N-Gram Representations For Comment Filtering

Dirk Brand
Computer Science Division
Stellenbosch University
dirkbrand@ml.sun.ac.za

Steve Kroon
Computer Science Division
Stellenbosch University
kroon@sun.ac.za

Brink van der Merwe
Computer Science Division
Stellenbosch University
abvdm@cs.sun.ac.za

Loek Cleophas
Dept. of Computing Science,
Umeå University
Dept. of Information Science,
Stellenbosch University
loek@fastar.org

ABSTRACT

Accurate classifiers for short texts are valuable assets in many applications. Especially in online communities, where users contribute to content in the form of posts and comments, an effective way of automatically categorising posts proves highly valuable. This paper investigates the use of N-grams as features for short text classification, and compares it to manual feature design techniques that have been popular in this domain. We find that the N-gram representations greatly outperform manual feature extraction techniques.

CCS Concepts

•**Information systems** → **Document representation;** *Sentiment analysis;* Information extraction; •**Computing methodologies** → **Supervised learning by classification;** *Knowledge representation and reasoning;*

Keywords

Classification, information retrieval, vector space models, feature design, N-gram models, NLP, text mining

1. INTRODUCTION

The problem of identifying and assessing the quality of short texts (e.g. comments, reviews or web searches) has been intensively studied since 2008 [15, 46, 27]. There are great benefits in 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 to content. Analysing short texts is a difficult problem, because traditional machine learning models generally perform better on larger samples. More data allow for better estimation of parameters for these models.

SAICSIT '15, September 28 - 30, 2015, Stellenbosch, South Africa
© 2015 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-3683-3/15/09...$15.00
DOI: http://dx.doi.org/10.1145/2815782.2815789

Short texts generally do not have much content, but still carry high variability in that they may use a large corpus of words. This makes it difficult to build a representative feature space for short texts [15].

We investigate the problem of classifying short texts in the context of website comment filtering, where the comments are filtered by an editor according to how suitable they are for live content. Internet news providers often allow users to contribute content in the form of comments. These comments are often not written in standard English and contain many colloquialisms and linguistic phenomena (similar to tweets). The success of a news website is greatly dependent on these comments, as they facilitate discussion, which boosts user engagement. Unfortunately, some users contribute malicious content that is often defamatory, anti-social, or racist. This can not be allowed if the news provider seeks legitimacy in the news space. We investigate feature extraction techniques for automatic detection of antisocial behaviour.

Many learning methods have been used for short text classification, including k-nearest neighbours (kNN) [24], naïve Bayes [22, 57] and Support Vector Machines (SVMs) [4]. SVM-based methods are particularly popular for this problem [63], because SVMs are very versatile and can easily deal with both dense and sparse data representations. A variety of kernels can also be applied to represent various priors on the data distribution. The quality of SVM-based methods depends on a variety of factors, but most notably the choice of the kernel and the quality of the training data [31]. Preprocessing input data is important for training SVMs: most importantly, the data should first be transformed into features of a type that can be processed by the specific kernel function (most kernels use numeric features). Therefore, an important focus for improving the quality of short comment classification is feature extraction [56, 15, 63], where proposed approaches are typically compared using one or two standard SVM kernels.

In [10], the efficacy of a simple bag-of-words model in comparison to manually designed features for regression to address a different problem, but in the same domain, i.e. Internet comments, was investigated. Results showed that neither of these models are sufficiently accurate. The chief focus of this paper is an investigation into N-gram-based approaches (e.g. bag-of-words models) for feature extraction, using the manually designed features as a baseline. N-gram models have been hugely successful in other natural language processing tasks [41, 20], but their performance on short texts such as Internet comments, where the representations will likely be very sparse, is not well-studied. This paper thus provides an empirical evaluation that sheds some additional light on whether N-gram models are suitable for short text classification. All experiments are performed using a representative data set of comments from an Internet news source.

The rest of the paper is structured as follows. First, Section 2 provides context for our work by presenting related research in the field of short text classification. Thereafter, some background information on the N-gram approaches is detailed in Section 3. Then, Section 4 discusses the source of the data and explains the methodology we follow to produce the various feature sets. Finally, the results of the study are given in Section 5, followed by our conclusions and future work (Section 6).

