Automatic Sentiment Classification of Product Reviews Using Maximal Phrases Based Analysis

Maria Tchalakova
Textkernel BV
Nieuwendammerkade
28A-17
NL-1022 AB Amsterdam
The Netherlands
maria.tchalakova@gmail.com

Dale Gerdemann
University of Tübingen
Wilhelmstr. 19-23
72074 Tübingen
Germany
dale.gerdemann@googlemail.com

Detmar Meurers
University of Tübingen
Wilhelmstr. 19-23
72074 Tübingen
Germany
dm@sfs.uni-tuebingen.de

Abstract

In this paper we explore the use of phrases occurring maximally in text as features for sentiment classification of product reviews. The goal is to find in a statistical way representative words and phrases used typically in positive and negative reviews. The approach does not rely on predefined sentiment lexicons, and the motivation for this is that potentially every word could be considered as expressing something positive and/or negative in different situations, and that the context and the personal attitude of the opinion holder should be taken into account when determining the polarity of the phrase, instead of doing this out of particular context.

1 Introduction

As human beings we use different ways to express opinions or sentiments. The field of sentiment analysis tries to identify the ways, in which people express opinions or sentiments towards a particular target or entity. The entities could be persons, products, events, etc. With the development of the Internet technologies and robust search engines in the last decade, people nowadays have a huge amount of free information. Because of this huge amount, however, the data needs to be first effectively processed so that it could be used in a helpful way. The automatic identification of sentiments would make possible the processing of large amounts of such opinionated data.

The focus of this paper is sentiment classification at document-level, namely classification of product reviews in the categories positive polarity or negative polarity. Training and testing data for our experiments is the Multi-Domain Sentiment Dataset (Blitzer et al., 2007), which consists of product reviews of different domains, downloaded from Amazon[1]. We explore the use of phrases occurring maximally in text as features for sentiment classification of product reviews. In contrast to many related works on sentiment classification of documents, we do not use general polarity lexicons, which contain predefined positive and negative words. Very often the same word or phrase could express something positive in one situation and something negative in another. We identify words and phrases, which are typically used in positive and negative documents of some specific domains, based on the frequencies of the words and phrases in the domain-specific corpora. After that we use these phrases to classify new sentiment documents from the same type of documents, from which the phrases are extracted.

[1]http://www.amazon.com/

2 Phrase Extraction

In order to extract distinctive phrases we use the approach of Burek and Gerdemann (2009), who try to identify phrases, which are distinctive for each of the four different categories of documents in their medical data. With distinctive they mean phrases, which occur predominantly in one category of the documents or another. The algorithm extracts phrases of any length. The idea is that if a phrase is distinctive for a particular category, it does not matter how long the phrase is. The algorithm looks for repeats of phrases of any length, and could also count different types of occurrences of phrases, e.g. maximal, left-maximal, or right maximal. Considering such types of occurrences, it is possible to restrict the use of certain phrases, which might not be much distinctive and therefore might not be representative for a category. Similar to Burek and Gerdemann (2009) we experiment with using all types of occurrences of a phrase as long as the phrase occurs maximally at least one time in the text.

2.1 Distinctiveness of Phrases

Distinctive phrases are phrases, which predominantly occur in one particular type of documents (Burek and Gerdemann, 2009). The presence of such phrases in a document is a good indicator of the category (or type) of the document. The general rule, as Burek and Gerdemann (2009) point out, is that if some phrases are uniformly distributed in a set of documents with different categories, then these phrases are not distinctive for any of the categories in the collection. On the other hand, if particular phrases appear more often in one category of documents than in another, they are good representatives for the documents of this type, and consequently are said to be distinctive[2].

