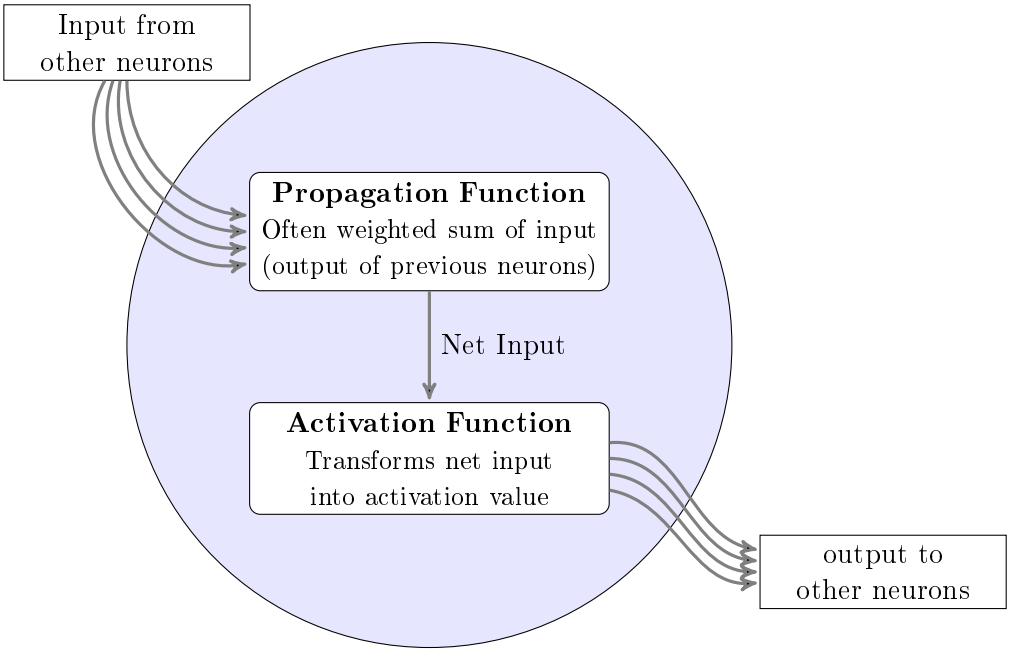


Figure B.1: The way a neuron usually processes data.

B.1.1 The Neuron

A neuron typically has a number of inputs, a computational procedure, and an output. The basic neuron transforms the inputs with this computational procedure, typically consisting of two nested functions: the propagation function and the activation function (see Figure B.1).

The propagation function transforms a set of inputs (often the outputs from neurons in previous layers) into a single net input to the activation function. The dominant propagation function in practice is a weighted sum of the inputs

$$\mathrm{Net_j} = \sum_{i \in I} Out_i \cdot w_{i,j}$$

where the net input, $\mathrm{Net_j}$, is calculated for node j. Here, a weight $w_{i,j}$ is assigned for each incoming edge (or connection) to node j.

The net input is then transformed by an activation function f that determines whether, or how strongly, the neuron "fires". The output of this function can then become input for subsequent neurons connected to it by an outgoing edge. Thus, it transforms the net input Net_j as follows:

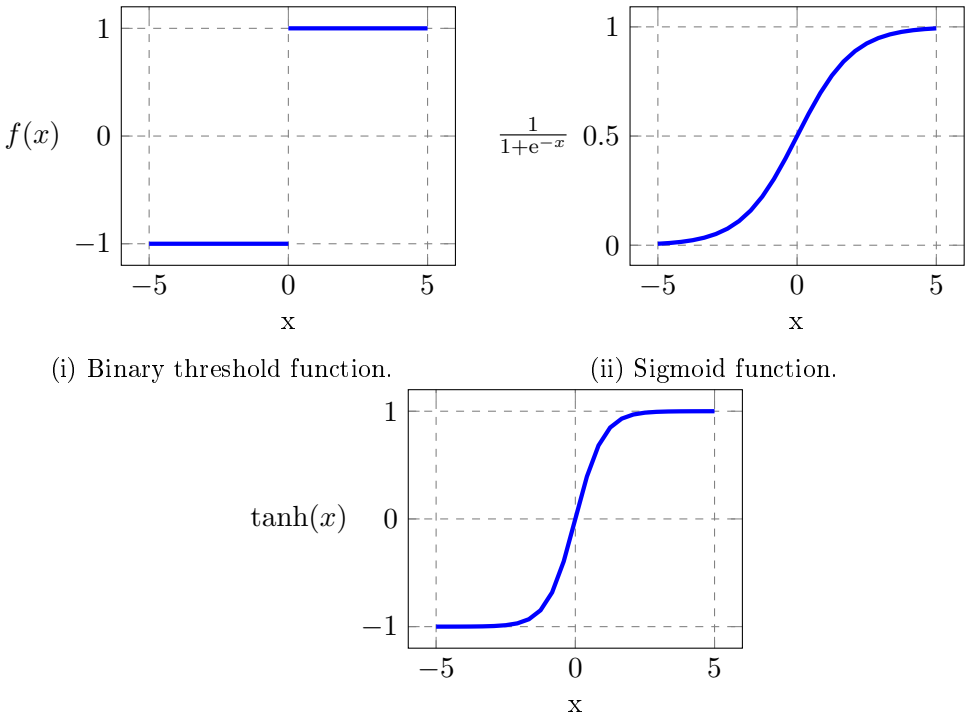
$$\begin{aligned}\mathrm{Out_j} &= f(\mathrm{Net_j}) \\ &= f\left(\sum_{i\in I} w_{i,j}\cdot \mathrm{x_i}\right)\end{aligned}$$

Some common activation functions are (Figure B.2):

- the binary threshold function $f_\theta(x) = \begin{cases} -1 & \text{if } x \leq \theta \\ 1 & \text{if } x > \theta \end{cases}$;
- the sigmoid function $f(x) = \frac{1}{1+\mathrm{e}^{-x}}$; and
- the hyperbolic tangent function $f(x) = \tanh(x) = \frac{\mathrm{e}^{x}-\mathrm{e}^{-x}}{\mathrm{e}^{x}+\mathrm{e}^{-x}}$.

