Spring 2016

Popularity Prediction of Reddit Texts

Tracy Rohlin
San Jose State University

POPULARITY PREDICITON OF REDDIT TEXTS

A Thesis

Presented to

The Faculty of the Department of Linguistics and Language Development

San José State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Arts

by

Tracy M. Rohlin

May 2016

The Designated Thesis Committee Approves the Thesis Titled

POPULARITY PREDICTION OF REDDIT TEXTS

by

Tracy M. Rohlin

APPROVED FOR THE DEPARTMENT OF LINGUISTICS AND LANGUAGE DEVELOPMENT

SAN JOSÉ STATE UNIVERSITY

May 2016

Dr. Hahn Koo	Department of Linguistics and Language Development
Dr. Roula Svorou	Department of Linguistics and Language Development
Dr. Chris Pollett	Department of Computer Science

Abstract

Popularity prediction is a useful technique for marketers to anticipate the success of marketing campaigns, to build recommendation systems that suggest new products to consumers, and to develop targeted advertising. Researchers likewise use popularity prediction to measure how popularity changes within a community or within a given timespan. In this paper, I explore ways to predict popularity of posts in reddit.com, which is a blend of news aggregator and community forum. I frame popularity prediction as a text classification problem and attempt to solve it by first identifying topics in the text and then classifying whether the topics identified are more characteristic of popular or unpopular texts. This classifier is then used to label unseen texts as popular or not dependent on the topics found in these new posts. I explore the use of Latent Dirichlet Allocation and term frequency-inverse document frequency for topic identification and naïve Bayes classifiers and support vector machines for classification. The relation between topics and popularity is dynamic -- topics in Reddit communities can wax and wane in popularity. Despite the inherent variability, the methods explored in the paper are effective, showing prediction accuracy between 60% and 75%. The study contributes to the field in various ways. For example, it provides novel data for research and development, not only for text classification but also for the study of relation between topics and popularity in general. The study also helps us better understand different topic identification and classification methods by illustrating their effectiveness on real-life data from a fast-changing and multi-purpose website.

TABLE OF CONTENTS

List of Figures vi

List of Tables vii

Introduction 1

Chapter 1: Reddit 4
1.1 Description of reddit.com 4

Chapter 2: Popularity prediction 10

Chapter 3: Text categorization 13

Chapter 4: Predictive models and feature selection methods 16
4.1 Description of naïve bayes classifiers 16
4.3 Previous research using nbcs and svms 20
4.4 TF-IDF 21
4.5 Latent Dirichlet Allocation 24

Chapter 5: Experiments 30
5.1: Data 30
5.2: Models 32

Chapter 6: Conclusion 48
6.1 Summary and general discussion 48
6.2 Further research 49

References 51

List of Figures

Figure 4.1: A simple Support Vector Machine. 18

Figure 4.2: An example of SVM with polynomial kernel. 19

Figure 4.3: An example of a probability distribution for four topics and three words using LDA. 26

Figure 5.1: Posts' scores distribution from each subreddit. 32

Figure 5.2: The number of high F1 scores per amount of LDA topics modeled. 40

Figure 5.3: The number of high accuracy models per amount of LDA topics. 41

List of Tables

Table 4.1: The top 10 TF-IDF scored words and their weights for each subreddit. 23

Table 4.2 : Each subreddit's top 10 words associated with the top 5 topics generated by the LDA model. 27

Table 5.1: Document and word counts per subreddit. 30

Table 5.2: Results from each subreddit, with the highest accuracy and F1 shaded. 35

Table 5.3: Results from r/learnpython, tuned for high F1 score. 39

Table 5.4: Results from r/xxfitness, tuned for accuracy. 39

Table 5.5: Results from r/learnprogramming with a very low alpha value. 41

Table 5.6: Results from r/learnprogramming with a very high alpha value. 42

Table 5.7: Results from the cross-domain popularity prediction experiment for each subreddit dataset. 45

Introduction

Since the advent of the Internet, researchers have been studying how communities form online and how topics are distributed within each community. The popularity of these topics can wax and wane or remain persistent over time, depending on how focused or generalized that community may be. Along with the formation of these communities, researchers and marketers have been trying to capture topic popularity in order to better predict when a new topic (about a product, person, place, or concept) will be popular given the community or platform used to display the message. In particular, this knowledge provides valuable information to businesses in marketing products and services to consumers. It also provides valuable insight to researchers on how topic distributions and popularity dynamics change over time.

Previous research has treated popularity prediction as a simple binary classification problem, where specific instances (such as tweets, posts, messages, etc.) Are labeled "popular" or "not popular" based on some predefined metric. Researchers have used this schema to study the popularity of social media texts like Twitter messages, Facebook posts, and foursquare posts as well as news-aggregator websites like Digg.com. To do so, researchers have used a variety of models including unsupervised and semi-supervised methods like k-nearest neighbors algorithm and expectation-maximization algorithm, as well as supervised methods like naïve Bayes classifiers (NBC) and support vector machines (SVM). Very little research has been conducted on dual purpose websites such as Reddit.com that serves as a news aggregator and community forum.

Reddit, the self-proclaimed "front page of the Internet," is a website where users can submit links or write posts (called "self-posts") about various topics. There are a multitude of niche sub-communities (called "subreddits") that a user can subscribe to and each post in the community can garner points when other users vote up ("upvote") or vote down ("downvote") a post. While the website has been around for over 10 years, there has been minimal research on predicting popularity of Reddit posts.

In order to fill this gap in the research, I explore methods to predict the popularity of a subreddit's post. I examine the use of term frequency inverse document frequency (TF-IDF) and Latent Dirichlet Allocation (LDA) to extract topics from the posts as well as NBC and SVM to classify whether the posts are popular or unpopular based on their topics. I not only assess if such prediction is feasible in the first place, but also which combination of topic identification and classification methods is most effective. The methods are evaluated on several datasets I have collected, including those from subreddits that are similar in overall topic but vary by subtopic (such as the subreddits r/xxfitness and r/fitness).

The rest of the thesis is organized as follows. Chapter 1 discusses the history and dynamics of Reddit, as well as some of the nascent research that has been conducted using the website's data. Chapter 2 discusses research pertaining to popularity prediction, whereas Chapter 3 explains the concept of text categorization and how it is used in prediction. Chapter 4 reviews the use of NBC and SVM for classification and TF-IDF and LDA for topic identification in previous research. Chapter 5 explains the

methodology and results of my constructed models. Finally, Chapter 6 provides a summary and analysis of the prediction experiment.

Chapter 1: Reddit

1.1 Description of Reddit.com

Founded in 2005, Reddit.com is a popular website that has over 36 million user accounts from over 215 different countries and has around 200 million unique monthly visits [1]. The "front page" of Reddit serves as a news aggregator where popular user-submitted links and posts are gathered from various subreddits [2]. In addition, each subreddit serves as a community forum, where users can submit text posts hosted on Reddit ("self-posts") or external links and pictures, comment on other users' posts, and vote on others' posts as well [3],[4]. Subreddit communities tend to be dominated by a particular format, such that a subreddit will be mostly text, link or picture based [2].

There are over 10,000 active subreddit communities, with varying numbers of subscribers; for example the r/funny subreddit has nearly 10.2 million subscribers whereas r/sagaedition has a mere 551 [5]. While all subreddits must abide by Reddit's site-wide rules regarding content, each subreddit is run independently by a group of volunteers who can design the look of the community's home page, create community-specific rules and posting guidelines, and ban users for violating the rules [1].

In order to post to the community or read posts, users do not necessarily have to subscribe to a particular subreddit, but are not allowed to vote or comment anonymously. Many subreddits have readers that ultimately do not post or comment but may still vote on posts. In fact, Singer et al. Found that online communities often have a large discrepancy between the number of users posting content and the users who only consume but do not post themselves [2]. These "lurkers" tend to outnumber users who

post. Regardless, users who post may suffer downvoting on Reddit if the post is off-topic, does not abide by community rules, or is simply not liked by the majority of the community's readers.

Each reader has one vote to spend on each post and can upvote or downvote a post as well as change his or her previously cast vote. Although expressly prohibited by Reddit's site-wide rules, users may also surreptitiously sign up for multitudes of accounts in order to game the voting system. Users can likewise cheat the voting system by asking friends to upvote or downvote a particular post, or by creating a voting bloc that votes for each other's posts [6], [7]. In addition, each vote must be made by humans, i.e., voting by bot is prohibited [8]. Each post has a continuous tally of upvotes compared to downvotes, which results in the post's final score (i.e., the score is equal to upvotes – downvotes) [4]. For each upvote that a post receives, the user who submitted the post receives an equivalent number of "karma" points, which is effectively a representation of the user's reputation on Reddit. If the number of downvotes outweighs the number of upvotes, the total score for a post is capped at 0. Each post automatically gets one upvote when the user posts it, so it is possible for posts with a score of 0 to experience just one downvote and no upvotes to turn it into a negative post.

Negative posts are often pushed off the subreddit's front page whereas popular and newer posts linger on the front page, following Reddit's "hot" algorithm. In order to calculate a ranking, Reddit's epoch time (December 8, 2005, 7:46:43 a.m.) Is subtracted from the time of the post's creation. Formally, where A is the post creation time and B is Reddit's epoch time:

$$t_1 = A - B$$

And x is the score, or the difference between upvotes and downvotes, and y and z are dependent on the score x:

$$x = U - D$$

$$y = \begin{cases} 1 & if\ x > 0 \\ 0 & if\ x = 0 \\ -1 & if\ x < 0 \end{cases}$$

$$z = \begin{cases} |x| & if\ |x| \geq 1 \\ 1 & if\ |x| < 1 \end{cases}$$

Thus, the final ranking of a post on the subreddit's home page as shown in [10]:

$$f(t_1, y, z) = ylog_{10}z + \frac{t_1}{45000}$$

As a result, submission time has a strong influence on how the posts are ranked on the homepage, and newer stories often get higher scores than older stories. Posts that have minimal difference between upvotes and downvotes (that is, the number of upvotes cancels the number of downvotes) still have a lower ranking than those that have mostly upvotes. As explained in [9], voting is also locked on archived posts, where archiving occurs six months after the post's creation.