2. PREVIOUS WORK

Our work leans heavily on previous studies of both classification and ranking of comments. No studies were found that specifically use N-gram-based approaches for Internet comment classification, however N-grams have been used in other short text classification tasks.

Cavnar and Trenkle used N-gram-based techniques for building word frequency profiles for longer documents and used these profiles to categorize the documents under consideration [13]. They achieved a very high accuracy using the statistical properties of simple character N-grams of lengths 1 to 5.

Mishne investigated the accuracy of using N-gram approaches for classifying the mood or sentiment of writers of blog posts [41]. 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 "bag-of-words" approaches with their own domain-specific features to categorise texts [57]. 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.

Lampe and Resnick [35] used the properties of the comments left by users (comment length, word usage), as well as the properties of the authors themselves (frequency of posting, frequency of response) in order to classify comments. Wanas et al. [63] sought to improve on the work done by Lampe and Resnick. The features that Wanas et al. used, were based on features designed by Weimer et al. [64], and consisted of various features categorised into five classes. These classes were relevance, originality, forum-specific, surface (frequency of capitalised words, quality of grammar, etc.) and posting component (presence and quality of weblinks in posts) features. They focused their investigation on designing features that take various linguistic phenomena, present in online forums, into account.

3. N-GRAM MODELS

Since designing and creating a manual feature set is a time-consuming process, and since a user can manipulate the system if they know the features that are being used, we rather investigate alternative representations for comments. In information retrieval a piece of text is often represented by certain keywords or terms [47]. A set of weights can also be associated with these terms to show their relative importance to the text [52]. This idea of text representation is often called the *N-gram model*, which is a specific type of vector space model for texts [53]. It has been shown that N-gram representations can be trained on a wide variety of linguistic tasks [11].

An N-gram is a series of objects (letters, words, syllables, or other linguistic units) from a longer piece of sample text. An N-gram is often taken to be a contiguous sequence, but it could be any co-occurring set of objects (e.g. the first and third character of words, i.e. skip-grams [16]). Unless stated otherwise, we consider N-grams as contiguous sequences of words or characters. 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 and are sliced so that N-grams overlap. 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".

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 and $k-2$ trigrams.

N-gram representations 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 statistical machine translation) [5], DNA sequencing [59], and compression algorithms [29].

4. METHODOLOGY

The N-gram-based feature sets are based on popular techniques in information retrieval [47] and are part of a class of data representations often referred to as vector space models [53]. The baseline manual feature set is based on work done by previous authors [43, 49, 30], as well as some additional features devised by us.

All of the constructed feature sets are used to train SVM classifiers [62, 18], which are then evaluated and compared by using standard metrics, including accuracy, precision, recall and F1-scores. The Radial Basis Function (RBF) and linear kernels [32] for SVM will be considered by using the SVM classifier from the Scikit-Learn Python library [45]. These are very common kernels, and previous studies on comment classification have also made use of them [30]. These kernels are also attractive for N-gram techniques, since they can efficiently deal with the sparse matrices induced by our N-gram representations.

The data we use are comments obtained in cooperation with News24 (further discussed in Section 4.1). This input data set has to undergo various transformations to be suitable as training data for the SVM classifiers. This is shown in Figure 1 and detailed in the following sections.

The training sets are labelled data sets with each data point represented by a feature vector and an associated label (called the *class value*). Each feature vector represents a comment by a feature representation model, as detailed in Section 4.2. Then, feature selection is applied to reduce the dimension of the feature set. This is necessary due to constraints on the time required to train the SVM classifiers and is discussed in Section 4.3. Each of these features are normalized so that its l^2 norm (i.e. the sum of the squared values for that feature over all samples) equals 1. Normalizing the data leads to improved performance in algorithms such as Support Vector Machines (depending on the kernel) [26, 31, 23]. This is further detailed in Section 4.4. Finally, a classifier is trained to predict the class value of a training feature vector for each data set. These trained classifiers are used to predict the value of an unlabelled feature vector (representing a new comment). The best choice of feature representation is our chief topic of investigation.