The binary threshold function has the downside of being discontinuous, and thus not differentiable. Differentiability of the activation function is important for the backpropagation step in Section B.1.3. For the discussions below, the sigmoid function will be used, since the derivative has a convenient form, specifically:

$$f'(x) = \frac{\mathrm{e}^{-x}}{(1+\mathrm{e}^{-x})^2} = f(x)(1-f(x))$$



(i) Binary threshold function.

(ii) Sigmoid function.

(iii) Hyperbolic tangent function.

Figure B.2: Common activation functions.

An activation function is often given a threshold value Θ that indicates when the neuron becomes active.[1] The threshold value of a neuron is often hard to change during runtime and often groups of nodes have the same threshold, so a common technique is to replace the threshold value with an additional weighted edge connecting an additional artificial input that continuously fires (i.e. outputs a value of 1) [96]. This is called the bias neuron. It is connected to a node (or a group of nodes with the same threshold value) via a weighted edge with a weight of $-\Theta$. The inclusion of the bias node produces an equivalent network to having the threshold value inside nodes.

Figure B.3 shows a simple neuron with a binary threshold function. It has three inputs and a bias node to offset the binary threshold. A neuron like this is often referred to as a **perceptron** and transforms its input to a single binary output.

A common way to combine values in the output layer of a network, is with the softmax function. The softmax function itself is not an activation function as it is formulated here, but is instead a useful function for producing a distribution over the values of the output layer. It is useful in deep learning applications that will be discussed in Section 3.4. The softmax function

[1] This threshold value is often trained using a learning function.

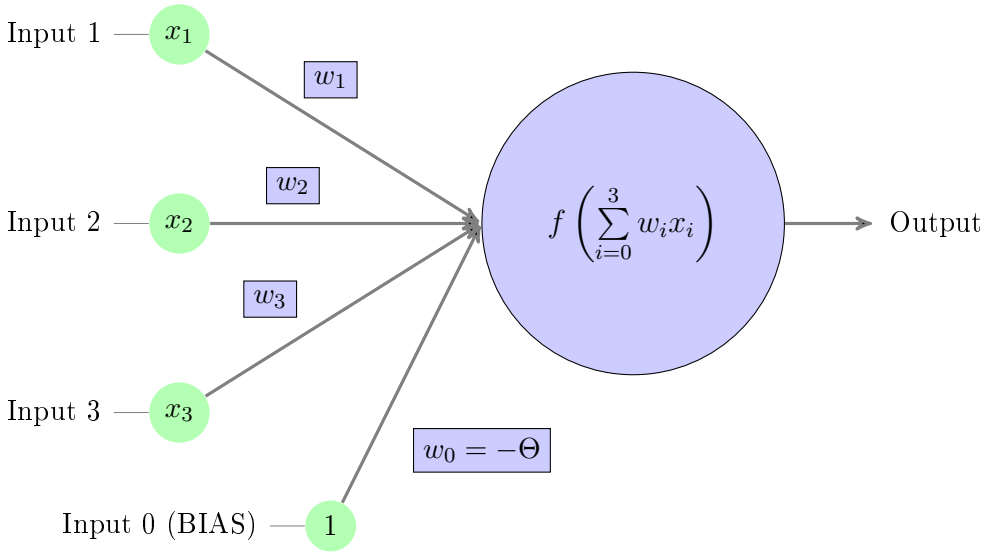


Figure B.3: A perceptron with a binary threshold function that transforms the inputs to a binary output value.

transforms a vector of arbitrary real values into a probability distribution. The function can be used to calculate the probability of a single component of the input vector:

$$f(\mathbf{x})_j = \frac{e^{x_j}}{\sum_{i=1}^{K} e^{x_i}} \forall j \in 1, ..., K$$

where $\mathbf{x}$ is a K-dimensional input vector. In the case of a neural network, the function can be used to calculate the "strength" of a neuron's output as a probability over all the other neurons in its layer in the network.

B.1.2 Feed-Forward Neural Networks

The simplest neural networks follow a feed-forward network pattern, which means that edges between nodes are directed and do not form any directed cycles.

The simplest form of feed-forward network is the single-layer perceptron network (SLP) and consists of a single layer of perceptron units. A single perceptron can learn any binary distribution that can be separated by a plane.

When perceptron units are grouped into multiple layers where each layer's output forms the input to the next layer (i.e. each node in one layer is connected to all the nodes in the next layer), the resulting feed-forward network is called a multi-layer perceptron. With this architecture, there is an input layer (that receives data points as inputs), followed by one or more hidden

layers and an output layer. Multi-layer networks are usually characterised by the number of hidden layers (i.e. a 2-layer network has two hidden layers and additional input and output layers). Figure B.4 illustrates a feed-forward neural network with one hidden layer.