There are different weighting schemes, which one can use to determine the importance of a term for the semantics of a document. Burek and Gerdemann (2009) implement their own scoring function for weighting the extracted phrases. One of their reasons not to use the standard weighting function tf-idf is that the idf measure does not take into account what the category of the documents is, in which the term occurs. This is important in their case, because their data consist of four categories, which could be grouped in two main classes, namely *excellent* and *good* on the one hand, and *fair* and *poor* on the other hand. A problem when using tf-idf will appear, if there is a rare phrase, which occurs in a small number of documents, however, it clusters in documents from the two different classes, for example, in *excellent* and *fair*, or in *good* and *poor*. This will not be a good distinctive phrase for this categorization of the data. Another motivation to develop their own scoring function is to cope with the problem of *burstiness* (see section 2.2.1).

2.2 Extraction of Phrases

This section describes the algorithm of Burek and Gerdemann (2009) for extracting distinctive phrases and how we have modified and used it in the context of our work. We first show how the phrases are ranked, so that one knows what phrases are more or less distinctive than others.

2.2.1 The Scoring Algorithm

The extracted phrases are represented by occurrence vectors. These vectors have two elements - one for the number of documents with category *positive polarity*, and another for the *negative polarity*. Each element of the vector stores the number of distinct documents, in which the phrase occurs. For example, if a phrase occurs in 10 positive reviews, and 1 negative review, the occurrence vector of this phrase is <10, 1> . This shows that for the representation of the phrases we take into account the document frequency of the phrase, and not its term frequency. The motivation behind this choice is to cope with the problem of burstiness of terms. Madsen et al. (2005) explain burstiness in the following way: *The term burstiness (Church and Gale, 1995; Katz, 1996) describes the behavior of a rare word appearing many times in a single document. Because of the large number of possible words, most words do not appear in a given document. However, if a*

[2]If the number of occurrences of such phrase in the whole collection of documents is very small, however, the clustering of the phrase in some documents of a specific category, may be purely accidental. (Burek and Gerdemann, 2009)

word does appear once, it is much more likely to appear again, i.e. words appear in bursts.

We assign a score to a phrase by giving the phrase one point, if the phrase occurs in a document with positive polarity and zero points, if it occurs in a document with negative polarity.

Let us take again the occurrence vector of <10, 1> . According to the way the points are given, the vector will be assigned a score of 10 ((1 point * 10) + (0 points * 1) = 10). Is this a good score, which indicates that the phrase is distinctive for documents of category positive polarity? We can answer this question, if we randomly choose another phrase, which occurs in 11 documents, and see what the probability is, that this phrase would have a score, which is higher than or equally high to the score of the phrase in question (Burek and Gerdemann, 2009). In order to calculate this probability, the scoring method performs a simulation, in which occurrence vectors for randomly chosen phrases are created. Let us pick randomly one phrase, which hypothetically occurs in 11 reviews. Let also, have a data of 600 positive reviews and 600 negative reviews. The probability then, that the random phrase would occur in a positive or a negative review is 0.5. Based on these probabilities, the simulation process constructs random vectors for the random phrase, indicating whether the phrase occurs in a positive or in negative review. For example, if in a particular run, the simulation says that the random phrase occurs in a positive review, then we have a random vector of <1, 0>. Otherwise, <0, 1> for a negative review. The program calculates as many random vectors as the number of reviews, in which the random phrase is said to occur. In this example, the number of documents is 11. Therefore, 11 random vectors will be constructed. They may look like this: <1, 0>, <1, 0>, <0, 1> , <1, 0>, <1, 0>, <0, 1>, <0, 1>, <0, 1>, <1, 0>, <1, 0>, <0, 1>. These vectors are then summed up, and the result vector <6, 5> is the random occurrence vector for the random phrase. It tells us that the phrase, hypothetically, occurs in 6 positive and in 5 negative reviews. The score for the random phrase is now calculated in the same way as for the non-random phrases: 1 point is given for each occurrence of the phrase in a positive review, and 0 points otherwise. So, the score for this phrase is 6 (((1 point * 6) + (0 points * 5) = 6)). This process is performed a certain number of times. For the experiments presented in section 3.2, we run the simulation 10,000 times for each extracted phrase. This means that 10,000 random vectors per phrase are created.

