Sentiment polarity classification using statistical data compression models

Dominique Ziegelmayer and Rainer Schrader
Institut für Informatik
Universität zu Köln
Weyertal 80, 50931 Köln
{*ziegelmayer, schrader*}*@zpr.uni-koeln.de*

Abstract

With growing availability and popularity of user generated content, the discipline of sentiment analysis has come to the attention of many researchers. Existing work has mainly focused on either knowledge based methods or standard machine learning techniques. In this paper we investigate sentiment polarity classification based on adaptive statistical data compression models. We evaluate the classification performance of the lossless compression algorithm Prediction by Partial Matching (PPM) as well as compression based measures using PPM-like character n-gram frequency statistics. Comprehensive experiments on three corpora show that compression based methods are efficient, easy to apply and can compete with the accuracy of sophisticated classifiers such as support vector machines.

1. Introduction

Today, a huge amount of information is publicly available on the world wide web. Especially popular e-commerce and review sites offer a high number of evaluative, user generated texts. They have thus become a valuable source of opinions on various objects such as products, services and institutions. With growing availability of opinionated texts, the discipline of sentiment analysis has come to the attention of many researchers and organizations.

Most of the existing work on sentiment analysis focuses on document level or sentence level sentiment polarity classification, i.e. the task of determining the opinion orientation (e.g. positive or negative) of a document or a sentence. Early solutions to this problem mainly employed knowledge based methods such as linguistic heuristics or predefined seed words [1], [2]. With the widespread availability of opinionated online documents in forums, blogs, news and review sites, data-driven-approaches had a significant upturn. Especially the increase in labeled sentiment relevant data opened the door for both supervised and unsupervised learning algorithms. With a few exceptions (e.g. sentic computing [3]), most of the active research can be seen as the application of standard text categorization techniques [4]. For a survey on the developments in sentiment analysis as well as state-of-the-art approaches, see [5]–[7].

We investigate a novel approach that has proven successful in conventional text classification tasks such as authorship attribution or topic categorization. We employ statistical data compression as a non-standard method to sentiment polarity classification. Using compression in classical text categorization was discovered independently by researchers and has been applied to a variety of problems [8]–[13]. One appealing point about compression based text classification is that it requires virtually no preprocessing and has the potential to automatically capture non-word or metaword features (i.e. features spanning more than one word).

In this paper, we evaluate the classification performance of the lossless compression algorithm Prediction by Partial Matching (PPM) as well as compression based measures using PPM-like frequency statistics over character n-grams (i.e. character sequences of a fixed length n). Comprehensive experiments on different data sets show that compression based methods are efficient, easy to apply and can compete with the accuracy of sophisticated classifiers such as support vector machines (SVM). Moreover, since the compression models are based on characters rather than on words, our methods can cope better with spelling mistakes and informal language.

1.1. Classification using compression models

Statistical compression algorithms build up models consisting of extensive statistics about the documents

processed. Hence, they can easily be applied to text classification tasks. For this, compression models for each class are created and subsequently used to evaluate the target document. In order to determine the affiliation of a target document d to some model M of class C, we are basically interested in the cross entropy [14] between the optimal probability distribution p for the source of document d and the probability distribution q given by the compression model used for evaluation.
The cross entropy $H(p, q)$ determines the average number of bits per symbol required to identify an event from a set of possibilities if a coding scheme is used based on a given probability distribution q, rather than the true distribution p. Cross entropy for two probability distributions p and q over the same probability space is defined as:

$$\mathrm{H}(p, q) := \mathrm{E}_p[-\log q] = \mathrm{H}(p) + D_{\mathrm{KL}}(p\|q)$$

where $H(p)$ is the entropy of p, and $\mathrm{D}_{\mathrm{KL}}(p\|q)$ is the Kullback-Leibler divergence of q from p [14].
The exact cross entropy is hard to compute, since it would require knowing the source distribution p. Therefore, in practice, it is mostly estimated using criteria that sufficiently correlate with the cross entropy such as the joint compression ratio or the length of resulting compression models [9], [13], [15].