4.1 The Data Set

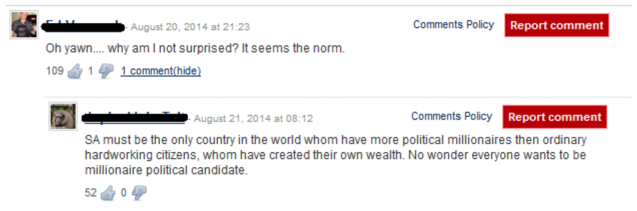
News24 provided us with a data set containing articles and comments that were left on these articles. Metadata about both the articles and comments are included (e.g. author name, date of posting and article title). The comments themselves are used as input data for training our models.

Thus, the input data set consists of N comments, denoted as $\{c_1, c_2, ..., c_N\}$. For each comment c_i, a set of m features $F_{c_i} = \{f_1, f_2, ..., f_m\}$ is extracted. Thus, a candidate feature set consists of rows of the form $\{(F_{c_1}, r_{c_1}), ..., (F_{c_N}, r_{c_N})\}$, where a tuple (F_{c_i}, r_{c_i}) indicates a feature set F_{c_i} for comment c_i, and the associated class value r_{c_i}.

We found experimentally that comments with fewer than 20 words are hard to classify with the methods described in this paper and are, as such, not included in the data set. Thus, the data set used consist of 79017 samples, but the number of features in each feature set differs depending on the representation.

News24 allows its users to leave comments on news articles. A user can either leave a comment on an article directly (referred to as a *parent comment*) or reply to a parent comment (referred to as a *child comment*). Figure 2 shows an excerpt from a comment thread where one user has posted a comment and another user replied to that comment. Each parent comment can have multiple replies, forming a *thread.* Each article typically has multiple parent comment threads associated with it. The collection of threads on an article is referred to as the *article comments.* Figure 3 shows statistics describing the data set.

Users are also able to vote on comments in the form of likes and dislikes, as well as report comments that they feel are of low quality (e.g. that they consider demeaning or defamatory). Figure 2 shows an example of likes and dislikes attributed to a comment. The editorial team can then decide whether the comment should be removed from the site (i.e. be made *hidden*). The editors also have automatic filters, based on high-level criteria (as discussed below), for



[redacted] - August 20, 2014 at 21:23

Oh yawn.... why am I not surprised? It seems the norm.

[redacted] - August 21, 2014 at 08:12

SA must be the only country in the world whom have more political millionaires then ordinary hardworking citizens, whom have created their own wealth. No wonder everyone wants to be millionaire political candidate.

Figure 2: Part of a typical News24 comment thread.

Original number of comments	130713
Number of comments with $\geq$ 20 words	79017
Number of parent comments	56124
Number of child comments	22893
Average number of child comments per parent	0.48
Average number of comments per article	16.24
Total number of words	7.3 Million
Average number of words per comment	93.32
Percentage of ‘hidden’ comments	34.8%

Figure 3: Corpus statistics. Numbers are listed for comments with twenty or more words.

removing comments. Thus, some comments are visible and some are hidden. We present the following goal: to predict this status for unlabelled comments (typically newly posted comments) automatically, i.e. to classify a comment to reflect the ideological orientation of the editors.

After a comment is reported, editors remove the comment (i.e. make “hidden”) based on whether:

- it contains abusive language, hate speech or profanity,
- it contains completely incorrect grammar,
- it includes “text-speak”[1],
- it includes nicknames or insulting names for the individual the article is about,
- it refers to racial stereotypes or contains racial slurs.

It is relevant to note that News24 is primarily based in South Africa, so the user comments typically feature a unique language domain, known as South African English (SAE). SAE contains various colloquialisms and slang that are specific to the South African context. SAE may also include words from other official South African languages (e.g. Zulu, Xhosa or Afrikaans). Comments that a provided language model [54] identified as being predominantly English were considered in this work. Thus, comments can still contain words from other languages which pollute the standard English vocabulary.