The last step is to compare the scores of the random phrase with the score of the actual phrase, and to see how many of the 10,000 random vectors give a score higher than or equally high to the score of the actual phrase. If the number of random vectors, which give a higher than or equally high score to the actual phrase, is bigger than the number of random vectors, which give a score lower than the actual phrase, then the actual phrase is assigned a positive score, and the value of this score is the approximate number of random vectors, from which higher than or equally high scores to the actual phrase score are calculated. If the number is lower, the phrase is assigned the approximate number of random vectors, from which lower scores than the actual phrase score are calculated, and a minus sign is attached to the number, making the score negative.

2.2.2 The Phrase Extraction Algorithm

The main idea of the algorithm is that if a phrase is distinctive for a particular category, it does not matter how long the phrase is - as long as it helps for distinguishing one type of document from another, it should be extracted. In order to extract phrases in this way, the whole collection of documents is represented as one long string. Each phrase is then a substring of this string. It will be very expensive to compute statistics (i.e. tf and df) and to run the simulation process (see 2.2.1) for each substring in the text. The reason is that the amount of substrings might be huge - there are a total of N(N + 1) / 2 substrings in a corpus (Yamamoto and Church, 2001). Yamamoto and Church (2001) show how this problem can be overcome by grouping the substrings into equivalence classes and performing operations (i.e. computing statistics) on these classes instead of on the individual elements of the classes. They use for this the suffix array data structure. The number of the classes is at most 2N – 1.

2.2.3 Maximal Occurrence of a Phrase

The suffix array data structure allows for easy manipulation of the strings. The algorithm extracts phrases if they repeat in text, and if the phrases occur maximally at least once in the text. If the phrase do not occur maximally at least one time, then it may not be a good linguistic unit, which could stand on its own. Example of such words might be the different parts of certain named entities. For instance, the name *Bugs Bunny*. If *Bugs* or *Bunny* never appear apart from each other in the text, then this imply that they comprise a single entity and they should always appear together in the text. In this case it does not make sense, for example, to count only *Bugs* or only *Bunny* and calculate statistics (e.g. tf or df) for each of them. They should be grouped instead into a class.

Burek and Gerdemann (2009) mention three different types of occurrences of a phrase: left maximal, right maximal, and maximal. A left maximal occurrence of a phrase S[i,j] means that the longer phrase S[i-1,j] does not repeat in the corpus (Burek and Gerdemann, 2009). For example, in the sentences below, the phrase *recommend* is not left maximal, because it can be extended to the left with the word *highly*:

I highly *recommend* the book.
You *highly recommend* this camera.

On the other hand the phrase *highly recommend* is left maximal.

In a similar way we define the notion of right maximal occurrence of a phrase. A maximal occurrence of a phrase is when the occurrence of the phrase is both left maximal and right maximal (Burek and Gerdemann, 2009). The phrase *highly recommend* in the example sentences above is in this sense maximal.

It is not clear a priori which of these types should be taken into account for the successful realization of a given application. One could consider only the left maximal, only the right maximal, only the maximal occurrences of the phrases, or all occurrences. We experimented with *all* occurrences. Our motivation is that using all phrases would give us a big enough number of distinctive phrases and we will most probably not have a problem with data sparseness.

3 Sentiment Classification of Product Reviews

For the experiments presented below we used a supervised machine learning approach, and different sets of features. Reviews from two domains, *books* and *cameras & photos*, are used as training and testing data.