Posts that prove to be extremely popular can be displayed on Reddit's front page, which garners even more attention, but most posts do not receive such attention [4], [11]. Submissions that achieve front-page status are more available to users who may have not been readers of the original subreddit that the post came from, as mentioned in [11]. Many users may subscribe to few or no subreddits at all, instead choosing to view only the top links of the day on the front page. This has the effect of amplifying upvotes, such

that popular posts that make it to the front page of Reddit, or remain on the subreddit's front page, often receive even more votes due to their visibility.

Reddit's intricate voting system lends itself well to the problem of popularity prediction, mainly because an easily available metric (the final score) is attached to each post and can be used to delineate popular versus unpopular posts. In addition, its users are highly active and the API (application program interface) and its data are publicly available [4]. For this reason, researchers are beginning to utilize Reddit's massive data stores for various areas of study, as described in the next section.

1.2 Previous Research

Some of the previous research conducted on Reddit has looked at the distribution of subreddits over time, predicting the popularity of the same post over several subreddits, as well as underprovision in subreddit communities. For example, Singer et al. Showed that there was an increasing diversification of subreddits (or topics) between 2008 and 2012 [2]. Simultaneously, they found that there was a concentration of types of submissions, mainly self-posts and images (like those found in the most heavily subscribed subreddits r/funny and r/pics). The research indicates that self-posts are the primary driver of conversations as shown by the large number of comments attributed to self-posts. Indeed, over 50% of the comments made on Reddit are attached to text self-posts as opposed to link or picture based posts. Because of the high level of engagement and the ease of capturing self-posts, I focus my later experiments entirely on self-posts, as described later in Chapter 5.

Other research indicates that the majority of links are clicked on but not voted upon in subreddit communities. In [13], Gilbert describes how up to 52% of link-based posts were overlooked the first time they were submitted. According to the study, this means that "if you submit a great link to Reddit, more than half the time someone else will get the karma associated with the upvotes" [pp. 805]. The researchers also found that posts already considered popular tend to have the most page views and in turn receive even more votes, which then causes the majority of new submissions to be ignored. This may cause issues in predicting popularity because users that like content but ultimately do not vote on the content will not have their votes shown in the data. Ultimately the total score is still a robust measure of popularity, however. The other alternative, calculating number of clicks on a post, would only measure interest in a post, not whether a user actually likes the content.

One study conducted by Lakkaraju, mcauley and Leskovec, looked into how a post's title can influence popularity, regardless of content [14]. The researchers studied resubmissions, meaning posts that contain the same content but differ either in title, time posted, or subreddit posted to. Popularity for a particular piece of content tends to wane after multiple resubmissions, but the researchers found that choice of title can boost the popularity of a post, even though it may have been seen in the same subreddit previously. Again, this study hints at some of the inherent problems when predicting popularity of online texts: if the topic is not new or fresh, users may downvote the post due to saturation effects. Likewise, popularity of a post seems to be highly dependent on the

subreddit it is posted to. These issues are explored further in the next chapter, which discusses popularity prediction.

Chapter 2: Popularity Prediction

Ability to predict popularity of a topic is beneficial in various ways. Companies can formulate more effective marketing strategies, for example by using predicted popularity to build recommender systems and provide targeted advertising as described in [11], [15], and [16]. In addition, web services can achieve higher efficiency and provide a better user experience based on prediction. The Internet service provider can better anticipate the number of requests for contents based on their predicted popularity and adjust infrastructure needs accordingly [12]. Contents predicted to be more popular can be made more readily accessible to users as well [6].

But predicting topic popularity is a hard problem. Topics in general often have a periodic effect, i.e. They peak and decline in popularity [20]. Due to sheer volume of submissions, a vast majority simply fail to reach users and lose opportunities to be rated despite their relevance. For example, in their research on Digg.com (a pre-cursor news aggregator website similar to Reddit), Szabo and Huberman found that only 7.1% of submissions gather enough votes to be promoted to the front page, with only 30% of those receiving over 1000 votes [12]. A similar tendency is also found for social media sites like Twitter, as described in [17].

In addition, topic popularity is affected by various extraneous factors other than the topic itself. These include the nature of the community in which a topic is addressed, the time at which the topic is addressed, and the interaction between the two. In [18], Hu, et al. Discovered that within communities that have moderate interest in a topic, popularity can fluctuate heavily through a given time span. Topics in communities that

have either low or high interest in the topic will maintain a steady popularity level. The study also found that topic popularity will rise earlier in highly interested communities compared to that of moderately interested communities. Another community-related factor that influences popularity is how active the community is. That is, submissions may receive more attention in active communities than in less active communities, as in [18]. This is echoed in [12], where a story's popularity was slower to grow when fewer visitors were on the site (due to time effects) and increased more quickly as more users were on the site.

The activity of a particular community (and thus the popularity of its content) is highly dependent on time effects. These effects tend to coincide with peak website use among users. The study in [6] found popularity effects dependent on the year and in [15], the hour of posting heavily influenced whether a post became popular. The researchers in [15] found that posts may be more popular when posted during busier times of the day although these posts also experience more competition for attention during peak times. In particular, [13] found that the best time to post on Reddit to ensure popularity was in the early morning hours. Posts made in the late afternoon and evening were often ignored. This mirrors the study in [6] which found a higher rate of comments (the researchers' metric of popularity) on the French newspaper website 20Minutes between the hours of 6am and 11am compared to other times. Lastly, saturation effects typically occur after one day on the Yahoo news website [16], on Digg.com [12], and with Twitter messages [21].

Popularity prediction has other issues, however, inherent to the problem of categorization. As an example, according to [3], it can be easy to delineate strictly popular or unpopular posts, but it is often difficult to categorize medium level popularity posts. In addition, a predictive model may not be applied for each possible dataset, as the predictive methods can be influenced heavily by the type and size of the data set, by the site's framework, and other external factors [6]. Nevertheless, Tatar et al. Have found that in many cases a generic (or simple) prediction model was sufficient in predicting popularity. These models are often constructed as binary classification systems, which alleviate the problem described in [3]. Text categorization using classification models are further explored in Chapter 3.

Chapter 3: Text Categorization

Text classification systems categorize texts into different classes based on shared characteristics. A popular example is that of the spam filter, which categorizes incoming emails as spam or not spam, dependent on what words occur in the email and how often those are associated with spam as a whole. It is framed as a classification problem that maps endogenous characteristics of a given email to either one of two categories.

These classes need not be strictly binary, however. As explained in [23], the set of categories are predefined and can be used to categorize by type (e.g., technical reports, emails, or web pages) or by topic (e.g., sports, world news, or financial news). According to Joachim [24], text categorization models have several uses including classification of news stories, guiding searches through hypertext and finding information on the web that a user may be interested in. Hence, text categorization is an inter-disciplinary field that touches on information retrieval, machine learning, statistics, computational linguistics, and data mining [25].

The study in [22] explains the process thoroughly: The document $\langle d_j | c_i \rangle$ is an element of $D \times C$ where D is the domain of documents and C is the set of classes $C = \{c_1, c_2, .. c_n\}$. Each document, d_j is assigned to a class, c_i, such that each document $\langle d_j | c_i \rangle$ has a Boolean value for that particular class. If d_j truly belongs to c_i then $\langle d_j | c_i \rangle$ is labeled true, otherwise the label is false. The process begins by hand-labeling the training set with a particular category and then training a classification model to assign a category to a new, unseen document [23]. The classification model approximates the

unknown target function $\Phi: \mathrm{D} \times \mathrm{C} \rightarrow \{T, F\}$, such that $\widehat{\Phi}$ and Φ (the approximation and the real categorization) are as close as possible [22].

Since text categorization also uses machine learning techniques (similar to popularity prediction), there are some issues that may occur when creating and testing the classification model. Classification accuracy can be dependent on the data set and the model used. Indeed, [24] describes how the models can have low accuracy rates, especially when the amount of training data is small or the quality of training data is low. Poor quality data includes situations where the sample of training documents is not representative of the unseen testing data, leading to high variance. In other words, a situation may occur where the model works well on the training data but not on the testing data due to over-fitting. Sometimes this occurs due to the high dimensionality of feature space (with many of these features (words) being redundant) [23]. According to [26], to reduce the risk of over-fitting, the researcher should gather more data, reduce the feature space and/or increase the regularization parameter (as discussed in Chapter 4 which discusses feature selection methods and classification models).

Regardless of its caveats, a two-class classification system often goes hand-in-hand when popularity prediction methods are being used. In the training set, each document will have the class of "popular" or "unpopular" attached to it. A validation set is often used to tune the parameters of the model, and then the model is run on the test data. Formally, the training set is $Tr = \{d_{1,} d_{2,} \ldots, d_{|Tr|,}\}) \subset D$ and validation set (if being used) is $Va = \{d_{|Tr|+1}, \ldots, d_{|TV|}\}$. Thus, the testing set is $Te = \{d_{|TV+1|,} d_{|TV2|,} \ldots, d_{|TV\Omega|,}\}$ where $|TV\Omega|$ is the size of the testing and validation set

[22, pp. 10]. Several classifier systems exist, including two of the models I have chosen: Naïve Bayes and Support Vector Machines. These models are discussed in Chapter 4 along with the feature selection methods.