4.2 Feature Extraction

Both word and character N-grams are investigated. 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 — we consider unigrams ($N = 1$), bigrams ($N = 2$) and trigrams ($N = 3$) (detailed in Section 4.2.2). Other N-gram representations

[1]Internet colloquialisms and abbreviations.

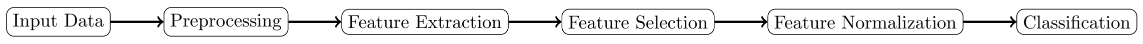


Figure 1: The pipeline that a comment goes through to be classified.

with $N > 3$ could be used, but the resulting feature vectors are extremely sparse, making them unsuitable for training a classifier. For each choice of N, we then consider either N-gram features alone (denoted by "="), or a feature vector using all N-grams of that order or lower (denoted by "$\leq$"). Finally, we consider three different vectorization methods for determining the value of each component in the resulting comment vector: the binary count (denoted by "B"), the frequency count[2] (denoted by "F") and the TFIDF-normalized frequency (denoted by "T"). These three representations are also explained in detail in Section 4.2.2. We summarize the selection of N, whether lower-order N-grams are included or not, and the vectorization method in our notation by concatenating N and our symbols denoting the other choices. For example, "$\leq$3T" represents the use of unigrams, bigrams and trigrams with TFIDF-normalized vectorization.

We decided to use character N-grams 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 use bad words or derogatory terms without being filtered, by changing the spelling of the words. We also investigate character skip-grams, since they are even better suited for identifying cases where a user changes a letter in a word to obfuscate the word.

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 TFIDF-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 single characters left out, are included (i.e. the character skip-grams for the word "bird" are "ird", "brd", "bid" and "bir"). Again, the binary and TFIDF-normalized frequency vectorization methods are used. This produces two feature sets, namely CSB and CST.

4.2.1 Preprocessing

For the word N-gram representations, two types of preprocessing are done before the feature vectors are generated: stop-word removal and lemmatization. The character N-grams are constructed without any preprocessing.

Before the feature sets are constructed, stop-words [48] are removed from the comments. The list of stop-words is taken from the `python NLTK` corpora [7]. Also, words are transformed into their base dictionary form (called the lemma) through a process called lemmatization. As an example, the words *car*, *cars*, *car's* and *cars'* are all mapped to *car*. This is done to group plurals and other word variations into a single representative term. Lemmatization is performed with an implementation from `NLTK` [7], using the WordNet [40] lexical database.

4.2.2 Constructing N-gram Representations

Let V be the vocabulary of all terms (or N-grams) in the corpus. Then each data sample is represented by a vector of term weights $\langle e_{i1}, ...e_{i|V|}\rangle$ and a class value $y_i \in \{0, 1\}$. Here e_{ij} is the weight of term j in sample i and $|V|$ is the number of terms in the corpus. Depending on the representation, the terms in the vocabulary are either the distinct words (unigrams) or sequence of terms (bigrams or trigrams), or the character N-grams for $N \in \{2, 3, 4, 5, 6, 7, 8\}$, and are referred to as term-features.

All the representations are implemented using a row-wise sparse data representations from `scipy` [33]. 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 $N > 1$.

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

- **Binary Count**. For each comment C_i, the j^{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 [38]. For each comment C_i, the j^{th} term-feature will contain the frequency of the N-gram in the comment.

- **TFIDF**[3]. Longer documents 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 [51]. For each comment C_i, the j^{th} term-feature is given by:

$$\text{tfidf}(t_j, C_i) = t_j * \ln\left(\frac{|C|}{f(w_j, C) + 1}\right)$$

  where t_j is the frequency of term w_j in comment C_i, normalised by the number of words in the comment, and the second term is the natural logarithm of the total number of comments C divided by the number of comments w_j appears in.[4] This is based on the implementation used by the Python library `scikit-learn` [45] for TFIDF.

[2]The frequency count is only investigated for unigrams, since we found 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.

[3]TFIDF abbreviates *Term Frequency - Inverse Document Frequency [2]* Normalization.

[4]The frequency is artificially increased by 1 to avoid zero division errors. This adjustment can also be interpreted as smoothing via a suitable prior.