2. Approach

Taking the success of compression based text classification as a motivation, we have conducted extensive experiments on three different corpora, containing documents from the movie database IMDb, the popular e-commerce site Amazon and the microblogging service Twitter.

2.1. Corpus creation and analysis

We have performed a detailed analysis of all corpora employed in our experiments and found that they vary greatly in size, complexity and language employed. Besides the average number of characters per text we evaluated the average number of words per text, the average number of words per sentence and the number of unique words employed. Although we are aware that this is not complete, we defined all words to be separated by blanks and all sentences to end with a period, a question mark or an exclamation mark in our analysis.

2.1.1. IMDb corpus. For our experiments we employed the popular `polarity dataset v2.0` extracted by Pang and Lee [16] from the Internet Movie Database (IMDb) archive of `rec.arts.movies.reviews` newsgroup. To avoid domination of the corpus by a small number of profilic reviewers, Pang et al. imposed a limit of less than 20 reviews per author per sentiment category. This resulted in a set of 2,000 reviews written by 312 authors with a total of length of 7,786,004 characters. The IMDb corpus exhibits an average text length of 3,893 characters (755 words) ranging from a minimum of 91 characters to a maximum of 14,957 characters. Moreover, with 22 words per sentence and 48,205 distinct words the language employed seems to be quite complex. This confirms the assessment of Turney [17], who found the movie review domain to be one of the most challenging for sentiment polarity classification.

2.1.2. Amazon corpus. Using a custom-build web-spider, we have extracted a high number of product reviews from the website `amazon.com`, mainly from the category electronics. From those, we randomly extracted 1,000 positive sentiment (5-star) and 1,000 negative sentiment (1-star) documents with the constraint that the whole corpus may not contain more than 20 reviews per author. This resulted in a set of 2,000 reviews with a total length of 682,124 characters written by 1,999 different authors. The Amazon corpus exhibits an average text length of 341 characters (66 words) ranging from 48 to 3,001 characters. With an average length of 13 words per sentence and 9,380 distinct words, the language employed in the Amazon corpus seems to be less complex than the one employed in the IMDb corpus.

2.1.3. Twitter corpus. In order to evaluate our methods with short and possibly noisy data, containing mainly informal language, we chose to use documents from the microblogging service Twitter. We employed the free twitter data set from sanders analytics (`http://www.sananalytics.com/lab/twitter-sentiment`), containing of 5,513 hand-classified tweets. Unfortunately, only a few tweets were labeled as positive or negative rather then irrelevant or neutral. Hence, we extracted a maximum of 500 positive labeled and 500 negative labeled documents. This resulted in a set of 500 positive labeled and 500 negative labeled documentss with a total length of 97,261 characters. The Twitter corpus exhibits an average text length of 97 characters (15 words) ranging from 9 to 140 characters. With an average length of

8 words per sentence and 3,716 distinct words, the language employed in the Twitter corpus seems to be rather simple.

2.2. Cross Entropy measures

We have conducted experiments with a variety of measures as estimates for cross entropy. These consisted of the joint compression ratio of the PPM algorithm as well as compression based measures using PPM-like character n-gram frequency statistics. Our choice to employ PPM was basically motivated by its success in other text classification tasks (e.g. [9], [13], [15]). We have, however, performed additional experiments using different compression algorithms as well as off-the-shelf programs such as RAR and ZIP that led to consistent results. Yet, due to space limitations, we are forced to postpone any details to a subsequent article.

In order to allow for an unbiased comparison with the standard approach, namely a support vector machine, we have implemented an interface, exporting data for the svmlight [18] package. Following Pang et al. [16], [19], we use unigram-presence features occurring at least four times in the training text. Please note that no parameter tuning has been performed.