Chapter 4: Predictive Models and Feature Selection Methods

4.1 Description of Naïve Bayes Classifiers

Bayesian classifiers look for the most likely class for a given data point. Applied to text classification problems, the data points would be texts or documents to be classified. The probability that a given document d_j belongs to class c_i is proportional to the product of (i) the prior probability of choosing class c_i and (ii) the likelihood of generating document d_j given the class. That is, $P(c_i|d_j,\theta) \propto P(c_i|\theta)P(d_j|c_i,\theta)$ where θ denotes parameters of the classifier. Typically, a document is represented as a vector of features describing which words constitute the document. For example, one could first assume a vocabulary of words expected to be found in documents and then characterize a given document as a vector of zeros and ones with each binary value indicating whether a word in the vocabulary is present in the document or not.

Naive Bayes classifiers (NBCs) assume that features (e.g. Words in the document) are conditionally independent from each other given the class. So the likelihood of a document for a given class is factored into a product of conditional probabilities of words in the document given the class. For example, $P(d_j|c_i) = \prod_{k=1}^{|T|} P(w_{kj}|c_i)$ where $|T|$ is the vocabulary size and w_{kj} denotes presence or absence of some word w_k in document d_j. As [22] explains, the independence assumption may not be theoretically justified but produces robust results in practice.

The parameters of such a NBC consists of the prior probabilities and the word probabilities used for calculating the likelihood. They can be estimated by counting the fraction of times a class or a co-occurrence of a word and a class is observed in data. For

example, $P(c_i)$ would be the fraction of documents in data that are labeled c_i and $P(w_k = 1|c_i)$ would be the fraction of documents labeled c_i that contain the word w_k. But relying exclusively on observed frequencies can pose a problem when words that did not appear in the training data appear in the test data. The *new* word will have a zero probability, i.e. $P(w_{kj} = 1|c_i) = 0$ and result in a zero likelihood, i.e. $P(d_j|c_i) = 0$. This issue is addressed by smoothing methods such as additive smoothing [28], which adds some value α to observed frequencies, i.e.

$$P(w_k = 1|c_i) = \frac{count(w_k,c_i)+\alpha}{D_{c_i}+\alpha|V|}$$

Where $count(w_k, c_i)$ is the number of times w_k appears in documents labeled c_i, D is the total count of all the words in documents labeled c_i, and $|V|$ is the vocabulary size. NBCs built this way are often used to establish a baseline against which other methods such as support vector machines (SVMs) or k-nearest neighbors (k-NN) are compared. It is easy to train them and improve them by incorporating new training data. They can perform well especially when the vocabulary is large [27] and perform at a level comparable to SVMs when coupled with good smoothing methods [29]. But in general, discriminative methods such as SVMs perform better than NBCs in terms of classification accuracy.

4.2 Support Vector Machine

The support vector machine (SVM) is a linear classifier that projects data points represented as vectors into some space and is trained to find the maximum margin hyperplane that separates the vectors into two classes according to which sides of the hyperplane they are on [31]. This is illustrated in Figure 4.1 where the hyperplane is a

line dividing the vectors into circles and squares. The hyperplane is determined from a small set of training examples called support vectors, which determine the margins of the hyperplane and are orthogonal to the hyperplane itself [22, 31]. Applied to text classification, documents in the training set are first represented as vectors $\{x_1,\ x_2, \ldots\ x_n\}$ in some space $X \subseteq\ \mathbb{R}^d$ and tagged with a set of labels $\{y_1,\ y_2, \ldots\ y_n\}$ where $y_i \in \{-1, 1\}$. The model then finds the best hyperplane that separates the data by a maximal margin which label documents on one side of the hyperplane 1 and those on the other side -1.

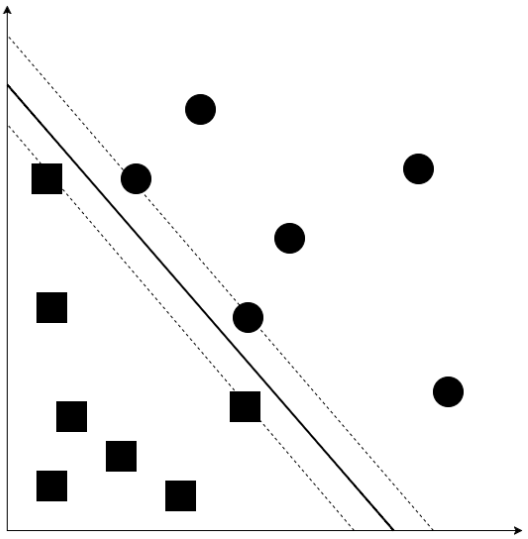


Figure 4.1: A simple Support Vector Machine. The hyperplane separates the two classes. Instances on the hyperplane's margins are called support vectors.

In addition to the original document space X, it is possible to project the original training data into a higher dimensional feature space F by way of a kernel operator [31]. This includes projecting the data into polynomial space or transforming the data using a radial basis function (RBF). Thus data that may not be linearly separable in the original space may still be separable in a higher dimension. This is illustrated in Figure 4.2 where

the vectors in the original two-dimensional space cannot be separated by any straight line but can be separated by a plane after they are projected into a three-dimensional space using the polynomial kernel.

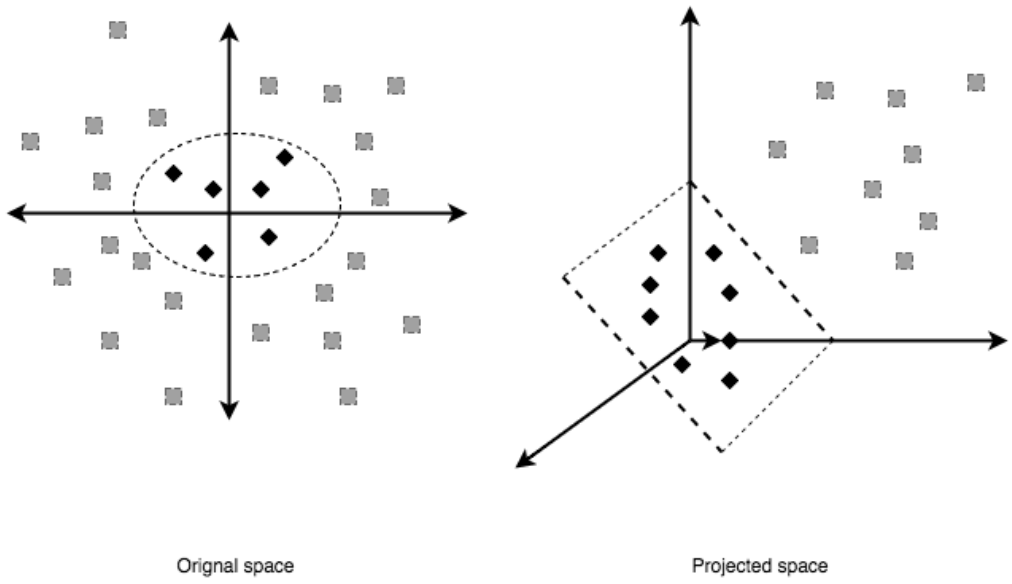


Figure 4.2: An example of SVM with polynomial kernel.

SVMs have many advantages over other classification models. They reduce the amount of work required for feature engineering because one can use kernel functions to efficiently transform the vectors into corresponding vectors in a higher-dimensional space in which they are linearly separable. This makes the model a good fit for text data, whose vector representations may consist of thousands of features. Working in higher dimensional spaces makes the model more robust and resistant to noise such as spelling and grammatical errors, which are prevalent in text data [17, 24]. It also helps the model generalize well to categorizing new instances and reduce the risk of over-fitting since

SVMs tend to favor discriminative features that have broad coverage [3]. For example, the top 1% most discriminative features that the SVM identifies usually work as well as or even better than the original set of features. Unfortunately, SVMs do not work well with a small data set (as shown in [32]) and require that the data be separable in the first place (as shown in [24]).

4.3 Previous research using NBCs and SVMs

Both Naïve Bayes classifiers and SVMs have been used extensively in popularity prediction and text categorization research ([3], [21], [24], [29-35]). They are often compared to each other, as well as to other models. Results show that SVMs usually outperform NBCs as well as other methods such as k-nearest neighbors algorithm (knn) and decision trees. For example, [3] used both SVM and NBC to predict the message popularity by way of Facebook "likes". The study in [21] used and compared both models as well as knn and decision trees to predict how many users will adopt a hashtag on Twitter. The researchers achieved an F1-score of 39.7% using the NBC, but 58.2% using the SVM. In [24] Joachims used the polynomial and RBF kernel SVMs to categorize 50,000 medical abstracts by disease in the Ohsumed corpus, and compared their performance to a decision tree model and a NBC. The SVM with a RBF kernel had the best accuracy with over 86.4% of texts being correctly categorized compared to the polynomial kernel (86%), C4.5 decision tree (79.4%), and NBC (72%).

However, studies suggest NBCs can sometimes outperform SVMs. In [29] both NBC and SVM were used to categorize data from the Yahoo! Webscope website. In the study, NBC was shown to have better results than SVM when the training data set was small.

The researchers in the study concluded that NBC with smoothing techniques applied would be a good model for short text classification. Because SVMs and NBC are robust, computationally efficient and have relatively high accuracy, I have chosen to use them in designing my popularity prediction models. These models are paired with the feature extraction methods TF-IDF and LDA, described in the following chapters.

4.4 TF-IDF

The intuition behind TF-IDF is that topic words are likely to satisfy the following two properties: (i) they appear often in a given document and (ii) they are not words that are easily found in just any document. Formally, the TF-IDF score of a word w_k in a document d_j among a collection of documents D is calculated as follows [35]:

$$tfidf(w_k, d_j) = count(w_{k,} d_j) \times \log(\frac{|D|}{count(D(w_k))})$$

Where $count(w_{k,} d_j)$ is the number of times w_k occurs in d_j, $|D|$ is the total number of documents in the collection, and $count(D(w_k))$ is the number of documents in the collection that contain w_k. Often the scores are transformed to fall in the range [0, 1] via cosine normalization. That is, the normalized TF-IDF score of w_k in d_j denoted $\widehat{w}_{kj}$ is calculated as follows:

$$\widehat{w}_{kj} = \frac{tfidf(w_{k,} d_j)}{\sqrt{\sum_{s=1}^{|T|}(tfidf(w_{s,} d_j))^2}}$$

Where $|T|$ is the total number of words in the document.