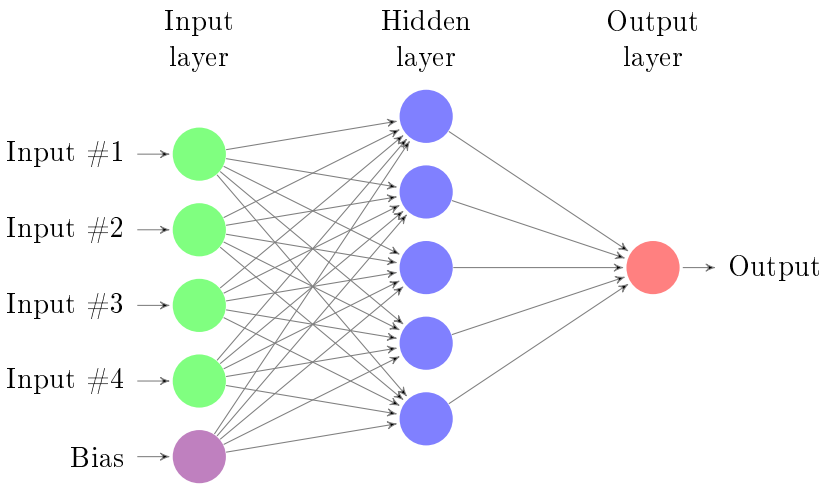


Figure B.4: A basic 1-layer feed-forward neural network.

B.1.3 The Backpropagation Algorithm

The feed-forward approach to neural networks is a popular approach because of the existence of an efficient training algorithm, viz. backpropagation. The backpropagation algorithm [109] was originally proposed to train the weights of the various connections in a feed-forward network, which are often initially random. The backpropagation algorithm is an optimization technique that takes the output of a neural network for an input, compares it to a desired output (usually provided by labelled training data), calculates an error value, and propagates that error backward through the network to update its weights [142]. This process is repeated many times until a convergence criteria is met.

Essentially, the procedure seeks to minimise the error of the network on a given data set using a gradient descent approach [31]. Consider a single neuron n with a sigmoid activation function f, weights $\mathbf{w}_n = (w_0, ..., w_m)$ and inputs $\mathbf{x} = (x_0, ..., x_m)$. Also consider a training set of sample inputs of the form $\mathbf{x}_k$ each with a label $y_k \in \{0, 1\}$. A standard method to compute the error of the neuron n on a sample from the training set is to use the sum of squared errors:

$$\mathrm{E} = \frac{1}{2} \sum_k (y_k - f(\mathbf{x}_k))^2$$

where $f(\mathbf{x}_k)$ is the observed output and y_k is the desired output for node n.[2] Note that this error function E is a function of the set of weights $\mathbf{w}_n$ of node n, since $f(\mathbf{x}_k) = f\left(\sum_{i=0}^{m} w_i x_i\right)$.

Once the error has been obtained, gradient descent suggests updating the weights by:

$$\mathbf{w}_n \leftarrow \mathbf{w}_n - \eta \cdot \nabla \mathrm{E} \tag{B.1}$$

where ∇E is the error gradient and η is a parameter called the learning rate (fixed to be between 0 and 1), which controls the step size of weight updates: if the value is too large, the algorithm may pass optimal solutions by, but too small a value will lead to unreasonable computation time.

In each of the partial derivatives $\frac{\partial E}{\partial w_i}$ of ∇E, the other weights besides w_i (i.e. $w_j \; \forall j \in 0, ..., m, j \neq i$) are constant, so by the chain rule:

$$\begin{aligned} \frac{\partial \mathrm{E}}{\partial w_i} &= -\sum_k (y_k - f(\mathbf{x}_k)) \frac{\partial f}{\partial w_i} \\ &= -\sum_k (y_k - f(\mathbf{x}_k)) f'(\mathbf{x}_k)(\mathbf{x}_k)_i \end{aligned} \tag{B.2}$$

and given that $f(\mathbf{x})$ is the sigmoid function $\frac{1}{1+\mathrm{e}^{-x}}$ and w_i only appears in the i^{th} term of the summation in $f(\mathbf{x})$ (i.e. as coefficient for $(\mathbf{x}_k)_i$, the i^{th} term of $\mathbf{x}_k$).

Further, using the definition for $f'(\mathbf{x})$, the equation becomes:

$$\frac{\partial \mathrm{E}}{\partial w_i} = -\sum_k (y_k - f(\mathbf{x}_k)) f(\mathbf{x}_k)(1 - f(\mathbf{x}_k))(\mathbf{x}_k)_i$$

Thus, $\mathbf{w}_n$ can be updated by:

$$\mathbf{w}_n \leftarrow \mathbf{w}_n + \eta \cdot \sum_k (y_k - f(\mathbf{x}_k)) f(\mathbf{x}_k)(1 - f(\mathbf{x}_k))\mathbf{x}_k \tag{B.3}$$

When considering an entire multi-layer feed-forward network of neurons, the scheme described above becomes more complex. One problem is that the expected value (i.e. y_k for a sample $\mathbf{x}_k$) is only known for the output neuron, but there is no immediate expected output for the internal nodes of the network, and therefore no calculation of error. It is reasonable to assume, however, that the error of an internal node would be dependent on the errors at the output nodes. Consider the example network in Figure B.5 where four input neurons are connected to a single hidden neuron which is in turn connected to four output neurons. Here $w_{i,h}$ indicates the weight of the

[2] The formula is halved for mathematical convenience leading to cleaner partial derivatives in Equation B.2.

edge from input neuron I_i to the hidden neuron and similarly $w_{h,j}$ indicates the weight of the edge from the hidden neuron to output neuron O_j.