Hereinafter, we denote the set of positive training documents by $T^+ = \{d_1^+, d_2^+, ..., d_o^+\}$, the set of negative training documents by $T^- = \{d_1^-, d_2^-, ..., d_p^-\}$ and the set of test documents by $T^? = \{d_1^?, d_2^?, ..., d_q^?\}$. The context (i.e. substring) i of order n (i.e. the fixed context length used for the model) for an arbitrary document $d_j = \{x_1, x_2, ..., x_m\}$ is defined as $c_{i,n} := \{x_{i-n}, x_{i-n+1}, ..., x_{i-1}\}$. Finally, $g(T, s)$ denotes the number of repetitions of a string s in a set of documents T.

2.2.1. Prediction by partial matching. Prediction by Partial Matching was first published by [20] and is used today in popular implementations such as the RAR compression program. Even though it was invented almost three decades ago, PPM remains among the best-performing lossless compression algorithms for natural text.

The basic concept behind PPM is to predict a symbol x_i by its context $c_{i,n} = \{x_{i-n}, x_{i-n+1}, ..., x_{i-1}\}$. For each order a probability distribution p_j is used to predict the successor symbol from its context and is updated after a symbol has been processed. If a symbol has not yet occurred in the model of order i, the algorithm switches to the next lower order $i-1$. If necessary, a default model of order -1 is used which always predicts a uniform distribution among all possible symbols.

Please note that although we employed the PPMd implementation from [21], any implementation (including off-the-shelf such as RAR or ZIP) may be used.

2.2.2. C-measure. An interesting criteria, namely the C-measure, was introduced by Hunnisett and Teahan [22]. It is based on the PPM compression algorithm but does not store lower order contexts.

In the subsequent definitions, let $d_j^? = \{x_1, x_2, ..., x_m\}$ be an arbitrary test document of length m and n be the order of the model. Formally, the C-measure for $d_j^?$ w.r.t. a positive (resp. a negative) model is defined as follows:

$$C^{\{+,-\}} := \sum_{i=n}^{m} a_{i,n}^{\{+,-\}} \text{ with}$$

$$a_{i,n}^{\{+,-\}} := \begin{cases} 1, & \text{if } g(T^{\{+,-\}}, c_{i,n}) > 0 \\ 0, & \text{otherwise} \end{cases}$$

Hunnisett showed that the C-measure and the joint compression ratio for PPM are highly correlated and that it slightly outperforms the compression based approach in the author attribution task [22].

2.2.3. C_k-measure. With a small modification, we were able to keep the computational properties of the C-measure and optimize it for sentiment polarity classification. We define the C_k-measure for $d_j^?$ w.r.t. a positive (resp. a negative) model as follows:

$$C_k^{\{+,-\}} := \sum_{i=n}^{m} a_{i,n,k}^{\{+,-\}} \text{ with}$$

$$a_{i,n,k}^{+} := \begin{cases} 1, & \text{if } g(T^+, c_{i,n}) > k \cdot g(T^-, c_{i,n}) \\ 0, & \text{otherwise} \end{cases}$$

$$a_{i,n,k}^{-} := \begin{cases} 1, & \text{if } g(T^-, c_{i,n}) > k \cdot g(T^+, c_{i,n}) \\ 0, & \text{otherwise} \end{cases}$$

Moreover, we define the $C_{1/\epsilon}$-measure as the special case of the C_k-measure, where ϵ is an arbitrary small number such that the $C_{1/\epsilon}$-measure counts the presence of context $c_{i,n}$ if it is exclusive to one of the training sets T^+ or T^-. Finally, we define the C-measure as the C_0-measure, where contexts are counted independently from their ratio in T^+ and T^-.

2.2.4. F_k-measure. To exploit the frequency information in the training documents, we have modified the C_k-measure further. We define the F_k-measure (frequency measure) for $d_j^?$ w.r.t. a positive (resp. a negative) model as follows:

$$F_k^{\{+,-\}} := \sum_{i=n}^{m} b_{i,n,k}^{\{+,-\}} \text{ with}$$

$$b^{+}_{i,n,k} := \begin{cases} g(T^{+}, c_{i,n}), & \text{if } g(T^{+}, c_{i,n}) > \\ & k \cdot g(T^{-}, c_{i,n}) \\ 0, & \text{otherwise} \end{cases}$$

$$b^{-}_{i,n,k} := \begin{cases} g(T^{-}, c_{i,n}), & \text{if } g(T^{-}, c_{i,n}) > \\ & k \cdot g(T^{+}, c_{i,n}) \\ 0, & \text{otherwise} \end{cases}$$

Finally, we define $F_{1/\epsilon}$ and F_0 analogous to the C_k-measure.