4.3 Feature Selection

Reducing the dimensionality of the feature space helps to reduce noise in the data by removing irrelevant and redundant features [58]. This could result in faster and better performance for many regression and classification models [25], as well as prevent over-fitting of a model [61].

A popular technique for dimensionality reduction is feature selection. Feature selection [12] involves selecting a subset of the original features to maximize the relevance of the feature set for predicting the class variable. To identify the most relevant features, a popular type of feature selection, called univariate feature selection, is applied. This means the algorithm considers each feature's relationship to the class variable independently and scores the relevance of the features to the class variable. The scoring mechanism used is the χ^2 (chi-squared) statistical test [37].

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 SVM is directly correlated to both the number of samples and the number of features (for both linear and RBF kernels) [8]. We determined experimentally that this arbitrary maximum does not significantly impact the results, but provides a substantial decrease in running time of experiments.

For illustration, the ten most relevant unigram, bigram, and trigram features are given below:

- Unigram - airtime, buy, monkey, customer, data, illuminati, enjoy, gb, get, message.
- Bigram - buy mtn, application form, http www, jacob zuma, psychic twitter, new promo, note wait, receive message, serial number, say dear.
- Trigram - jacob zuma finally, 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.

4.4 Feature Normalization

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 obervations 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) is equal to 1. Thus, zero value features are unaffected, which is advantageous for maintaining sparseness.

5. RESULTS

Before any features are extracted, the data set is partitioned into training and test sets that are stratified (i.e. the class distribution is maintained through the split). The training and test sets make up 60% and 40% of the data set (for each feature approach), respectively. After the partitioning, each comment in the training set is transformed through the pipeline in Figure 1 to form a final training set for training the classifiers.

Table 1: The number of features of each feature set.

Feature Representation	Number of Features
Manual Features	38
$= 1B$, $= 1F$, $= 1T$	43566
$= 2B$, $= 2T$	589597
$= 3B$, $= 3T$	866469
$\leq 2B$, $\leq 2T$	633163
$\leq 3B$, $\leq 3T$	1499632
$C28B$, $C28T$	4462649
CSB, CST	484006

Table 1 shows the number of features per training set for each feature representation (before dimensionality reduction).

5.1 Parameter Tuning

Techniques such as SVMs are often highly sensitive to the choice of parameters [17]. A grid search was 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 [6].

The cost parameter C, as well as the bandwidth parameter γ for the RBF kernel, was tuned. The search was done over a logarithmic distribution of values (10^{-1} to 10^7 for C and 10^{-4} to 10^1 for γ). Each value was tuned by taking a random subset of 10265 samples from the feature set (25% of the training samples) and doing a 3-fold cross-validated search over the parameters.[5] The resulting parameters are shown in Appendix B.[6]

5.2 Baseline Manual Features

The features used for the baseline manual feature set are discussed below. The features can be categorised into post and social features. The social features are "per-user" features and are calculated using social network analysis [50, 49, 1] on a user graph, called a sociogram [1, 14]. A summary of these features is presented in Table 5.2, with detailed descriptions in Appendix A. Even more comprehensive descriptions can be found in [10].

5.3 Evaluation

Classification performance is evaluated with various scoring metrics. For each technique, we present the accuracy, precision, recall and F1-score. We focus particularly on the F1-score [60], which is the harmonic mean of precision and recall [19]. This is preferable to the *accuracy* score, since a classifier could still achieve relatively high accuracy if it succeeds at classifying one class, but completely fails to classify the other, due to the imbalance in class size.

[5]Since the experiments are very time-consuming, parameter tuning with cross-validation on larger data sets was not feasible.

[6]The standard deviation of the 3-fold evaluation for each parameter pair for each feature set, was in the order of 10^{-3}.

Table 2: A summary of the manual features.

Post Features	Timeliness, lengthiness, part-of-speech count, uppercase frequency, question frequency, exclamation frequency, capitalized sentence frequency, comment complexity, spelling, profanity, readability, relevance, subjectivity, sentiment and comment-article sentiment overlap.
Social Features	In-degree, out-degree, user age, post count, post rate, PageRank value, hub and authority values (from Hyperlinked-Induced Topic Search).