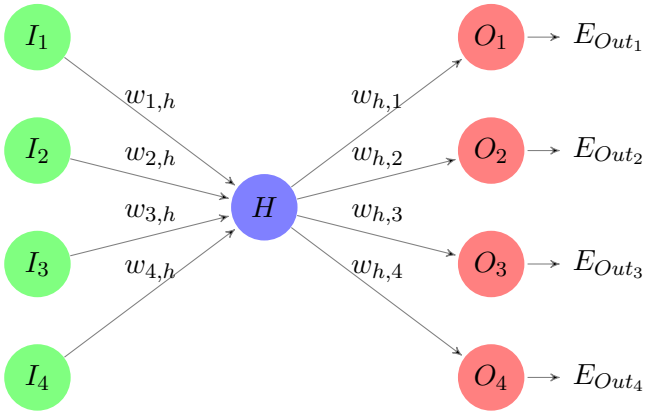


Figure B.5: An example network to illustrate backpropagation of error.

One possible method to calculate the error for neuron H, is by taking the sum over the errors of the output neurons (E_{Out_i}), weighted by the weights of the edges from H to the corresponding output neurons (w_i). This is calculated as follows:

$$\sum_j w_{h,j} E_{\text{Out}_\text{j}}$$

This process can be generalized to the entire network by considering the entire network to be a function with an error function that consists of all the internal weights of the nodes in the network. The partial derivative of the total error E with respect to any of the weights on the edges between the input neurons and the hidden neuron, is given by (by using the chain rule twice):

$$\frac{\partial \text{E}}{\partial w_{i,h}} = \frac{\partial \text{E}}{\partial \text{Out}_\text{H}} \times \frac{\partial \text{Out}_\text{H}}{\partial \text{Net}_\text{H}} \times \frac{\partial \text{Net}_\text{H}}{\partial w_{i,h}} \tag{B.4}$$

where Out_H is the output of the hidden neuron, Net_H is the net input (in this case just the sum of the outputs from the input neurons) and $w_{i,h}$ is the weight on the edge from the i^{th} input neuron to the hidden neuron. In the last term, Net_H only depends on $w_{i,h}$, which means the last term becomes:

$$\frac{\partial \text{Net}_\text{H}}{\partial w_{i,h}} = \frac{\partial \sum_{i*} w_{i*,h} \text{Out}_{\text{i}*}}{\partial w_{i,h}} = \text{Out}_\text{i} \tag{B.5}$$

where Out_i is the output of the i^{th} input neuron. Also, for sigmoid activations, the middle term can be rewritten as:

$$\frac{\partial \text{o}_\text{H}}{\partial \text{Net}_\text{H}} = f_H(\text{Net}_\text{H})(1 - f_H(\text{Net}_\text{H}))$$

The first term is the error derivative with respect to the output of the hidden neuron, but the error is indirectly a function of the output neurons of the network, which means the first term can be written as the sum of the error functions of all the output neurons and the hidden neuron:

$$\frac{\partial \mathrm{E}}{\partial o_H} = \sum_j \left(\frac{\partial \mathrm{E}}{\partial o_j} * \frac{\partial o_j}{\partial \mathrm{Net_j}} * \frac{\partial \mathrm{Net_j}}{\partial o_H} \right) = \sum_j \left(\frac{\partial \mathrm{E}}{\partial o_j} * \frac{\partial o_j}{\partial \mathrm{Net_j}} * w_{h,j} \right) \tag{B.6}$$

The first two terms in the sum are commonly written as δ_{O_j}, the backpropagation error of output neuron O_j. Therefore, Equation B.5 becomes:

$$\frac{\partial \mathrm{E}}{\partial w_{i,h}} = \sum_j \delta_{O_j} w_{h,j} f_H(\mathrm{Net_H})(1 - f_H(\mathrm{Net_H}))o_i$$

so that the weights of a hidden neuron are updated by:

$$\mathbf{w}_H \leftarrow \mathbf{w}_H + \eta \cdot \sum_j \delta_j w_{h,j} f_H(\mathrm{Net_H})(1 - f_H(\mathrm{Net_H}))o_i$$

Essentially, the error for an entire network is determined at the output layer and backpropagated through the network to update weights at individual nodes. The backpropagation algorithm goes through several training iterations (or epochs) before the network is sufficiently optimised. The algorithm usually runs until some stopping criterion is reached. The stopping criterion could be any of a number of conditions, such as the minimum desired total error.

Appendix C

Language Regularities in Word Embedding Models

Using the same techniques as Mikolov et al. [115, 114] for discovering language regularities in a word embedding model, the following list of "close" words and their cosine distance to "south_africa" was discovered in the `word2vec` models described in Section 3.4.2:

- 'sa' – 0.855,
- 'rsa' – 0.844,
- 'country' – 0.818,
- 'south_african' – 0.687, and
- 'south_africas' – 0.672.

Similarly, using the model trained on Slashdot comments, the following list of words related to "south_africa" was discovered:

- 'switzerland' – 0.838,
- 'belgium' – 0.835,
- 'south_america' – 0.815,
- 'spain' – 0.801, and
- 'philippines' – 0.800.