2.3. Computational complexity

All compression based measures presented above (i.e. C-, C_k- and F_k-measure) can be implemented efficiently. Given a set of training documents $T = T^{-} \cup T^{+}$ with $n = |T|$, our implementation requires $\mathcal{O}(n\ log(n))$ time and $\mathcal{O}(n)$ space for model creation. Moreover, using a perfect hash function in the training phase, an arbitrary test document $d^{?}_{j}$ with $|d^{?}_{j}| = m$ can be classified in $\mathcal{O}(m)$ time.

2.4. Model creation

In order to minimize the effects of an unfortunate split of training and test data we employed a k-fold cross validation method to measure the average accuracy [23]. Moreover, to ensure comparability to the results by Pang and Lee [16], we chose k to be ten. Hence, all corpora were divided into ten equal sized folds, maintaining balanced class distributions.
In each pass we divide the training set $T = \{d_1, d_2, ..., d_{o+p}\}$ (consisting of nine folds) into two disjoint subsets, T^{+} containing the positive labeled documents (e.g. 5-Star reviews) and T^{-} containing the negative labeled documents (e.g. 1-Star reviews).
Please note that we do not apply any preprocessing methods such as stemming, tagging or negation resolution. The only intermediate step performed is a data cleansing such as a removal of all HTML-tags as well as a conversion of all characters to upper case.

2.5. Evaluation

From an information-theoretic perspective we are interested in the cross entropy between the training documents of each class and documents from the test set. For compression based classification with PPM, we basically employ an approximate minimum description length (AMDL) procedure [11], [12]. Given n categories $C_1, C_2, ..., C_n$ and corresponding training sets $T_1, T_2, ..., T_n$, ADML runs the compression algorithm on each T_i to produce a compressed file $Co(T_i)$. Subsequently it appends T_i to each test document $d^{?}_{j} \in T^{?}$ and runs the compression algorithm to produce compressed files $Co(T_i \cup d^{?}_{j})$. Finally, it assigns $d^{?}_{j}$ to the category C_i that minimizes the difference in the size of the compressed files $s_i := |Co(T_i \cup d^{?}_{j})| - |Co(T_i)|$.
In our case we slightly modify the ADML-procedure to derive a measure for cross entropy and postpone the classification step. We compute a score s^{+}_{j}, s^{-}_{j} (joint compression ratio) for each test document $d^{?}_{j} \in T^{?}$ and each training set T^{+} and T^{-} as follows:

$$s^{\{+,-\}}_{j} := \frac{|Co(T^{\{+,-\}} \cup d^{?}_{j})| - |Co(T^{\{+,-\}})|}{|d^{?}_{j}|}$$

When using the C_k- or F_k-measure, ADML is not needed. Since the value returned is already a measure for cross entropy it may be used without any further modification.
Classification is usually performed by assigning the test documents to the class whose training text maximizes cross entropy. We basically follow this approach, but instead of an immediate classification, we first compute a ratio between cross entropy measures of the positive and the negative model. This allows to account for differences in the size of positive and negative models and to treat the ratio as a regression value to determine not only the polarity direction but also the polarity strength of the documents. On the downside, however, we are forced to estimate a threshold to convert the regression into a binary classification.
To obtain a normalized, one-dimensional value for cross entropy, we treat the output of the measures as well as the scores defined above equally and derive a regression ratio r_j. Let $s_{j,+}$ (resp. $s_{j,-}$) be the score or the value of an arbitrary measure for cross entropy for the test document $d^{?}_{j}$ and the positive (resp. negative) trainings set T^{+} (resp. T^{-}). The regression ratio r_j for the test document $d^{?}_{j} \in T_{?}$ is defined as:

$$r_j := \frac{(s^{+}_{j} - s^{-}_{j})}{(s^{+}_{j} + s^{-}_{j})}$$

In our training phase, we separate a small amount of documents from T^{+} (resp. T^{-}) and compute their regression ratio using the other documents as reduced trainings sets. Given a uniform distribution of positive and negative documents among this set, we found the median to be a rather simple but good choice to derive a threshold. In order to build maximal models (i.e. use the maximum amount of training data available), we subsequently add the documents to the corresponding model. Finally, in the test phase, we classify the documents into our sentiment classes (i.e. positive vs. negative) by comparing their respective regression ratios with the precomputed threshold.