Table 3: Results for RBF SVM classification.

Feature Set	Accuracy	Precision	Recall	F1
Manual Features	0.836	0.807	0.641	0.714
$= 1B$	0.856	0.812	0.719	**0.762**
$= 1F$	0.851	0.863	0.638	0.734
$= 1T$	0.844	0.780	0.717	0.747
$= 2B$	0.843	0.805	0.675	0.734
$= 2T$	0.834	0.770	0.690	0.728
$= 3B$	0.851	0.840	0.662	0.741
$= 3T$	0.828	0.750	0.699	0.723
$\leq 2B$	0.852	0.839	0.665	0.742
$\leq 2T$	0.838	0.779	0.690	0.732
$\leq 3B$	0.851	0.833	0.670	0.743
$\leq 3T$	0.806	0.689	0.719	0.703
$C28B$	0.852	0.870	0.634	0.733
$C28T$	0.813	0.695	**0.742**	0.717
CSB	**0.860**	**0.888**	0.645	0.747
CST	0.857	0.856	0.666	0.749

Results

Table 3 presents the performance scores of the different feature sets with the RBF kernel SVM classifier. Similarly, Table 4 presents the performance scores for the linear kernel SVM classifier.

Table 3 shows that nearly all the N-gram representations outperform the manual feature set on all metrics. There seems to be relatively small differences between the results for the binary and the TFIDF-normalization vectorization methods, although it seems that the TFIDF models perform slightly worse. It is surprising that the unigram models perform better than the higher-order N-gram representations, with the binary unigram model showing the highest F1-score. This implies that we get no additional information from higher order N-grams. However, this could be caused by the sparseness of the higher order N-gram representations, leading to weaker results.

The naïve character N-gram model (with TFIDF-normalization) showed the highest recall (by a relatively large margin), but comes at the cost of the lowest precision. The binary character skip-gram model had the highest accuracy and precision overall, but then at the cost of a lower recall. These results illustrate the precision-recall tradeoff, where higher precision can be obtained at the cost of lower recall (and vice versa). This highlights why we use the F1 score, which takes both precision and recall into account, as the primary basis of comparison for our models.

Table 4: Results for linear SVM classification.

Feature Set	Accuracy	Precision	Recall	F1
Manual Features	0.830	0.786	0.647	0.710
$= 1B$	**0.856**	0.825	0.699	0.757
$= 1F$	0.853	0.821	0.692	0.751
$= 1T$	0.849	0.804	0.701	0.749
$= 2B$	0.850	0.829	0.673	0.743
$= 2T$	0.847	0.814	0.679	0.740
$= 3B$	0.852	0.838	0.669	0.744
$= 3T$	0.852	0.843	0.705	**0.768**
$\leq 2B$	0.845	0.866	0.610	0.716
$\leq 2T$	0.849	0.819	0.681	0.744
$\leq 3B$	0.851	0.842	0.660	0.740
$\leq 3T$	0.845	0.806	0.682	0.739
$C28B$	0.843	0.868	0.601	0.711
$C28T$	0.802	0.671	**0.750**	0.709
CSB	0.848	**0.869**	0.621	0.724
CST	0.846	0.785	0.714	0.748

Table 4 shows better results for higher order N-grams than with lower order N-grams, with the TFIDF-normalized trigram feature set showing the highest F1-score. The character models again show promise, with the TFIDF-normalized character N-gram model achieving the highest recall score (by a relatively large margin), and the binary character skip-gram model having the highest precision. However, these results seem to be due to a similar precision-recall tradeoff as mentioned earlier.

6. CONCLUSION AND FUTURE WORK

We investigated the performance of N-gram approaches for automatically classifying short texts, using a compilation of leading techniques in feature extraction. We showed that all of the proposed N-gram representation approaches outperformed the manual feature set approach. Surprisingly, we did not obtain consistently improved performance from higher order N-gram models. While a trigram model gave the best results for the linear kernel by a narrow margin, the unigram representations were best for the RBF kernel, and second-best for the linear kernel. This may very well be caused by the sparsity of higher-order N-grams in our short text data set. Also, it might be explained by the observation that editors may predominantly *hide* comments based purely on the presence of certain words, in which case higher order N-grams should provide no additional gain over unigrams.