Appendix D

Word Clusters in Word Embedding Models

The following are examples of clusters (only a few words are shown) extracted from the News24 `word2vec` model:

- surprise, suprise, surprises, suprises, surprize;
- people, others, individual, individuals, peoples, persons, fellow_citizens, unjustifiably;
- one, another, any, each, whichever, any_other;
- get, getting, gets, getter; and
- anc, da, ancyl, eff, cope, ruling_party, youth_league, agang, polokwane, lekota.

and similarly, for the Slashdot model:

- average, typical, versus, translates, may_vary;
- people, someone, anyone, everyone, he, guy, nobody, who, somebody, someone_else;
- would, might, may, wouldn, unlikely;
- even, still, assuming, technically; and
- re, youre, were, theyre.

Appendix E

Experiment Reproducibility

The source code for reproducing any of the experiments in Chapter 5 can be found in a code repository on `Github`.[1]

The News24 data sets is not available for the general public, but we released the Slashdot data set and it can be found on `Dropbox`.[2]

To reproduce any of the experiments, follow the `README` file in the code repository.

[1] `https://github.com/DirkBrand/Comment-Classification`.

[2] `https://www.dropbox.com/sh/qt8he7czi5y9y9m/AABa4X5kcW7r-jH3n8vlqoJea?dl=0`.

Bibliography

[1] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 183–194. ACM, 2008.

[2] A. Aizawa. An information-theoretic perspective of TF–IDF measures. *Information Processing & Management*, 39(1):45–65, 2003.

[3] A. Aizerman, E. M. Braverman, and L. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[4] A. U. Alvarez. Bad words list. `http://urbanoalvarez.es/blog/2008/04/04/bad-words-list/`. Accessed March 2014.

[5] C. Alvarez, P. Langlais, and J. yun Nie. Word pairs in language modeling for information retrieval. In *in 7th Conference on Computer Assisted Information Retrieval (RIAO)*, 2004.

[6] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. *arXiv preprint cs/0006013*, 2000.

[7] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[8] Y. Bar-Yam. *Dynamics of complex systems*, volume 213. Addison-Wesley Reading, MA, 1997.

[9] Y. Bengio. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[10] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[11] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[12] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[13] K. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1(1):23–34, 1992.

[14] J. Bergstra and Y. Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[15] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. O'Reilly Media, Inc., 2009.

[16] C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[17] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[18] D. Blankenhorn. The importance of Wikipedia. `http://opensource.com/business/11/11/importance-wikipedia`. Accessed March 2014.

[19] D. M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *the Journal of Machine Learning Research*, 3:993–1022, 2003.

[21] Blinklist. Intensedebate review: Customizing your comment section. `http://blinklist.com/reviews/intensedebate`. Accessed March 2014.

[22] R. J. Bolton and D. J. Hand. Statistical fraud detection: A review. *Statistical Science*, pages 235–249, 2002.

[23] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

[24] L. Bottou. *Large-scale kernel machines*. MIT Press, 2007.

[25] D. Brand and B. van der Merwe. Comment Classification for an Online News Domain. In *Proceedings of the First International Conference on the Use of Mobile Informations and Communication Technology (ICT) in Africa.*, pages 50–56. ACM, 2014.

[26] D. Brand, B. van der Merwe, S. Kroon, and L. Cleophas. N-Gram Representations for Comment Filtering. In *Empowered by Technology — Proceedings of SAICSIT 2015.* ACM, 2015.

[27] M. Brennan, S. Wrazien, and R. Greenstadt. Learning to extract quality discourse in online communities. *AAAI Workshop - Technical Report*, WS-10-02:2–7, 2010.

[28] M. Brennan, S. Wrazien, and R. Greenstadt. Using machine learning to augment collaborative filtering of community discussions. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1569–1570. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[29] L. Canter. The misconception of online comment threads: Content and control on local newspaper websites. *Journalism Practice*, 7(5):604–619, 2013.

[30] R. Caruana and D. Freitag. Greedy Attribute Selection. *Proceedings of the 11th International Conference on Machine Learning*, 48:28–36, 1994.

[31] A. Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[32] W. B. Cavnar and J. M. Trenkle. N-Gram-Based Text Categorization. *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

[33] A. Chapman. Bag of words meets bags of popcorn. `https://www.kaggle.com/c/word2vec-nlp-tutorial`. Accessed 2014-02-22.

[34] N. V. Chawla. Data mining for imbalanced datasets: An overview. In *Data Mining and Knowledge Discovery Handbook*, pages 853–867. Springer, 2005.

[35] B.-C. Chen, J. Guo, B. Tseng, and J. Yang. User reputation in a comment rating environment. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 159–167. ACM, 2011.

[36] M. Chen, X. Jin, and D. Shen. Short text classification improved by learning multi-granularity topics. *IJCAI International Joint Conference on Artificial Intelligence*, pages 1776–1781, 2011.

[37] P. Chen, H. Xie, S. Maslov, and S. Redner. Finding scientific gems with Google's PageRank algorithm. *Journal of Informetrics*, 1(1):8–15, 2007.

[38] W. Chen, Q. Zeng, L. Wenyin, and T. Hao. A user reputation model for a user-interactive question answering system. *Concurrency and Computation: Practice and Experience*, 19(15):2091–2103, 2007.

[39] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec. Antisocial behavior in online discussion communities. *arXiv preprint arXiv:1504.00680*, 2015.

[40] W. Cheng, C. Greaves, and M. Warren. From N-Gram to Skipgram to Concgram. *International Journal of Corpus Linguistics*, 11(4):411–433, 2006.

[41] V. Cherkassky and Y. Ma. Selection of meta-parameters for support vector regression. In *Artificial Neural Networks - ICANN*, pages 687–693. Springer, 2002.