Table 1. Results for the IMDb corpus

No	Method	Accuracy
(1)	PPMd	82.35%
(2)	C_0-measure	83.10%
(3)	$C_{2.5}$-measure	84.90%
(4)	$F_{2.5}$-measure	85.30%
(5)	**SVM (pres. unigram)**	**86.35%**

Table 2. Results for the Amazon corpus

No	Method	Accuracy
(1)	PPMd	86.15%
(2)	C_0-measure	85.15%
(3)	$C_{2.5}$-measure	87.95%
(4)	**$\mathbf{F_{2.5}}$-measure**	**90.55%**
(5)	SVM (pres. unigram)	86.35%

3. Results

All results reported below reflect the average accuracies of a ten-fold cross validation test. Moreover, all data contains balanced class distributions (i.e. the same number of positive and negative texts) such that the random-choice baseline result obviously would be 50%.

3.1. IMDb corpus

The average classification accuracies for the IMDb corpus are shown in lines (1)-(5) of Table 1. All compression based classifiers clearly surpass the random baseline of 50% as well as the 65.83% reported early by Turney [17]. However, they all remain below the results reported by Pang and Lee [16] and particulary below state-of-the art approaches (e.g. see [24]).
The PPMd algorithm performs worst with an average accuracy of 82.35%. It is slightly outperformed by the C_0-measure exhibiting and average accurcy of 83.10%. Several experiments with the k-measure family (i.e. the C_k-, and F_k-measure) have shown that the optimal size for k ranges between 1.5 and 2.5. Using the $C_{2.5}$-measure instead of the C_0-measure leads to an increase in the accuracy of roughly 2%. Even better results are achieved using the $F_{2.5}$-measure with an average accuracy of 85.30%. As a reference, the SVM using unigram-presence features exhibits the best classification performance with an average accuracy of 86.35%. Running a McNemar's test [25], however, suggests that there is not a statistically significant difference between average accuracies of the $F_{2.5}$-measure and the SVM. Please note that using the same method, Pang and Lee [16] reported an accuracy of 87.15%. We suppose that this is due to preprocessing, parameter tuning or to slightly different feature vectors.
Even though we have not yet analyzed all misclassifications, we took a closer look at documents that were misclassified by most of our methods. Among those, we found mostly subjective reviews that we suppose are even hard to classify by humans. Below, we quote an example that has no obvious sentiment polarity but is marked as a negative sentiment review in the corpus:

> *"He is duncan Macleod of the clan Macleod. He's been pimpin' it since he was born in the village of glennfillan in 15somethingsomething, and he continues to pimp it in modern day. He is immortal and he cannot die."*

Altogether, the results show that compression based sentiment classification is indeed possible and roughly on par with sophisticated classifiers such as SVMs.

3.2. Amazon corpus

The average classification accuracies for the Amazon corpus are shown in lines (1)-(5) of Table 2.
The C_0-measure performs worst with an average accuracy of 85.35%. With 86.15% and thus an increase of roughly 1%, PPMd performs slightly better than the C_0-measure. Our experiments with the k-measure family on the Amazon corpus confirm that the best choice for k ranges between 1.5 and 2.5. Using the $C_{2.5}$-measure instead of the C_0-measure, leads to an increase in the average accuracy of almost 3%. Finally, with an average accuracy of 90.55%, the $F_{2.5}$-measure exhibits the best classification performance. As a reference, the SVM using unigram-presence features leads to an average accuracy of only 86.35%. Please note that the average accuracy for SVM obtained on the Amazon corpus incidentially equals the one obtained on the IMDb corpus and that the results per fold differ greatly.
When analyzing the worst misclassifications, we found the Amazon corpus to contain a significant amount of spelling mistakes as well as false data. Among those is either mislabeled data (possibly due to a misunderstanding of the rating scale) or subjective texts given a star-rating that does not match a humans intitution. An example is the review quoted below. Even though it was labeled to be negative (1-star), it clearly expresses a positive sentiment polarity. Please note that we have marked spelling mistakes.