In sum, a word is more likely to be a topic word if it appears often in a given document (term frequency) and there are only a few documents in which the word

appears (inverse document frequency). Incorporating the inverse document frequency allows one to build a model with a smaller feature space and a higher classification accuracy than when using the term frequency alone. But the approach does have some caveats. According to [35], it can be a poor choice for certain domain-specific datasets because the inverse document frequency prefers rare features. If all the documents in the collection were in the same topic domain, some topic words would be ubiquitous and have the worst inverse document frequency. The study in [28] also states that TF-IDF only offers a small amount of description length reduction. More importantly, the feature set dimensionality for TF-IDF is the entire vocabulary of the corpus, which results in a huge computation when determining the weight of each term in a document [23]. This ultimately results in a computation of O(nm) where n is the number of tokens and m is the number of documents.

Despite its weaknesses, TF-IDF remains a popular method in text categorization, information retrieval and popularity prediction research when determining the topic distribution of a set of documents. For example, [33] used TF-IDF when trying to extract keywords from Japanese abstracts. Studies show that TF-IDF can be effective when coupled with SVMs. For example, the researchers in [23] used TF-IDF with the SVM model to categorize texts from Chinese academic journals and texts from the Reuters corpus. They found that TF-IDF performed better than other methods such as Latent Semantic Indexing (LSI) when categorizing English texts as Reuters although the opposite was true when categorizing Chinese texts. Studies also show that TF-IDF can

be effective with NBCs too. For example, the researchers in [40] used TF-IDF with NBCs to determine ambivalently punctuated Chinese texts and achieved 91% accuracy.

As an illustration of effectiveness of the approach, Table 4.1 shows the top ten words with the highest TF-IDF scores identified in posts from six subreddits. Each subreddit is named after the broad topic discussed in the community (e.g. "learnprogramming" in r/learnprogramming). Note the relevance of words in the list to the community topic (e.g. "float" to "learnprogramming", "chinup" to "xxfitness").

Table 4.1 (continued below): The top 10 TF-IDF scored words and their weights for a whole subreddit corpus.

r/learnprogramming	
float	0.924535
num	0.899910
books	0.867517
double	0.861241
arraylength	0.8417568
calculator	0.8365434
blogs	0. 834947
reach	0.822829

r/learnpython	
password	0.892562
blah	0.891450
filename	0.869728
limit	0.867090
similar	0.844441
virtualenv	0.823746
price	0.823638
slides	0.822980

r/fitness	
shoes	0.948515
king	0.873034
cycling	0.869757
plates	0.868517
oatmeal	0.843767
belt	0.810969
traps	0.784525
g (grams)	0.839534

r/askmen	
dated	0.860421
challenge	0.846030
wondered	0.838315
cheater	0.837391
dislike	0.822690
express	0.816906
surgery	0.801718
choose	0.799288

r/askwomen	
blah	0.922855
law	0.911031
song	0.884825
beer	0.878346
skirt	0.838661
gay	0.823846
period	0.820234
balls	0.816302

r/xxfitness	
chinup	0.830705
pictures	0.816889
grip	0.770956
buddy	0.758762
pair	0.751004
thongs	0.745934
forearms	0.744894
shoes	0.744225

4.5 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a method based on a generative three-level Bayesian probability model. In LDA, each document is modeled as a finite mixture of probabilities over an underlying set of topics based on which words are commonly associated with said topics [28]. In other words, the model finds the probability distributions of different topics given a set of documents and discovers the word distributions for each topic, as explained in [41]. Each topic is itself modeled as an infinite mixture over an underlying set of topic probabilities [28]. Other topic models, especially unsupervised clustering models, often restrict a document as being associated with a single topic. LDA, on the other hand, allows a document to be represented by a mixture of topics, each represented as a point on the word probability distribution. In a sense, each document shares the same set of topics but exhibit those topics to differing

degrees. The document can then be categorized based on its finite mixture using a classifier that has learned the distribution of topics for a particular class.

The model assumes the following generative process for each document: First, decide the number of words (N) the document will contain according to a Poisson distribution (ξ). Second, choose a topic mixture (θ) – how the topics will be distributed in the document – according to a Dirichlet distribution (α). Third, choose a topic (z_n) for each word position according to the topic mixture. Finally, choose a word (w_n) for each position according to the topic of the position (z_n) and its multinomial distribution (β).

The topic structure of a given document can be identified by computing the following posterior probability:

$$P(\theta, z|w, \alpha, \beta) = \frac{P(\theta, z, w|\alpha, \beta)}{P(w|\alpha, \beta)}$$

Where w denotes the set of n words in the document and z denotes the corresponding n topics [28, 38]. The posterior probability is computed using approximate inference algorithms such as Laplace approximation, variational approximation, and Markov chain Monte Carlo. Figure 4.3 shows an example of a probability distribution using three words and four topics. Each corner of the triangle represents a probability of 1 for a particular word, whereas the midpoint on an edge between two peaks represents a probability of 0.5. The middle peak represents the uniform distribution between all three words. A point on each peak represents a particular probability for the word given the topic, i.e., $P(w|z)$.

Unfortunately, LDA can be computationally expensive because each document is required to model the topic distribution and the word distributions must also be calculated for each topic. It can require O((N+1)k) time, where k is the number of topics in β. In addition, [18] states that LDA may not be an appropriate choice for short texts such as forum posts and microblogs (like Twitter) because those tend to be focused on a single topic.

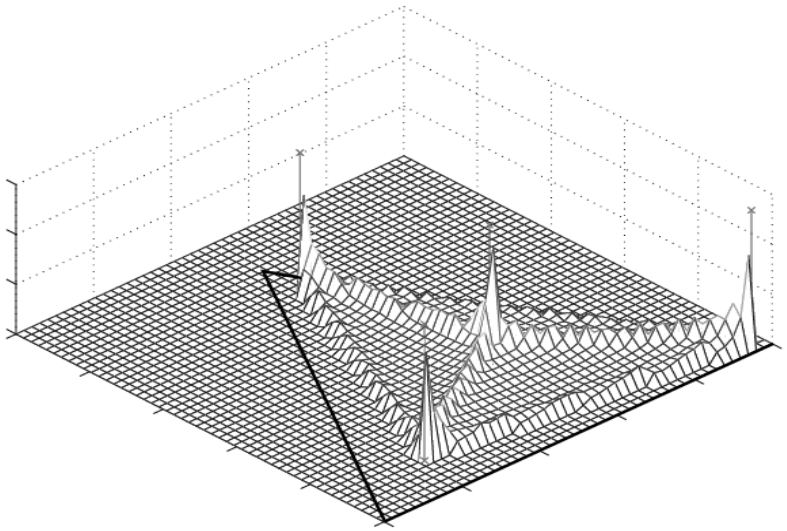


Figure 4.3: An example of a probability distribution for four topics and three words using LDA. [28, pp. 999]

Despite such issues, LDA, similar to TF-IDF, is a popular method in information retrieval, text summarization and text categorization. For example, the researchers in [4] used LDA to determine the point in which a discussion among a comment thread on Reddit began to diversify, i.e. When the discussion began to deviate from the original comment or post. To do this, the researchers used a hierarchal version of LDA to cluster words into a hierarchy of topics such that general words occurred at the top of the

hierarchy and more domain specific words were towards the leaves of the hierarchy. As a result, the study found that Reddit comments typically have one or two sub-threads that receive the most votes and comments, and that this attention in sub-threads occurs quickly. Another study used LDA when trying to predict popularity of posts coming from restaurants and businesses on Facebook and youtube [3]. The researchers used LDA to identify 10 main topics relating to the businesses' posts to determine the relationship between content, media type, and popularity. The study in [16] similarly used LDA to model topics using youtube video tags to determine popularity. LDA was also used in [15] to predict user choices for pay-per-view movies purchased by users of two European internet protocol television (IPTV) providers. Lastly, LDA was used for popularity prediction in [27] to determine topic distributions for hashtags on Twitter.

As an illustration of the effectiveness of the approach, Table 4.2 shows top ten words associated with top five topics identified via LDA for the six subreddits.

Table 4.2 (continued below): The top 10 words associated with the top 5 topics generated by the LDA model per subreddit.