[42] CmdrTaco. Slashdot moderation. `http://slashdot.org/moderation.shtml`. Accessed February 2014.

[43] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural Language Processing (almost) from Scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[44] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[45] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[46] H. Daumé III. *A Course in Machine Learning*. 2012.

[47] P. B. de Laat. Coercion or empowerment? moderation of content in wikipedia as 'essentially contested' bureaucratic rules. *Ethics and information technology*, 14(2):123–135, 2012.

[48] N. Diakopoulos and M. Naaman. Towards quality discourse in online news comments. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pages 133–142. ACM, 2011.

[49] Digg. Digg: Frequently asked questions. Accessed February 2014.

[50] D. Domingo, T. Quandt, A. Heinonen, S. Paulussen, J. B. Singer, and M. Vujnovic. Participatory journalism practices in the media and beyond: An international comparative study of initiatives in online newspapers. *Journalism Practice*, 2(3):326–342, 2008.

[51] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.

[52] J. D'Onfro. Facebook just added more daily active users in the US than Twitter added monthly active users in the world. `http://www.businessinsider.com/facebook-vs-twitter-user-growth-2015-7`. Accessed 2015-10-04.

[53] N. R. Draper, H. Smith, and E. Pownell. *Applied regression analysis*, volume 3. Wiley New York, 1966.

[54] R. Řehůřek. Word2vec in python, part two: Optimizing. `http://rare-technologies.com/word2vec-in-python-part-two-optimizing/`. Accessed September 2015.

[55] Z. Elberrichi and B. Aljohar. N-grams in texts categorization. *Scientific Journal of King Faisal University (Basic and Applied Sciences)*, 8(2):1428H, 2007.

[56] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

[57] T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.

[58] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. *ACL*, pages 363 – 370, 2005.

[59] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[60] N. FitzGerald, G. Carenini, G. Murray, and S. Joty. Exploiting conversational features to detect high-quality blog comments. In *Advances in Artificial Intelligence*, pages 122–127. Springer, 2011.

[61] R. Flesch. A new readability yardstick. *Journal of Applied Psychology*, 32(3):221, 1948.

[62] E. Frank and R. R. Bouckaert. Naive Bayes for text classification with unbalanced classes. In *Knowledge Discovery in Databases: PKDD 2006*, pages 503–510. Springer, 2006.

[63] J. Gao, P. Pantel, M. Gamon, X. He, L. Deng, and Y. Shen. Modeling interestingness with deep neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2014.

[64] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[65] P. E. Greenwood and M. S. Nikulin. *A guide to chi-squared testing*, volume 280. John Wiley & Sons, 1996.

[66] D. G. Gregg and J. E. Scott. The role of reputation systems in reducing on-line auction fraud. *International Journal of Electronic Commerce*, 10(3):95–120, 2006.

[67] S. R. Gunn et al. Support vector machines for classification and regression. *ISIS technical report*, 14, 1998.

[68] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer. Using kNN model-based approach for automatic text categorization. *Soft Computing*, 10:423–430, 2006.

[69] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[70] Z. S. Harris. Distributional structure. *Word*, 1954.

[71] J. Hedley. jsoup: Java html parser. `http://jsoup.org/`. Version: 1.8.3.

[72] R. Herbrich and T. Graepel. A PAC-Bayesian margin bound for linear classifiers. *Information Theory, IEEE Transactions on*, 48(12):3140–3150, 2002.

[73] A. Heß, P. Dopichaj, and C. Maaß. Multi-value classification of very short texts. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5243 LNAI:70–77, 2008.

[74] G. Hinton, J. McClelland, and D. E. Rumelhart. Distributed representations. pages 77–109, 1986.

[75] G. E. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11(10):428–434, 2007.

[76] T. Hirsimäki. On compressing N-gram language models. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume 4, 2007.

[77] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57. ACM, 1999.

[78] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[79] C.-F. Hsu, E. Khabiri, and J. Caverlee. Ranking comments on the social web. In *Computational Science and Engineering, 2009. CSE'09*, volume 4, pages 90–97. IEEE, 2009.

[80] C. W. Hsu, C. C. Chang, and C. J. Lin. A practical guide to support vector classification. *BJU international*, 101(1):1396–400, 2008.

[81] E. H. Huang, R. Socher, C. D. Manning, and A. Ng. Improving word representations via global context and multiple word prototypes. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882, 2012.

[82] C. Hughes. Spamassassin. `http://spamassassin.apache.org/`. Accessed September 2015.

[83] S. Jamali and H. Rangwala. Digging Digg: Comment mining, popularity prediction, and social network analysis. *2009 International Conference on Web Information Systems and Mining, WISM 2009*, pages 32–38, 2009.

[84] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. *International Conference on Machine Learning*, pages 143–151, 1997.

[85] T. Joachims. *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.

[86] I. Jolliffe. *Principal Component Analysis*. Wiley Online Library, 2002.

[87] E. Jones, T. Oliphant, and P. Peterson. SciPy: Open source scientific tools for Python, 2014.

[88] M. Karagiannopoulos, D. Anyfantis, S. Kotsiantis, and P. Pintelas. Feature selection for regression problems. *Proceedings of HERCMA'07*, 2007.

[89] T. Kaszuba, A. Hupa, and A. Wierzbicki. Comment classification for internet auction platforms. In *Advances in Databases and Information Systems*, pages 129–136. Springer, 2010.