> *"Excel_ent product. I get about 15 times the life of ordinary batteries. I would reccom_end to anybody who uses their camera a lot."*

Table 3. Results for the Twitter corpus

No	Method	Accuracy
(1)	PPMd	78.80%
(2)	C_0-measure	76.20%
(3)	$C_{2.5}$-measure	83.10%
(4)	$\mathbf{F_{2.5}}$**-measure**	**84.40%**
(5)	SVM (pres. unigram)	77.80%

This review is also an example for the advantage of a character based method such as compression. Although the words that contain most sentiment relevant information, namely "Excellent" and "recommend" both suffer from spelling mistakes, the document was clearly identified to be positive. This is mainly due to the fact that sequences around the misspellings were successfully processed and led to an overall correct classification. We suppose that both spelling mistakes and false data are the main reason for the better classification performance of the k-measure family over the standard approach using SVM. A further discussion, however, will be postponed to the subsequent section.

3.3. Twitter corpus

The average classification accuracies for the Twitter corpus are shown in lines (1)-(5) of Table 3.
Analogous to the results obtained on the Amazon corpus, the C_0-measure exhibits the lowest average accuracy with 76.20%, followed by PPMd with 78.80%. Again, using a k between 1.5 and 2.5 for the C_k- and F_k-measure leads to the best classification results. Thus we recommend $k = 2.5$ as a standard parameter for the k-measure family. A drastical increase of more than 5% can be achieved using the $C_{2.5}$-measure instead of the C_0-measure. Finally, with an average accuracy of 84.40%, the $F_{2.5}$-measure exhibits the best classification performance. As a reference, the SVM using unigram-presence features leads to an average accuracy of only 77.80% and is thus more than 6% below our result.
To complete our experiments, we again analyzed the worst misclassifications. Although the Twitter corpus employs mainly informal language, it contains significantly less spelling mistakes than the Amazon corpus. This is probably due to the fact, that tweets usually consist of only one or two sentences.
Unfortunately, the analysis was not as enlightening as we hoped: The misclassifications seem to have no clear similarity and are generally easy to classify by humans. However, we found a few mislabeled documents as well as one document clearly showing the advantage of our approach.
The example quoted below was labeled positive although it clearly states a negative sentiment. We assume that in this case the annotator was labeling rather reputation than on sentiment:

hey @apple I hate my computer i need a #mack wanna send me a free one.

The second example was a short text without any separators between words. Although it is impossible for word based approaches to derive a valid sentiment polarity, our method may recognize subwords and thus correctly classified the document:

#Twitter'sMalfunctioningAgain

We suppose that the advantage of our methods is mainly due to the fact that they operate on character sequences rather than words. Moreover, adjusting k to a value between 1.5 and 2.5 seems to effectively suppress irrelevant or false features and leads to a significant increase in the average accuracy.

4. Discussion

In order to understand the comparatively good performance of our methods, especially on the Amazon and Twitter corpus, we have analyzed misclassification as well as random documents from all corpora. We have experienced a significant amount of false data such as misclassification especially in the Amazon and the Twitter corpus. Using the k-measure family, however, helps effectively eliminate such noise by learning patterns only if they appear at least k-times more often in the corresponding model. Our results show that this may lead to an improvement of more than 5%.
Besides false data, we have experienced a high number of spelling mistakes, especially in the Amazon corpus. This obviously poses a problem to most automatic classification approaches, including ours. However, since compression based methods are mostly character based, they have an advantage over word based approaches. Although they cannot successfully process misspelled words, they are able to process substrings around the misspelling. Compression based classification may thus correctly determine a document's sentiment even if most sentiment relevant words contain spelling mistakes.
To complete our experiments we performed a detailed analysis of all compression models and found that besides sentiment relevant sequences such as `'s_very_good'` (e.g. a substring from the sentence "This product is very good") the models also included sequences containing object names (e.g. product-, film-