Character N-grams show great promise, as we used a naïve approach but already achieved notable results, with character skip-grams achieving the highest precision for SVM classification with both the RBF and linear kernels. Further investigation into more sophisticated character models should yield interesting results. In addition, a more careful investigation of what properties of these models lead to differing precision-recall tradeoffs could be insightful.

The feature extraction models we currently use can be improved. On this front, we would like to consider smoothing techniques and contextual language models [42] to improve on the N-gram representation techniques used in this paper. 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. Both of these ideas are loosely inspired by the research of Sidorov et al. [55].

Alternative feature extraction models, including deep learning techniques [28] and distributed representation models (e.g. word embeddings [39]), could also be considered.

Future work will consider alternative approaches for short text classification. These approaches will include both alternative SVM kernels (e.g. string kernels [36]), as well as alternative classification techniques. It would also be interesting to know which samples our classifiers failed to classify, so as to learn how to improve the techniques further.

7. SOURCE CODE

The source code for this paper is kept in a Github repository [9]. Unfortunately the data sets used in this study can not be made public; however the repository provides guidelines for setting up a data set.

8. ACKNOWLEDGEMENTS

Some of this work was done at the MIH Media Lab at Stellenbosch University.

9. REFERENCES

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

[4] A. Basu, C. Walters, and M. Shepherd. Support vector machines for text categorization. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 7–pp. IEEE, 2003.

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

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

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

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

[9] D. Brand. Comment Classification. https://github.com/DirkBrand/Comment-Classification, 2015.

[10] 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. UMICTA, ACM, 2014.

[11] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large language models in machine translation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.* Citeseer, 2007.

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

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

[14] 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.

[15] 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.

[16] W. Cheng, C. Greaves, and M. Warren. From n-gram to skipgram to concgram. *International Journal of Corpus Linguistics*, 11(4):411–433, 2006.

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

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

[19] H. Daumé III. A Course in Machine Learning. http://ciml.info, 2012.

[20] Z. Elberrichi and B. Aljohar. N-grams in Texts Categorization. *Scientific Journal of King Faisal University (Basic and Applied Sciences)*, 8(2):25–39, 2007.

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

[22] 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.

[23] S. R. Gunn. Support Vector Machines for Classification and Regression. *ISIS technical report*, 14, 1998.

[24] 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.

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

[26] R. Herbrich and T. Graepel. A PAC-Bayesian margin bound for linear classifiers. *Information Theory, IEEE*

Transactions on, 48(12):3140–3150, 2002.

[27] 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.

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

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

[30] 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.

[31] C. W. Hsu, C. C. Chang, and C. J. Lin. A Practical Guide to Support Vector Classification. *BJU international*, 101(1):1396–400, 2008.

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

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

[34] J. M. Kleinberg. Hubs, Authorities, and Communities. *ACM Computing Surveys (CSUR)*, 31:5, 1999.

[35] 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.

[36] 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.

[37] R. G. Lomax and D. L. Hahs-Vaughn. *Statistical concepts: a second course*. Routledge, 2013.

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

[39] 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.

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

[41] 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.

[42] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088, 2009.

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

[44] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. 1999.

[45] 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.

[46] 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.

[47] 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.

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

[49] 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.

[50] 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.

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

[52] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

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

[54] N. Shuyo. Language Detection Library for Java, 2010.

[55] 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.

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

[57] 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.

[58] J. Tang, S. Alelyani, and H. Liu. Feature Selection for Classification: A Review. *Data Classification: Algorithms and Applications*, page 37, 2014.

[59] 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.

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

[61] L. van der Maaten, E. Postma, and J. van den Herik. Dimensionality Reduction: A Comparative Review. *Journal of Machine Learning Research*, 10:1–41, 2009.

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

[63] 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.

[64] 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.

APPENDIX

A. MANUAL FEATURES

This appendix details the features used in the manual feature set.