R/learnpython										
Rank:	Word #1	Word #2	Word #3	Word #4	Word #5	Word #6	Word #7	Word #8	Word #9	Word #10
Topic #1	import	code	files	folder	file	data	list	get	python	run
Topic #2	print	elif	string	search	else	number	input	numbers	rawinput	random
Topic #3	python	like	code	use	want	know	game	really	learn	make
Topic #4	deck	cards	rank	rank	populate	supports	boring	selfs	shuffled	removes
Topic #5	server	password	system	sql	username	send	event	script	information	true

r/learnprogramming										
Rank:	Word #1	Word #2	Word #3	Word #4	Word #5	Word #6	Word #7	Word #8	Word #9	Word #10
Topic #1	program-ming	like	know	learn	learning	want	get	java	really	language
Topic #2	code	file	program	like	help	using	files	get	run	new
Topic #3	class	functions	use	visual	using	studio	object	function	code	state
Topic #4	game	part	amount	month	ai	implementa-tion	previous	depth	games	rate
Topic #5	total	device	final	high	arduino	interact	nil	possibilities	serial	costs

r/askmen										
Rank:	Word #1	Word #2	Word #3	Word #4	Word #5	Word #6	Word #7	Word #8	Word #9	Word #10
Topic #1	like	men	think	sex	things	husband	edit	guys	want	feel
Topic #2	drunk	bro	inspired	traditional	sober	thread	bet	grinding	perceive	sexes
Topic #3	guilt	club	something	health	grown	drug	party	times	culture	afraid
Topic #4	work	job	new	want	time	much	advice	summer	make	live
Topic #5	I'm	don't	kiss	I've	date	dog	month	spending	rejected	texted

r/askwomen										
Rank:	Word #1	Word #2	Word #3	Word #4	Word #5	Word #6	Word #7	Word #8	Word #9	Word #10
Topic #1	like	get	look	day	work	know	wear	nice	keep	men
Topic #2	marriage	emotionally	religious	stable	safe	hangout	means	teacher	inexperienced	boot
Topic #3	last	years	name	year	back	week	children	going	work	things
Topic #4	hair	control	birth	ladies	use	feel	average	products	PIV	wondering
Topic #5	emotion -al	new	crazy	heels	emotion -ally	hitting	tip	spring	cry	resolution

r/fitness										
Rank:	Word #1	Word #2	Word #3	Word #4	Word #5	Word #6	Word #7	Word #8	Word #9	Word #10
Topic #1	get	questions	week	post	question	thread	sure	answer	time	fittit
Topic #2	press	bench	squats	weight	set	reps	leg	sets	week	lbs
Topic #3	youtube	milk	brands	vitamins	popular	raw	ounces	lactose	cooked	drinking
Topic #4	back	lbs	kg	form	squat	gym	weight	bar	bench	lb
Topic #5	new	training	fitness	knee	people	core	body	know	someone	run

r/xxfitness										
Rank:	Word #1	Word #2	Word #3	Word #4	Word #5	Word #6	Word #7	Word #8	Word #9	Word #10
Topic #1	thread	tuesday	newbie	questions	review	access	awesome	might	price	new
Topic #2	MG	protein	coffee	eat	pretty	yogurt	cup	oz	cups	breakfast
Topic #3	stack	day	days	leg	press	legs	side	abs	workout	raises
Topic #4	shoes	sports	women	trainer	bras	good	personal	fitness	anyone	want
Topic #5	weight	lbs	fat	started	body	lost	muscle	lifting	pounds	progress

Chapter 5: Experiments

5.1: Data

The data for the study are a corpus of 12,847 Reddit posts that I collected from six subreddits: two on relatively general topics (r/askwomen and r/askmen) and four on more niche topics (r/xxfitness, r/fitness, r/learnprogramming and r/learnpython). Each subreddit added about 2,000-2,500 documents to the corpus (see Table 5.1 for details). Roughly half of the documents from each subreddit were labeled *popular* and the other half *unpopular*. Each subreddit dataset was split into three subsets -- training (60%), validation (20%), and test (20%) -- with each split containing equal amounts of popular and unpopular posts. Below I describe in detail the data collection process, inclusion-exclusion criteria, and labeling criteria.

Table 5.1 (continued below): Document and word counts per subreddit.

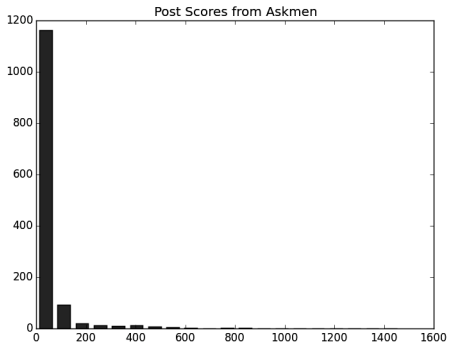
	Number of documents In corpus	Number of words In corpus
r/xxfitness	2,335	5,276
r/fitness	1,926	4,570
r/askmen	2,011	4,013
r/askwomen	1,893	3,483
r/learnpython	2,251	4,146
r/learnprogramming	2,431	4,407

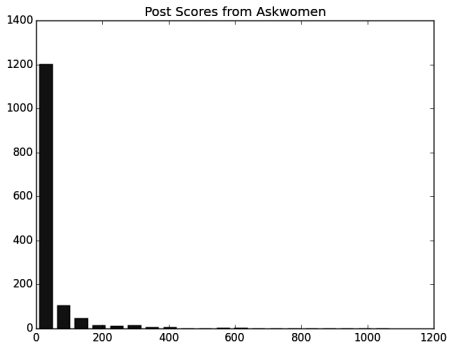
The six subreddits were chosen because they contained many posts that have been available to the users long enough and for which voting had closed (typically six months after creation). This was to reduce the recency effect and to alleviate the complication experienced by [12] in which the researchers found that content with a longer life cycle in which users were still voting on posts tended to have a large statistical error during prediction.

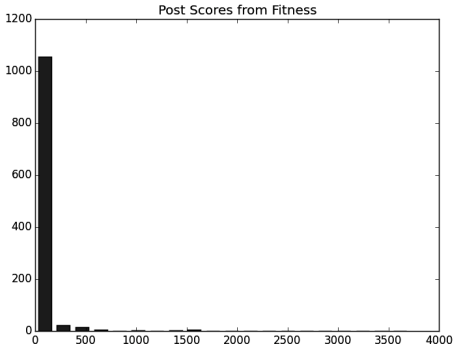
Posts within each subreddit were chosen only if (i) they had at least two voting points associated with it and (ii) contained at least 20 words. Reddit automatically assigns a score of 1 to every new post, so the first condition was imposed to ensure that each post had in fact been voted on. The second condition was imposed to ensure that each post contained enough information for topic identification and ultimately popularity prediction.

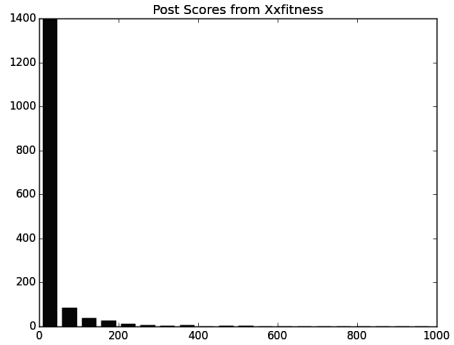
Each post in a subreddit was labeled *popular* or *unpopular* by comparing its voting score to a threshold, which I defined as the 75^{th} percentile score among the first 1,000 posts collected from the subreddit. More posts were gathered from the subreddit with preference given to those with higher scores than the threshold so that the dataset for each subreddit had roughly equal amounts of popular and unpopular posts. Note, however, that the distribution of scores for each subreddit is rather skewed to the right, with very few posts achieving a score above 50 as illustrated in Figure 5.1.

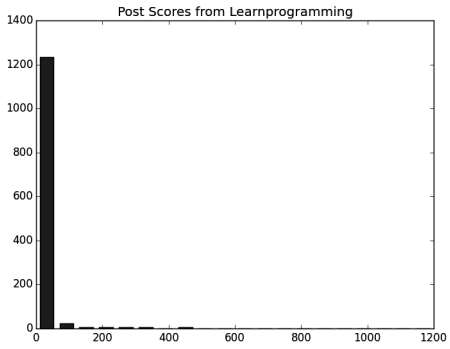
Figure 5.1 (continued below): The distribution of posts' scores from each subreddit.

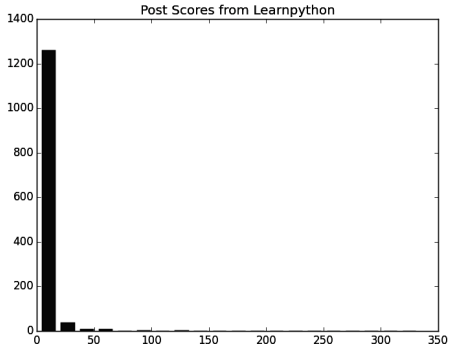












5.2: Models

The classifiers were trained to tag a feature vector representation of a given post as either popular or unpopular. Each post was first preprocessed and represented as a feature vector using the bag of words (BOW) model, term frequency – inverse document

frequency (TF-IDF), or Linear Dirichlet Allocation (LDA). The resulting feature vector was fed into either a Naïve Bayes classifier (NBC) or a support vector machine (SVM). I describe the details of preprocessing, feature representation, and classifiers below.

Each post was preprocessed by removing stop words – articles, prepositions, conjunctions and contractions as well as non-alphanumeric characters – and low frequency words that appeared in no more than two documents in the corpus. The idea to remove low frequency words is in line with previous studies such as [3] and [24]. The aim is to reduce the dimensionality of feature space as well as classification errors by removing spelling errors, which often appear in the corpus as low frequency words [29]. No stemming or lemmatization was performed on the documents as [22] indicated that doing so could hurt classifier performance.

The BOW model essentially represents the preprocessed document as a mere set of words in it. Words in the bag are not associated with any scores that denote the extent to which a given word captures the document topic. On the other hand, TF-IDF and LDA further assign such scores to the words in the bag. When using a NBC, the LDA or TF-IDF scores replace the word counts (where the instance of a each word is equal to 1). The score for a given word is then divided by the summed scores for the entire corpus to produce a probability similar to the base BOW model. Each word probability is then used with the class's prior probability to classify the given document.

Parameters of the NBC were smoothed by the additive method explained previously with $\alpha = 1$. I chose the linear kernel function for the SVM rather than polynomial or RBF kernels due to its popularity attested in [39] and its performance in a

preliminary analysis. The linear SVM was trained using stochastic gradient descent, which essentially finds the hyperplane $h(\theta)$ that minimizes a cost function through an iterative process [42]. So given a hypothesis hyperplane $h(\theta) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$ and a cost function of $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i - y_i)^2$, the best hyperplane is found by replacing the weights θ with $\theta_j = \theta_j - \alpha \frac{\delta}{\delta \theta_j} J(\theta_o, \ldots, \theta_n)$. In this formula, α represents the "size" of the step, or the degree to which the weights are adjusted. The SVM and NBC results are discussed in the next section.