3.1 Choosing Distinctive Phrases for Classification

Once the phrases with which we would like to represent the documents are extracted, we need to consider two things in the very beginning. On the one hand, the phrases should be as much distinctive as possible. On the other hand, even though a phrase might occur predominantly in negative reviews, it occurs very often also in positive reviews (once or at least several times), and vice versa. Should we consider such phrases? If yes, what would be the least acceptable number of occurrences of the phrases in the opposite type of reviews? We might choose as distinctive phrases those which occur only in positive or only in negative reviews, however, these phrases will be very few, and we might have the problem of data sparseness. On the other hand, using all extracted phrases might bring a lot of noise, because many of the phrases will not be very good characteristics of the data. We experimented with several different subsets of the set of all extracted phrases.

In order to decide what subsets of extracted phrases to use, we analyzed the set of all extracted phrases paying attention to their vectors and the scores, trying to find a trade-off between the two mentioned considerations above.

3.2 Experiments

SVM is used as a machine learning algorithm for the experiments (the implementation of the SVM package LibSVM[3] in GATE[4]).

[3]Libsvm: a library for support vector machines, 2001. software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.
[4]http://gate.ac.uk/

For each experiment we first divide the reviews of each domain into training and testing data with ratio two to one. From this training data we extract the distinctive phrases, which are later used as features to the learning algorithm. As evaluation method we apply the k-fold cross validation test, with k=10. For all experiments we used the default tf-idf weight for the n-grams. For each domain we conduct five different experiments, each time using different subsets of distinctive phrases. All experiments were performed with GATE.

For each domain the training data from which the phrases are extracted consists of about 665 negative and 665 positive reviews. The testing data consists of 333 negative and 333 positive reviews.

It is interesting to notice that although the results of the experiments are different, they are very close to each other, regardless of the big difference in the number of phrases used as features. Therefore, we decided to experiment with all extracted phrases. It turned out that the results of that experiment are the best. This would imply that the bigger number of phrases is helpful and it compensates for the use of phrases that are not much distinctive.

The results of all experiments for domain *books* are summarized in Table 1. The best achieved results of 81% precision, recall, and F-measure are given in bold. The rightmost column gives the number of negative (n.) and positive (p.) phrases used in each experiment.

Experiment	Reviews	P	R	F-m	Phrases used
Exp1	Negative Positive Overall	0.77 0.80 0.78	0.80 0.77 0.78	0.79 0.78 0.78	1685 n. 1116 p.
Exp2	Negative Positive Overall	0.75 0.80 0.77	0.80 0.74 0.77	0.77 0.76 0.77	924 n. 568 p.
Exp3	Negative Positive Overall	0.76 0.78 0.77	0.78 0.76 0.77	0.77 0.77 0.77	349 n. 178 p.
Exp4	Negative Positive Overall	0.77 0.79 0.78	0.79 0.77 0.78	0.78 0.78 0.78	10552 n. 9084 p.
Exp5	Negative Positive **Overall**	0.80 0.81 **0.81**	0.81 0.80 **0.81**	0.80 0.80 **0.81**	All: 24107 n. 21149 p.

Table 1: Domain *books*

Table 2 summarizes the results for domain *camera&photos*, showing the best results of 86% precision, recall, and F-measure in bold.

Similar to the experiments with reviews of domain *books*, the results for *camera&photos* in all five experiments are very close. Again the best results are obtained when all extracted distinctive phrases are considered.

Experiment	Reviews	P	R	F-m	Phrases used
Exp1	Negative Positive Overall	0.85 0.83 0.84	0.83 0.85 0.84	0.84 0.84 0.84	1746n. 1883 p.
Exp2	Negative Positive Overall	0.84 0.81 0.83	0.81 0.85 0.83	0.82 0.83 0.83	1013n. 1053 p.
Exp3	Negative Positive Overall	0.86 0.83 0.85	0.83 0.87 0.85	0.85 0.85 0.85	384 n. 432 p.
Exp4	Negative Positive Overall	0.85 0.83 0.84	0.83 0.86 0.84	0.84 0.84 0.84	7572 n. 9821 p.
Exp5	Negative Positive **Overall**	0.86 0.85 **0.86**	0.85 0.87 **0.86**	0.86 0.86 **0.86**	All: 16378 n. 17951 p.