[90] J. Kazmierska and J. Malicki. Application of the naïve bayesian classifier to optimize treatment decisions. *Radiotherapy and Oncology*, 86(2):211–216, 2008.

[91] S. Keibler. Importance of the online news portal. `http://www.buddy4study.com/blog/importance-online-news-portal`. Accessed March 2014.

[92] J. Kivinen and M. K. Warmuth. The perceptron algorithm vs. winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pages 289–296. ACM, 1995.

[93] J. M. Kleinberg. Hubs, authorities, and communities. *ACM Computing Surveys (CSUR)*, 31:5, 1999.

[94] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[95] T. Koo, X. Carreras, and M. Collins. Simple semi-supervised dependency parsing. In *Proceedings of ACL*, pages 1–11, 2008.

[96] D. Kriesel. A Brief Introduction to Neural Networks. *Dkriesel.Com*, 2005.

[97] C. Lampe and P. Resnick. Slash (dot) and burn: distributed moderation in a large online conversation space. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 543–550. ACM, 2004.

[98] K. S. Lashley. In search of the engram. *Society of Experimental Biology Symposium*, 1(4), 1950.

[99] Q. Le and T. Mikolov. Distributed Representations of Sentences and Documents. *International Conference on Machine Learning - ICML 2014*, 32:1188–1196, 2014.

[100] D. L. Lee, H. Chuang, and K. Seamons. Document ranking and the vector-space model. *Software, IEEE*, 14(2):67–75, 1997.

[101] H. Liu and H. Motoda. *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media, 2012.

[102] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

[103] L. Luce. Twitter sentiment analysis using python and nltk. `http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/`. Accessed October 2015.

[104] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958.

[105] T. Luong, H. Pham, and C. D. Manning. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, 2015.

[106] L. M. Manevitz and M. Yousef. One-class svms for document classification. *The Journal of Machine Learning Research*, 2:139–154, 2002.

[107] E. Mann. Plugin review - Spam free Wordpress. `http://eamann.com/tech/plugin-review-spam-free-wordpress`. Accessed 2014-03-15.

[108] A. M. Martínez and A. C. Kak. PCA versus LDA. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(2):228–233, 2001.

[109] J. L. McClelland, D. E. Rumelhart, P. R. Group, et al. Parallel distributed processing. *Explorations in the Microstructure of Cognition*, 2, 1986.

[110] W. McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.* " O'Reilly Media, Inc.", 2012.

[111] V. Metsis, I. Androutsopoulos, and G. Paliouras. Spam filtering with naive Bayes - which naive Bayes? *Ceas*, page 9, 2006.

[112] Y. Miao. From word2vec to doc2vec: an approach driven by Chinese restaurant process. `http://eng.kifi.com/`. Accessed 2015-08-05.

[113] T. Mikolov. Using Neural Networks for Modelling and Representing Natural Languages. In *Facebook Research*, 2014.

[114] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. *ArXiv Preprint ArXiv:1310.4546*, pages 1–9, 2013.

[115] T. Mikolov, G. Corrado, K. Chen, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, pages 1–12, 2013.

[116] T. Mikolov, W. T. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

[117] G. A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[118] G. Mishne. Experiments with mood classification in blog posts. *Proceedings of ACM SIGIR 2005 Workshop on Stylistic Analysis of Text for Information Access*, page 19, 2005.

[119] G. Mishne and N. Glance. Leave a Reply: An Analysis of Weblog Comments. *Third Annual Workshop on the Weblogging Ecosystem*, 23:1–7, 2006.

[120] J. Mitchell and M. Lapata. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429, 2010.

[121] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*, pages 1081–1088, 2009.

[122] S. M. Mola-Velasco. Wikipedia vandalism detection. In *Proceedings of the 20th international conference companion on World wide web*, pages 391–396. ACM, 2011.

[123] R. Munroe. Why accuracy alone is a bad measure for classification tasks, and what we can do about it. `http://blog.reddit.com/2009/10/reddits-new-comment-sorting-system.html`. Accessed October 2014.

[124] J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, and A. Y. Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 265–272, 2011.

[125] P. Norvig. How to write a spelling corrector. `http://norvig.com/spell-correct.html`. Accessed March 2014.

[126] M. P. O'Mahony and B. Smyth. A classification-based review recommender. *Knowledge-Based Systems*, 23(4):323–329, 2010.

[127] J. Osborne and E. Waters. Four assumptions of multiple regression that researchers should always test. *Practical Assessment, Research & Evaluation*, 8(2):1–9, 2002.

[128] J. Otterbacher. Helpfulness' in online communities: a measure of message quality. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 955–964. ACM, 2009.

[129] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Stanford Digital Library Technologies*, 1999.

[130] C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 159–168. ACM, 1998.

[131] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, and V. Dubourg. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[132] X. H. Phan, L.-M. Nguyen, and S. Horiguchi. Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections. *Proceeding of the 17th International Conference on World Wide Web - WWW '08*, pages 91–100, 2008.

[133] F. Provost. Machine learning from imbalanced data sets 101. *Proceedings of the AAAI'2000 Workshop*, 2000.

[134] V. V. Raghavan and S. K. M. Wong. A Critical Analysis of Vector Space Model for Information Retrieval. *Journal of the American Society for Information Science*, 37:279–287, 1986.

[135] R. Raina, A. Madhavan, and A. Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 873–880. ACM, 2009.

[136] Ranks.nl. Default english stopwords list. `http://www.ranks.nl/stopwords`. Accessed July 2014.

[137] C. R. Rao. The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society. Series B (Methodological)*, 10(2):159–203, 1948.