5.3: Results and Discussion

I evaluate and report performance in terms of classification accuracy and F1 score on the test data. Classification accuracy is essentially the fraction of documents classified correctly, i.e. $accuracy = \frac{TP+TN}{TP+TN+FN+FP}$ where TP, TN, FP, FN denote true positives, true negatives, false positives, and false negatives, respectively. F1 score is an average of two scores called precision and recall [35]. Recall is the fraction of documents correctly classified as popular out of all documents that should have been classified as popular, i.e. $Recall = \frac{TP}{TP+FN}$. Precision, on the other hand, is the number of documents correctly classified as popular out of all documents that the classifier categorized as popular, i.e. $Precision = \frac{TP}{TP+FP}$. F1 score is the harmonic mean of the two, i.e. $F1 = \frac{2*precision*recall}{precision+recall}$. The results are shown in Table 5.2.

Table 5.2 (continued below): Results from each subreddit, with the highest accuracy and F1 shaded. 't' indicates the number of topics used to obtain the LDA models.

r/learnprogramming	Naïve Bayes		SVM	
	Accuracy	F1	Accuracy	F1
BOW	0.6263	0.6967	0.5996	0.7041
TF-IDF	0.6263	0.7393	0.6181	0.7343
LDA	0.6324 t = 40	0.6992 t = 40	0.6283 t = 50	0.7454 t = 50

r/learnpython	Naïve Bayes		SVM	
	Accuracy	F1	Accuracy	F1
BOW	0.5610	0.6087	0.5144	0.5697
TF-IDF	0.5920	0.7023	0.5876	0.7023
LDA	0.6142 t = 40	0.6752 t = 70	0.5965 t = 80	0.7306 t = 10

r/askmen	Naïve Bayes		SVM	
	Accuracy	F1	Accuracy	F1
BOW	0.6030	0.6190	0.6104	0.6323
TF-IDF	0.6104	0.6180	0.6154	0.6437
LDA	0.6005 t = 40	0.6397 t = 100	0.6055 t = 70	0.6553 t = 40

r/askwomen	Naïve Bayes		SVM	
	Accuracy	F1	Accuracy	F1
BOW	0.5323	0.5672	0.5290	0.5380
TF-IDF	0.5226	0.5723	0.5290	0.5576
LDA	0.5355 t = 80	0.5505 t = 50	0.6032 t = 70	0.6154 t = 60

r/xxfitness	Naïve Bayes		SVM	
	Accuracy	F1	Accuracy	F1
BOW	0.7259	0.7181	0.6916	0.6842
TF-IDF	0.7323	0.7073	0.7216	0.7032
LDA	0.6660 t = 90	0.6516 t = 50	0.6210 t = 80	0.6380 t = 80

r/fitness	Naïve Bayes		SVM	
	Accuracy	F1	Accuracy	F1
BOW	0.6477	0.6683	0.6554	0.6970
TF-IDF	0.6606	0.7514	0.6503	0.6953
LDA	0.6269 t = 30	0.6538 t = 30	0.6244 t = 80	0.7259 t = 80

Table 5.2 shows classification performance on six subreddit data sets. In sum, classification accuracy is above chance level (~50%) for all six data sets. With only one exception (F1 for r/xxfitness), performance is better when the classifiers use topic features derived via TF-IDF or LDA than when they simply rely on BOW.

Previous research indicates that NBCs often performs worse than SVMs but this appeared to be only partly true for predicting popularity of Reddit texts. Four of the six datasets achieved higher F1 when using an SVM classifier. Surprisingly, the NBC model still had better accuracy, out-performing the SVM in four of the six data sets. For example, two data sets, r/learnprogramming and r/learnpython, achieved a higher F1 score from the SVM but ultimately had the highest accuracy when using the NBC model paired with LDA. Two other datasets, r/fitness and r/xxfitness, had higher accuracy and F1 scores using NBC models. This leaves both r/askwomen and r/askmen as the only two datasets that achieved better performance with an SVM, both in terms of F1 and accuracy.

Even though the most accurate model was often NBC, the second best model in terms of accuracy was often the SVM. Only r/askwomen and r/xxfitness achieved second

best accuracy with an NBC. The second best F1 scores were split evenly between SVM and NBC, however. It is clear that an overlap in theme can cause two separate datasets to have similar performance when using the same classifier, as is the case for r/askwomen and r/xxfitness, two subreddits related to women's issues. This was seen as well in r/fitness and r/xxfitness, relating to health and exercise, when they both performed better using the NBC.

In terms of feature selection methods, LDA often achieved equal accuracy to TF-IDF but had higher F1 scores than TF-IDF or a BOW model. When using LDA the SVM had the highest F1 score compared to BOW and TF-IDF for four of the six datasets. On the other hand, highest accuracy for each dataset was split evenly between TF-IDF and LDA. Only one dataset achieved its highest F1 score using TF-IDF, another using the BOW model and no dataset achieved highest accuracy using a BOW. Still, for each dataset TF-IDF often obtained better performance than BOW and LDA better performance than both.

The NBC model had somewhat similar performance, with both LDA and TF-IDF performing better than BOW for most of the datasets. Only one dataset, r/xxfitness, had a higher F1 score as a BOW, but most other datasets had the highest F1 scores using TF-IDF. Highest accuracy for all six datasets using NBC was split evenly between LDA and TF-IDF. Ultimately, however, a high accuracy score for a feature selection method for one classifier may have been beaten out by another classifier using a different feature selection method, leading to the even split for accuracy between LDA and TF-IDF.

Datasets that achieved high accuracy and F1 scores using different models and feature selection methods are likely due to high instances of false positives or false negatives skewing the F1 score. As the recall or precision score approaches 1 because of low amounts of falsely labeled instances, the F1 score begins to lower, but accuracy will begin to rise. Take, for example, the results from r/learnpython using an LDA model of 10 topics and the SVM classifier, as shown in Table 5.3:

Table 5.3: Results from r/learnpython, tuned for high F1 score.

R/learnpython	Predicted positive	Predicted negative
True positive	256	3
True negative	189	2

In this case the recall is 0.9884 and precision is 0.5753, primarily because there is a large number of true positives and low number of false negatives, making recall especially high. This means that the F1 score will be $\frac{1.1373}{1.5637}$ resulting in an F1 score of 0.7273. But because the classifier has labeled most of the instances as positives, which results in a large number of false positives, the accuracy remains much lower: 0.5733. Compare this to the NBC LDA results modeling 90 topics, as in Table 5.4:

Table 5.4: Results from r/xxfitness, tuned for accuracy.

R/xxfitness	Predicted positive	Predicted negative
True positive	144	86
True negative	70	167

As seen in the confusion matrix, this model has more correctly labeled true positives and true negatives. The recall at 0.6261 and precision at 0.6729 create an F1 score of 0.6486 that is less than the accuracy of 0.6660 (and smaller than the F1 ratio from r/learnpython). Unfortunately, tuning for accuracy often creates a situation where one model may have higher accuracy, but another a higher F1 score. Such is the case for 10 of the 36 models paired with feature selection methods tested on the datasets.

Other factors seen to be affecting the accuracy and F1 scores included the amount of topics modeled during LDA and the alpha regularization parameter used in the SVM model. How the number of LDA topics modeled affects the classifier models is slightly unpredictable, however. Most of the highest performing models (for both accuracy, as in Figure 5.2, and F1, as in Figure 5.4) had between 40 and 80 topics modeled.

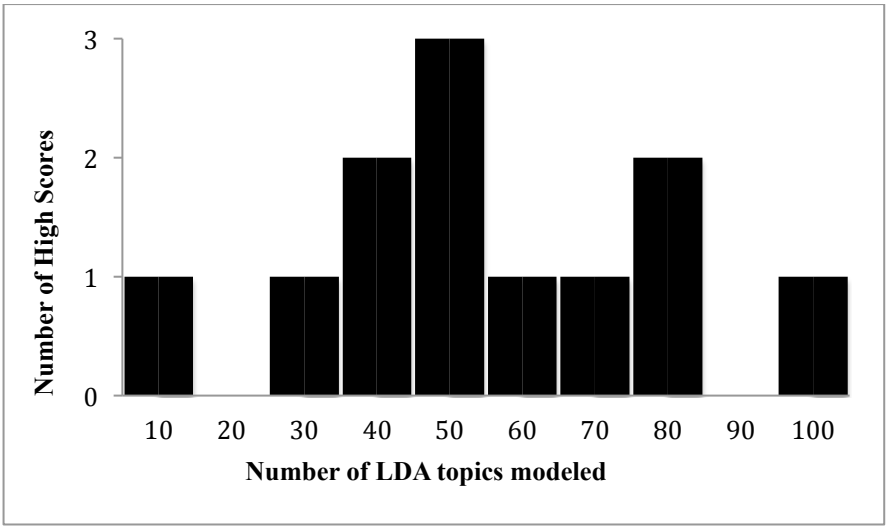


Figure 5.2: The number of high F1 scores per amount of LDA topics modeled.