Table 4. Means for the regression ratio per rating

Rating	$C_{2.5}$-measure	$F_{2.5}$-measure
5-star	0.1908	0.2565
4-star	0.0833	0.1141
3-star	-0.1985	-0.2683
2-star	-0.3702	-0.5034
1-star	-0.5597	-0.6882

or actor names). Given this, we were challenged to validate that our methods did actually learn the sentiment polarity and not correlations or rules of the form: A review about film XX is likely to be positive. For this, we created a test set of 5,000 documents, containing Amazon reviews of all ratings (i.e. 1-star, 2-star, ..., 5-star) in a balanced distribution. Moreover, we generated a training set containing 2,000 Amazon reviews of 1-star and 5-star ratings in a balanced distribution (that were not present in the test set). Finally, we derived a regression ratio for each document in the test set and calculated the means per class. We have found that although our training set contained 1-star and 5-star ratings only, we were basically able derive a ratio for all classes, such that a linear order (1-star $<$ 2-star $<$ ... $<$ 5-star) is clearly observable (see table 4). Please note that due to space limitations, we have to postpone further details to a subsequent article.

5. Conclusions and Future Work

In this paper we empirically and systematically evaluated the performance of the lossless compression algorithm Prediction by Partial Matching (PPM) as well as compression based measures using PPM-like character n-gram frequency statistics on the task of sentiment polarity classification. We have conducted extensive experiments on three representative corpora varying in size, complexity and language employed. We achieved a top accuracy of 90.55% using the $F_{2.5}$-measure on the Amazon corpus and outperformed support vector machines on the Twitter corpus by more than 6%.

Altogether, our experiments have shown that sentiment analysis based on compression models is possible and can compete with sophisticated classifiers. Our approach is rather simple and efficient in terms of running time and memory consumption. Moreover, it requires no preprocessing and may be performed with standard off-the-shelf compression programs such as `RAR` and `ZIP`. Although our methods are (yet) known to be inferior to state-of-the art approaches at least on the IMDb corpus, their simplicity and efficiency may make them to a promising alternative in sentiment polarity classification.

With the proposed evaluation framework, we not only obtain a binary classification, but a regression value that may effectively work in multi-class classification or regression tasks. Our experiments pinpointed that the accuracies vary greatly with the complexity, size and type of the language used in the corpora and the degree of false data. Using our k-measure family, we can cope better with spelling mistakes as well as misclassifications in the training set.

In the context of our experiments, we started evaluating the performance of the k-measures on the task of cross-domain polarity classification (i.e. learning a model using texts from one domain and testing with data from a different one). For this, we have downloaded John Blitzer's Multi-domain sentiment data set and reproduced their test setting [26]. Although our methods performed slightly better than Blitzer's baseline, they were not able to match the results after domain adaption using SCL and SCL-MI. We assume, however, that using appropriate feature selection and weighting as in [7], [24] as well as domain adaption techniques as in Blitzer's work, our methods may be able to compete. Yet, since our research in this subject is not final, we will postpone details to a subsequent article.

Another interesting branch of our research is motivated by the work of McNamee et al. [27]. They evaluated the use of character n-grams instead of words in information retrieval and found character n-gram tokenization to be highly effective especially in inflective languages. We thus started experimenting on texts in languages such as Italian, French and German. Our preliminary experiments indicate that their result is partly transferable to the domain of sentiment analysis. However, the improvement is yet not as strong as it is for information retrieval.

Finally, a deeper analysis of all corpora, our misclassifications and compression models may lead to a deeper understanding why and how our methods outperform the standard approach, especially on noisy corpora.

References

[1] W. Sack, "On the computation of point of view," in *Proceedings of AAAI*, 1994, p. 1488, student abstract.