[138] R. Rehurek. Practical data science in python. `http://radimrehurek.com/data_science_python/`. Accessed November 2015.

[139] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[140] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[141] P. Resnick and R. Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. *Advances in Applied Microeconomics*, 11:127–157, 2002.

[142] M. Riedmiller. Advanced supervised learning in multi-layer perceptrons—from backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3):265–278, 1994.

[143] M. Rowe, S. Angeletou, and H. Alani. Predicting discussions on the social semantic web. In *The Semanic Web: Research and Applications*, pages 405–420. Springer, 2011.

[144] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive Modeling*, 5, 1988.

[145] S. Russell and P. Norvig. *Artificial Intelligence: a Modern Approach*. Prentice Hall, 1995.

[146] J. Sabater and C. Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1*, pages 475–482. ACM, 2002.

[147] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pages 98–105, 1998.

[148] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. *Proceedings of the 15th International Conference on World Wide Web WWW 06*, pages:377, 2006.

[149] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[150] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1983.

[151] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[152] M. Sasaki and H. Shinnou. Spam detection using text clustering. In *Cyberworlds, 2005. International Conference on*, pages 4–pp. IEEE, 2005.

[153] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10:1299–1319, 1998.

[154] B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[155] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM, 2014.

[156] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, and L. Chanona-Hernández. Syntactic dependency-based N-grams as classification features. In *Advances in Computational Intelligence*, pages 1–11. Springer, 2013.

[157] J. B. Singer. Moderation in moderating comments. `http://www.mediaethicsmagazine.com/index.php/analysis-commentary/3746269-moderation-in-moderating-comments`. Accessed March 2014.

[158] F. Smith. Pattern Classifier Design by Linear Programming. *IEEE Transactions on Computers*, C-17(4):367–372, 1968.

[159] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[160] G. Song, Y. Ye, X. Du, X. Huang, and S. Bie. Short Text Classification: A Survey. *Journal of Multimedia*, 9(5):635–643, 2014.

[161] Y. Song, A. Kołcz, and C. L. Giles. Better naive bayes classification for high-precision spam detection. *Software: Practice and Experience*, 39(11):1003–1024, 2009.

[162] J. Spolsky. Who are the site moderators, and what is their role here? `http://stackoverflow.com/help/site-moderators`, May 2015.

[163] B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas. Short text classification in Twitter to improve information filtering. *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '10*, page 841, 2010.

[164] I. Steinwart and A. Christmann. *Support Vector Machines*. Springer Science & Business Media, 2008.

[165] M. Stelzner. Review: Should you use DISQUS comment system. `http://www.writingwhitepapers.com/blog/2009/07/11/should-you-use-disqus-comment-system-maybe`. Accessed March 2014.

[166] C. Stergiou and D. Siganos. Neural Networks. `http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html`. Accessed May 2015.

[167] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[168] R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[169] G. Szabo and B. A. Huberman. Predicting the popularity of online content. *Communications of the ACM*, 53(8):80–88, 2010.

[170] J. Tang, S. Alelyani, and H. Liu. Feature selection for classification: A review. *Data Classification: Algorithms and Applications*, page 37, 2014.

[171] A. Tomović, P. Janičić, and V. Kešelj. N-Gram-based classification and unsupervised hierarchical clustering of genome sequences. *Computer Methods and Programs in Biomedicine*, 81:137–153, 2006.

[172] A. Trench. Farewell to comments: Why we are making a change. `http://www.news24.com/Columnists/AndrewTrench/Farewell-to-comments-Why-we-are-making-a-change-20150908`. Accessed 2015-10-04.

[173] Tryolabs. Reddit's new comment sorting system. `http://www.tryolabs.com/`. Accessed February 2014.

[174] P. D. Turney, P. Pantel, et al. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, 2010.

[175] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[176] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

[177] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

[178] A. Veloso, W. Meira Jr, T. Macambira, D. Guedes, and H. Almeida. Automatic moderation of comments in a large on-line journalistic environment. In *ICWSM*. Citeseer, 2007.

[179] L. von Ahn. Why humans can solve some problems better than computers. `http://bigthink.com/think-tank/luis-von-ahn-on-recaptcha`. Accessed Feb 2015.

[180] N. Wanas, M. El-Saban, H. Ashour, and W. Ammar. Automatic scoring of online discussion posts. In *Proceedings of the 2nd ACM Workshop on Information Credibility on the Web*, pages 19–26. ACM, 2008.

[181] R. Y. Wang and D. M. Strong. Beyond accuracy: What data quality means to data consumers. *J. of Management Information Systems*, 12(4):5–33, 1996.

[182] R. L. Weide. Carnegie Mellon Pronouncing Dictionary. `www.cs.cmu.edu`.

[183] M. Weimer, I. Gurevych, and M. Mühlhäuser. Automatically assessing the post quality in online discussions on software. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 125–128. Association for Computational Linguistics, 2007.

[184] Wikipedia. Wikipedia of Jimmy Wales and Larry Sanger. `http://history-computer.com/Internet/Conquering/Wikipedia.html`. Accessed March 2014.

[185] D. M. Wilkinson and B. A. Huberman. Cooperation and quality in wikipedia. In *Proceedings of the 2007 International Symposium on Wikis*, pages 157–164. ACM, 2007.

[186] R. Xu, T. Chen, Y. Xia, Q. Lu, B. Liu, and X. Wang. Word Embedding Composition for Data Imbalances in Sentiment and Emotion Classification. *Cognitive Computation*, 2015.