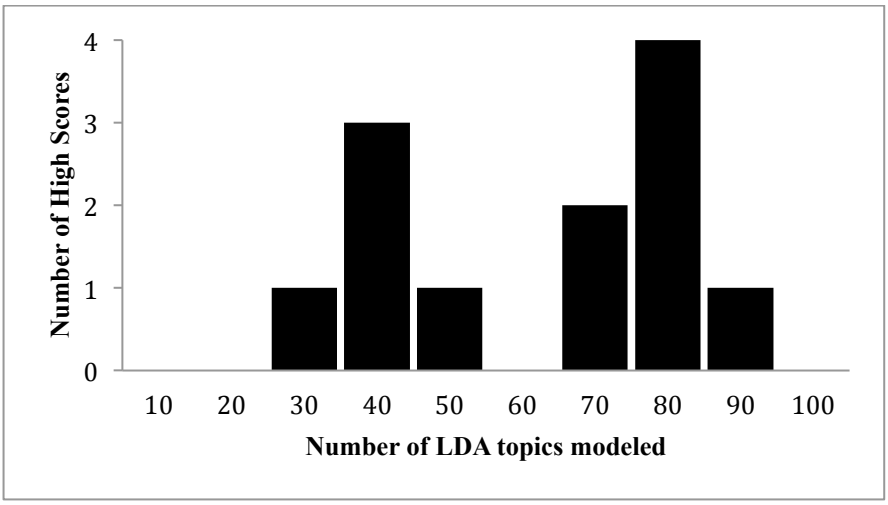


Figure 5.3: The number of high accuracy models per amount of LDA topics for 12 models. The majority of high accuracy models had either 40 or 80 LDA topics. Only one model had a high accuracy score out of either SVM or Naïve Bayes for 10 or 100 topics.

The alpha value applied to the regularization parameter in the SVM also has an effect on the accuracy of the classifier. If the value of alpha is too small or too large then the classifier will start classifying all documents as positives. For example, the dataset from r/learnprogramming in Table 5.5 uses the simple BOW produces a confusion matrix with mostly positive examples when $\alpha = 1.0\ x\ 10^{-13}$:

Table 5.5: Results from r/learnprogramming with a very low alpha value.

R/learnprogramming	Predicted positive	Predicted negative
True positive	185	97
True negative	121	84

This effect is shown further in Table 5.6, in which case the alpha is too large, in this case $\alpha = 10$:

Table 5.6: Results from r/learnprogramming with a very high alpha value.

R/learnprogramming	Predicted positive	Predicted negative
True positive	282	0
True negative	204	1

It is entirely possible, then, to erroneously obtain a high or low accuracy and F1 score due to an inappropriate alpha value. This effect was alleviated by tuning the alpha parameter for each dataset and for each feature selection method. In addition, a number of LDA topics were tested during the validation stage, typically at 10 topic intervals for both NBC and SVM models, with the alpha parameter also tuned when using the SVM.

In regards to the datasets themselves, the classifiers mostly performed the best with those from "niche" communities such as r/xxfitness and r/learnprogramming. The exception to this was r/learnpython, which had a high F1 score (0.7306 using the SVM with 10 LDA topics) but low accuracy (ranging between 0.5144 to 0.5965). Datasets from more general subreddits (r/askmen and r/askwomen) where users can ask questions on a variety of topics suffered lower accuracy and F1 scores. For example, r/askwomen's accuracy ranged from 0.5226 to 0.6032 and an F1 score of 0.5380 to 0.6154. This can be compared to the accuracy range of the niche community r/xxfitness, which had an accuracy range of 0.6210 to 0.7323 and an F1 range of 0.6380 to 0.7181. In this case the low end of both accuracy and F1 ranges are still several points higher for the more niche community than for the more general dataset. With wide variety of topics inundating the

community, it may be hard for the classifier to separate topics that are that are popular versus unpopular, thus resulting in lower classification accuracy.

In order to determine the feasibility of cross-domain application of these classifier systems, two subreddits were paired together based on similarity of general theme (e.g. R/xxfitness and r/fitness). The dataset from one subreddit was split evenly at random into testing and validation subsets and the other paired subreddit used as the training dataset. The goal of this process was to determine if there were enough shared topics among both subreddits such that a popular post in one subreddit could likely be popular in another subreddit. In this experiment the popularity cutoff of the original training set was used to label popular texts from the validation and testing sets. While most models suffered decreases in accuracy upwards of four or five points, most models still had accuracy that was better than chance. In half the datasets the F1 scores suffered significantly; for example, when predicting the popularity of a r/xxfitness post if it were to be posted in r/fitness, the F1 score dropped to a measly 0.5250, as compared to the original model which had an F1 of 0.7181.

In terms of the classification models themselves, SVM performed slightly better among the mixed datasets in terms of accuracy such that the highest accuracy models were split evenly between NBC and SVM. There was a reversal in F1 scores, however, meaning that NBC often had higher F1 scores than the SVM. In terms of feature selection, LDA still performed better than TF-IDF for both F1 and accuracy, but TF-IDF only had slightly higher accuracy and slightly lower F1 scores than BOW. Among each data set no discernible pattern was found such that one method had improved

performance compared to another. In a sense, sometimes the BOW had higher accuracy than TF-IDF and LDA, but other times TF-IDF or LDA was better than BOW. Ultimately it appears that the performance of each feature selection method was dependent on the dataset.

In addition to changes in model performance, another interesting result occurred when using two separate domains for testing and training: A dataset from subreddit A was likelier to be more accurate when using the subreddit B as testing, than if the other subreddit B was used to train the model to test subreddit A's data. Such was the case for r/learnpython, which had much better F1 scores and accuracy when tested on r/learnprogramming's dataset than when r/learnprogramming was used to create the model to test r/learnpython. This may be in part due to how the validation and testing set were randomly divided but also partly due to the quality of data in then testing set. The cross-domain models that performed better on one subreddit usually had similar high performance in the single-domain model from the test set. For example, r/learnpython had better performance when predicting the r/learnprogramming test set, but this may because r/learnprogramming itself had similar high accuracy during the single domain experiment. The subreddits r/askwomen and r/askmen had similar performance to each other during cross-domain prediction but also similar performance during single-domain prediction. Other factors such as the popularity cut-off score do not seem to have much influence: r/learnpython and r/learnprogramming had had a smaller difference in cutoff scores at 7.0 and 9.0, respectively, than r/askmen (37.0) and r/askwomen (33.5) which did not experience similar differences in cross-domain prediction accuracy. Thus the

dataset itself is likelier to be a factor in cross-domain popularity prediction. The results are portrayed in Table 5.7.

Table 5.7 (continued below): Results from the cross-domain popularity prediction experiment for each subreddit dataset. From A to B in the top left corner indicates that subreddit A was used as the training set and B was used as the test set. The shaded boxes indicate highest accuracy and F1 scores. The t variable under LDA indicates how many topics were modeled during feature selection.

r/learnprogramming	NBC		SVM	
to r/learnpython	Accuracy	F1:	Accuracy:	F1:
BOW	0.5689	0.4457	0.5200	0.4375
TF-IDF	0.5259	0.5876	0.5007	0.4374
LDA	0.6311 t = 90	0.6104 t = 30	0.5407 t = 100	0.5763 t = 90

r/learnpython	NBC		SVM	
to r/learnprogramming	Accuracy	F1:	Accuracy:	F1:
BOW	0.6140	0.7226	0.5879	0.7070
TF-IDF	0.6030	0.7394	0.6085	0.7476
LDA	0.6264 t = 50	0.7302 t = 70	0.6058 t = 40	0.7487 t = 10

r/fitness to r/xxfitness	NBC Accuracy	NBC F1:	SVM Accuracy:	SVM F1:
BOW	0.7071	0.7527	0.6486	0.7303
TF-IDF	0.6400	0.7515	0.6950	0.4500
LDA	0.6571 t = 90	0.7116 t = 70	0.6229 t = 30	0.7417 t = 30

r/xxfitness to r/fitness	NBC Accuracy	NBC F1:	SVM Accuracy:	SVM F1:
BOW	0.6707	0.5250	0.6603	0.4645
TF-IDF	0.6828	0.4986	0.6950	0.4500
LDA	0.6187 t = 20	0.5067 t = 20	0.6655 t = 10	0.4513 t = 80

r/askmen to r/askwomen	NBC Accuracy	NBC F1:	SVM Accuracy:	SVM F1:
BOW	0.5517	0.5498	0.5366	0.5275
TF-IDF	0.5409	0.5400	0.5517	0.5048
LDA	0.5539 t = 70	0.5801 t = 70	0.5647 t = 70	0.5313 t = 40

r/askwomen to r/askmen	NBC		SVM	
	Accuracy	F1:	Accuracy:	F1:
BOW	0.5323	0.5701	0.5439	0.5201
TF-IDF	0.5373	0.5766	0.5489	0.5627
LDA	0.5323 t = 80	0.5639 t = 100	0.5439 t = 80	0.5643 t = 70

Chapter 6: Conclusion

6.1 Summary and General Discussion

The results show that it is possible to predict the popularity of a subreddit's post to some extent based on the words it contains. Judging from the improvement when a simple bag of words representation was augmented with scores from TF-IDF or LDA, the topic of a document appears to be useful in predicting its popularity. Performance was better for posts in more niche communities than those in general communities. One could argue that topic identification is harder in more general communities since they entertain more diverse topics. So again the result, at least indirectly, suggests that document topic is a useful feature for popularity prediction.

Interestingly, comparison between different topic identification methods and different classification methods showed unexpected results. LDA, which is a more sophisticated frequency-based topic identification model than TF-IDF, did not always lead to better performance. Similarly, SVMs, which are often claimed to outperform NBCs, did not necessarily predict popularity better. The reasons for the atypical results are not clear at this point: Perhaps the data set was not large enough. Perhaps popularity prediction, especially for a platform like Reddit, is a difficult problem to begin with.

It is worth noting several factors to consider for better data collection and understanding the difficulty of the problem. As discussed earlier, important factors that influence prediction and classification include time effects and the length of the posts themselves. During data collection, only posts with more than 20 words were collected. If data collection had included concise posts that contained 10 or 15 words, the results

May have changed. For example, lowering the text length cutoff would have increased the dataset size and possibly increased model accuracy. Alternatively, lowering the text length may cause certain posts that have fewer topics to begin with to be inaccurately labeled. In addition, it may be true that topics that were once previously popular are no longer popular within a particular subreddit. If an older text is used to build the set of topics relating to popular posts, a newer text in a subreddit that has experienced a shift in topic popularities may be inaccurately labeled.