- **Post Features**
  - *Timeliness.* This feature reflects the response time of a user's comment in relation to when the relevant article was posted [63].
  - *Lengthiness.* This feature is a simple measure of the length of a comment relative to the average length of comments of that article [63].
  - *Part-of-speech Count.* The number of verbs, nouns and pronouns in the comment are all used as features.
  - *Uppercase Frequency.* This feature is the frequency of words in the comment that are uppercase [30], as a percentage of the total words.
  - *Question Frequency.* This feature is the counts of the number of sentences in the comment that end in a question mark [64]. The value is given as a percentage of the total number of sentences.
  - *Exclamation Frequency.* This feature is the counts of the number of sentences in the comment that end in an exclamation mark [64]. The value is given as a percentage of the total number of sentences.
  - *Capitalized Sentence Frequency.* This feature determines the percentage of sentences in the comment starting with a capital letter.
  - *Complexity.* The complexity of a comment is measured by the entropy of the words in the comment [30]. Intuitively, it represents the diversity in word choice in the comment. A low entropy score would indicate that a comment has few or repetitive words.
  - *Spelling.* This feature measures the frequency of misspelled words in the comment. The feature is calculated by looking up each word in a dictionary and recording the percentage of words that cannot be found in the dictionary.
  - *Profanity.* This feature measures the frequency of profane words in the comment. Similar to the spelling feature, the feature value is calculated by looking up each word in a dictionary of profane language and recording the percentage of words that can be found in the list of banned words. The list is built from a list published by Alejandro U. Alvarez [3].
  - *Readability.* The readability of a comment is defined as with what ease the reader is able to read the comment (determined by the Flesch Reading Ease Test (FRES) [21]).
  - *Relevance.* The relevance of a comment can be measured relative to its enclosing article. To calculate the relevance of a comment to the article, the overlap between the words in the comment and the words in the article is quantified. For this, a bag-of-words vector of the 100 most frequent words is generated from the body of the article.
  - *Subjectivity.* The subjectivity of the comment is also captured as a feature. The polarity of the comment is determined by a classifier (as a probability distribution), and if the difference between a comment's positive and negative values is over 20% (i.e. the positive value is below 40% or above 60%), the comment is classified as objective, otherwise it is classified as subjective.
  - *Sentiment.* To determine the sentiment of a comment, a trained classifier (Naïve Bayes) was used to predict the sentiment of a comment. The classifier produces a probability distribution of positivity of the comment. This score is then used as a feature.
  - *Comment-Article Sentiment Overlap.* To capture the commenter's alignment to the article, the polarity of the article's content is compared to that of the comment. If it matches, a one is used as a feature, otherwise a zero.
- **Social Features**
  - *In-degree.* The number of comments that have been left as a child comment to this user's parent comment.
  - *Out-degree.* The number of parent comments that this user has commented on.
  - *User age.* The difference between the user's first post ("join date") and the user's last post (in days).
  - *Post count.* The number of comments the user has posted.
  - *Post rate.* The post count divided by the user age.
  - *PageRank.* The PageRank [44] value of the user as calculated on a sociagram (with users as nodes and comments as directed edges).
  - *HITS.* The authority and hub value of the comment, as calculated by the Hyperlink-Induced Topic Search algorithm [34].

B. TUNED PARAMETERS

Table 5 shows the parameters that were tuned with a grid search for the various feature sets.

Table 5: Parameters obtained through parameter tuning.

Feature Set	SVM (RBF)		SVM (Linear)
	C	γ	C
Manual Features	100000	10.0	10000
$= 1B$	10	0.072	1
$= 1F$	10	0.072	1
$= 1T$	10	0.072	1
$= 2B$	100	0.001	1
$= 2T$	10	0.072	1
$= 3B$	1	0.072	10
$= 3T$	100000	0.0001	10
$\leq 2B$	100	0.002	0.1
$\leq 2T$	100	0.013	10
$\leq 3B$	10000	0.01	1
$\leq 3T$	10	0.1	10
$C28B$	1000	10	0.001
$C28T$	1000	10	0.01
CSB	10	0.0001	0.001
CST	100	1.0	1.0