[2] A. Huettner and P. Subasic, "Fuzzy typing for document management," in *ACL 2000 Companion Volume: Tutorial Abstracts and Demonstration Notes*, 2000, pp. 26–27.

[3] E. Cambria, *Sentic Computing Techniques, Tools, and Applications*, A. Hussain, Ed. Dordrecht: Springer Netherlands, 2012, 2012.

[4] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, 2002. [Online]. Available: http://portal.acm.org/citation.cfm?id=505283

[5] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundation and Trends in Information Retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008. [Online]. Available: http://dx.doi.org/http:/dx.doi.org/10.1561/1500000001

[6] B. Liu, *Sentiment Analysis and Opinion Mining*, ser. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2012.

[7] T. O'Keefe and I. Koprinska, "Feature Selection andWeighting Methods in Sentiment Analysis," in *Proceedings of 14th Australasian Document Computing Symposium*, December 2009.

[8] D. Benedetto, E. Caglioti, and V. Loreto, "Language trees and zipping," *Phys. Rev. Lett.*, vol. 88, no. 4, p. 048702, Jan 2002.

[9] E. Frank, C. Chui, and I. H. Witten, "Text categorization using compression models," in *Proceedings of the Conference on Data Compression*, ser. DCC '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 555–. [Online]. Available: http://portal.acm.org/citation.cfm?id=789087.789742

[10] J. Goodman, "Extended comment on language trees and zipping," *Condensed Matter Archive, Feb*, vol. 21, p. 0202383, 2002.

[11] D. V. Khmelev and W. J. Teahan, "A repetition based measure for verification of text collections and for text categorization," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, ser. SIGIR '03. New York, NY, USA: ACM, 2003, pp. 104–110. [Online]. Available: http://doi.acm.org/10.1145/860435.860456

[12] O. V. Kukushkina, A. A. Polikarpov, and D. V. Khmelev, "Using literal and grammatical statistics for authorship attribution," in *Problems of Information Transmission*, 2002, pp. 172–184.

[13] S. Goldwasser, A. C. Smith, N. Thaper, and N. Thaper, "Using compression for source based classification of text," 2001.

[14] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[15] W. J. Teahan and D. J. Harper, *Using Compression-Based Language Models for Text Categorization*. Kluwer Academic Publishers, 2003.

[16] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proceedings of the ACL*, 2004, pp. 271–278.

[17] P. Turney, "Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews," in *Proceedings of the Association for Computational Linguistics (ACL)*, 2002, pp. 417–424.

[18] T. Joachims, "Making Large-Scale SVM Learning Practical," in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. J. Burges, and A. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999.

[19] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques," in *Proc. of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002, pp. 79–86.

[20] J. G. Cleary, Ian, and I. H. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, vol. 32, pp. 396–402, 1984.

[21] D. Shkarin, "Ppm: One step to practicality." in *DCC*. IEEE Computer Society, 2002, pp. 202–211.

[22] D. S. Hunnisett and W. J. Teahan, "Context-based methods for text categorisation," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '04. New York, NY, USA: ACM, 2004, pp. 578–579. [Online]. Available: http://doi.acm.org/10.1145/1008992.1009129

[23] P. A. Devijver and J. Kittler, *Pattern recognition: A statistical approach*. Prentice Hall, 1982.

[24] G. Paltoglou and M. Thelwall, "A study of information retrieval weighting schemes for sentiment analysis," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ser. ACL '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 1386–1395. [Online]. Available: http://dl.acm.org/citation.cfm?id=1858681.1858822

[25] Q. McNemar, "Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, Jun. 1947. [Online]. Available: http://dx.doi.org/10.1007/BF02295996

[26] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification," in *In ACL*, 2007, pp. 187–205.

[27] P. McNamee, C. Nicholas, and J. Mayfield, "Addressing morphological variation in alphabetic languages," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '09. New York, NY, USA: ACM, 2009, pp. 75–82. [Online]. Available: http://doi.acm.org/10.1145/1571941.1571957