Lastly, the human component must never be forgotten when conducting popularity prediction research. Although explicitly prohibited by Reddit, it is possible that cliques have formed within a subreddit to upvote each other's posts or to downvote those of people they do not like. Another issue is that users who are generally disliked by the community (due to trolling behavior, consistent off-topic posts, or other factors) may see their post downvoted quickly, regardless of the topics contained in the post. These issues may have had a part in decreasing the accuracy of the classification models.

6.2 Further research

Future research could try to reduce some of the limitations previously mentioned. For example, future study could indicate how much of an effect a particular Redditor has on the popularity of their own post. In addition, the comments of popular users in a particular subreddit could be used to determine if a post is likely to be popular. This research would expand on the study in [11] where the researchers looked at the role of "mavericks" and "conformers" in swaying the popularity of online content. For example, if a user has a consistent amount of upvoted comments for a subreddit, does his comment

on a particular post from a different user contribute to the post's rise or downfall? If commenters or posters can influence popularity of content, further study could determine the degree to which it does.

Another area of research could examine to what effect topics wax and wane in popularity over time. That is, classification models could try to determine *when* a topic begins to wane in popularity by tracking the scores associated with posts that contain that topic. These models could also determine how much a reduction in topic popularity correlates with a reduction in model accuracy.

One last area of research could look at the effect of text on image-based posts or "memes," similar to the study conducted in [14] that examined how titles could influence popularity on Reddit. While such a study would require more machine learning processes in order to extract the text from the image or meme, pursuing such an endeavor could be valuable considering that the subreddits with the most subscribers tend to be image-based. One caveat is that memes and images tend to have a minimal number of words, which may make prediction difficult. Such a study would be interesting in its own right but could also be used for commercial endeavors in building recommender systems and developing targeted advertising.

References

[1] Reddit.com, "reddit.com: about reddit", 2012. [Online]. Available: https://www.reddit.com/about/. [Accessed: 20- Dec- 2015].

[2] P. Singer, F. Flock, C. Meinhart, E. Zeitfogel and M. Strohmaier, “Evolution of Reddit: from the front page of the internet to a self-referential community?”, in *23rd International "on World Wide Web*, Seoul, South Korea, 2014, pp. 517-522.

[3] B. Yu, M. Chen and L. Kwok, “Toward predicting popularity of social marketing messages.”, in *4th International Conference, SBP 2011*, College Park, MD, 2011, pp. 317-324.

[4] T. Weninger, X. Zhu and J. Han, “An exploration of discussion threads in social news sites: a case study of the Reddit community.”, in *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Niagara Falls, ON, Canada, 2013, pp. 579-583.

[5] T. Misera, "metareddit - all about reddit", *Metareddit.com*, 2015. [Online]. Available: http://metareddit.com. [Accessed: 20- Dec- 2015].

[6] A. Tatar, J. Leguay, P. Antoniadis, A. Limbourg, M. De Amorim and S. Fdida, “Predicting the popularity of online articles based on user comments”, *Proceedings of the International Conference on Web Intelligence, Mining and Semantics - WIMS '11*, 2011.

[7] Reddit Help, "What constitutes vote cheating or vote manipulation?". [Online]. Available: https://reddit.zendesk.com/hc/en-us/articles/205192985. [Accessed: 20- Dec- 2015].

[8] Reddit.com, "reddit.com: api documentation", 2015. [Online]. Available: https://www.reddit.com/dev/api#POST_api_vote. [Accessed: 20- Dec- 2015].

[9] Deimorz, "[reddit change] The logic for archiving posts has been changed slightly • /r/changelog", *reddit*, 2014. [Online]. Available: https://www.reddit.com/r/changelog/comments/25kvjo/reddit_change_the_logic_for_archiving_posts_has/. [Accessed: 20- Dec- 2015].

[10] A. Salihefendic, "How Reddit ranking algorithms work — Hacking and Gonzo", *Medium*. [Online]. Available: https://medium.com/hacking-and-gonzo/how-reddit-ranking-algorithms-work-ef111e33d0d9. [Accessed: 20- Dec- 2015].

[11] P. Yin, P. Luo, M. Wang and W. Lee, “A straw shows which way the wind blows”, *Proceedings of the fifth ACM international conference on Web search and data mining - WSDM '12*, 2012.

[12] G. Szabo and B. Huberman, 'Predicting the popularity of online content', *Communications of the ACM*, vol. 53, no. 8, p. 80, 2010.

[13] E. Gilbert, "Widespread underprovision on Reddit", *Proceedings of the 2013 conference on Computer supported cooperative work - CSCW '13*, 2013.

[14] H. Lakkaraju, J. J. Mcauley, J. Leskovec "What's in a name? Understanding the interplay between titles, content, and communities in social media". *ICWSM*, 2013.

[15] N. Barbieri, G. Manco, E. Ritacco, M. Carnuccio and A. Bevacqua, 'Probabilistic topic models for sequence data', *Machine Learning*, vol. 93, no. 1, pp. 5-29, 2013.

[16] S. Siersdorfer, S. Chelaru, J. Pedro, I. Altingovde and W. Nejdl, 'Analyzing and Mining Comments and Comment Ratings on the Social Web', *ACM Trans. Web*, vol. 8, no. 3, pp. 1-39, 2014.

[17] J. Martineau, T. Finin, A. Joshi and S. Patel, "Improving binary classification on text problems using differential word features", *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09*, 2009.

[18] Z. Hu, J. Yao, B. Cui and E. Xing, "Community Level Diffusion Extraction", in *2015 ACM SIGMOD International Conference on Management of Data*, Melbourne, AU, 2015, pp. 1555-1569.

[19] Redditmetrics.com, "New subreddits by date - reddit history", 2015. [Online]. Available: http://redditmetrics.com/history. [Accessed: 21- Dec- 2015].

[20] C. Claridge, 'Constructing a corpus from the web: message boards', in *Corpus Linguistics and the Web*, 1st ed., M. Hundt, N. Nesselhauf and C. Biewer, Ed. Amsterdam: Rodopi, 2006, pp. 87-108.

[21] Z. Ma, A. Sun and G. Cong, "Will this #hashtag be popular tomorrow?", *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '12*, 2012.

[22] F. Sebastiani, 'Machine learning in automated text categorization', *CSUR*, vol. 34, no. 1, pp. 1-47, 2002.

[23] W. Zhang, T. Yoshida and X. Tang, 'A comparative study of TF*IDF, LSI and multi-words for text classification', *Expert Systems with Applications*, vol. 38, no. 3, pp. 2758-2765, 2011.

[24] T. Joachims, *Text categorization with support vector machines*. Dortmund: Dekanat Informatik, Univ, 1997.]

[25] B. Trstenjak, S. Mikac and D. Donko, 'KNN with TF-IDF based Framework for Text Categorization', *Procedia Engineering*, vol. 69, pp. 1356-1364, 2014.

[26] A. Ng, "Bias vs. Variance", https://www.coursera.org, 2011.

[27] A. Mccallum and K. Nigam, “A comparison of event models for naive Bayes text classification” in *AAAI-98 workshop on learning for text categorization*, Madison, WI, 1998.

[28] D. Blei, A. Ng and M. Jordan, “Latent Dirichlet Allocation”, in *NIPS 2001*, Vancouver, British Columbia, CA, 2001.

[29] Q. Yuan, G. Cong and N. Thalmann, “Enhancing naive bayes with various smoothing methods for short text classification”, *Proceedings of the 21st international conference companion on World Wide Web - WWW '12 Companion*, 2012.

[30] N. Dewdney, C. Vaness-Dykema and R. Macmillan, “The form is the substance”, *Proceedings of the workshop on Human Language Technology and Knowledge Management*, 2001.

[31] S. Tong and D. Koller, 'Support vector machine active learning with applications to text classification', *Journal of Machine Learning Research*, vol. 2, no. 2002, pp. 45-66, 2002.

[32] K. Somasundaram and G. Murphy, “Automatic categorization of bug reports using Latent Dirichlet Allocation”, *Proceedings of the 5th India Software Engineering Conference on - ISEC '12*, 2012.

[33] S. Zeilikovitz, 'Using background knowledge to improve text classification', Ph.D, Rutgers University, 2015.

[34] S. Jamali, 'Comment Mining, Popularity Prediction, and Social Network Analysis', Master's of Science, George Mason University, 2006.

[35] P. Soucy and G. Mineau, “Beyond TFIDF weighting for text categorization in the vector space model”, in *Proceedings of the 19th international joint conference on Artificial intelligence*, Edinburgh, Scotland, 2005, pp. 1130-1135.

[36] M. Hundt, N. Nesselhauf and C. Biewer, *Corpus linguistics and the web*. Amsterdam: Rodopi, 2007.

[37] H. Li and K. Yamanishi, 'Topic analysis using a finite mixture model', *Information Processing & Management*, vol. 39, no. 4, pp. 521-541, 2003.

[38] D. Blei, “Probabilistic topic models”, *Communications of the ACM*, vol. 55, no. 4, p. 77, 2012.

[39] A. Sun, E. Lim and Y. Liu, 'On strategies for imbalanced text classification using SVM: A comparative study', *Decision Support Systems*, vol. 48, no. 1, pp. 191-201,

2009.

[40] S. Lu, D. Chiang, H. Keh and H. Huang, 'Chinese text classification by the Naïve Bayes Classifier and the associative classifier with multiple confidence threshold values', *Knowledge-Based Systems*, vol. 23, no. 6, pp. 598-604, 2010.

[41] M. Danilevsky, C. Wang, N. Desai, X. Ren, J. Guo and J. Han, "Automatic Construction and Ranking of Topical Keyphrases on Collections of Short Documents", in *SIAM International Conference on Data Mining*, Philadelphia, PA, 2014, pp. 1-9.

[42] A. Ng, "Gradient Descent for Multiple Variables", https://www.coursera.org, 2011.