Table 2: Domain *camera&photos.*

In order to evaluate how well the results of the experiments are we performed several more experiments, in which the texts were represented with unigrams (1-grams) and bigrams (2-grams). Pang and Lee (2008) note that: *whether higher-order n-grams are useful features appears to be a matter of some debate. For example, Pang et al. (2002) report that unigrams outperform bigrams when classifying movie reviews by sentiment polarity, but Dave et al. (2003) find that in some settings, bigrams and trigrams yield better product-review polarity classification*. Bekkerman and Allan (2004) review the results of different experiments on text categorization in which n-gram approaches were used, and conclude that the use of bigrams for the representation of texts does not show general improvement (Burek and Gerdemann, 2009). It seems intuitive that when bigrams are used, we would have a better representation of the texts, because we would know what words combine with what other words in the texts. However, there is a data sparseness problem.

It seems interesting to compare the results obtained by representing the texts as unigrams, bigrams, and distinctive (maximally occurring) phrases, because the model based on phrases might use both unigrams and bigrams, and it allows also any other higher n-grams, that is, more context (and semantics) of the text is preserved.

Tables 3 and 4 present the results of the experiments using bag-of-tokens (1-gram) models, while Tables 5 and 6 present the experiments with the 2-gram models. GATE was used as a working environment, and SVM as learning algorithm.

Reviews	**Precision**	**Recall**	**F-measure**
Negative	0.77	0.82	0.79
Positive	0.82	0.75	0.78
Overall	0.79	0.79	0.79

Table 3: Domain *books*, 1-gram.

Reviews	**Precision**	**Recall**	**F-measure**
Negative	0.86	0.84	0.85
Positive	0.84	0.86	0.85
Overall	0.85	0.85	0.85

Table 4: Domain *camera&photos,* 1-gram.

Reviews	**Precision**	**Recall**	**F-measure**
Negative	0.72	0.80	0.75
Positive	0.78	0.69	0.73
Overall	0.75	0.75	0.75

Table 5: Domain *books,* 2-gram.

Reviews	**Precision**	**Recall**	**F-measure**
Negative	0.84	0.83	0.83
Positive	0.83	0.84	0.83
Overall	0.83	0.83	0.83

Table 6: Domain *camera&photos,* 2-gram.

Features	Precision	Recall	F-measure
All phrases	**0.81**	**0.81**	**0.81**
1-gram	0.79	0.79	0.79
2-gram	0.75	0.75	0.75

Table 7: Comparison, Domain *books.*

Features	Precision	Recall	F-measure
All phrases	**0.86**	**0.86**	**0.86**
1-gram	0.85	0.85	0.85
2-gram	0.83	0.83	0.83

Table 8: Comparison, Domain *camera&photos.*

Tables 7 and 8 summarize the overall results using 1-gram and 2-gram models and a model based on distinctive phrases for the representation of the texts. For both domains the best results are achieved with the model based on phrases (all phrases). For the domain *books* the overall precision, recall and F-measure results achieved with that model (81%) are 2% higher than the results obtained using the 1-gram model, and 6% higher than the results obtained using the 2-gram model. For domain *cameras & photos*, an improvement of 1% and 3% is achieved with the phrase model in comparison with the 1-gram and 2-gram models, respectively.

4 Related Work

Close to our work seems to be Funk et al. (2008). They classify product and company reviews into one of the 1-star to 5-star categories. The features to the learning algorithm (also SVM) are simple linguistic features of single tokens. They report best results with the combinations *root & orthography*, and *only root*. Another interesting related work is that of Turney (2002). He uses an unsupervised learning algorithm to classify a review as *recommended* or *not recommended*. The algorithm extracts phrases from a given review, and determines their pointwise mutual information with the words *excellent* and *poor*. Turney (2002) points out that the contexual information is very often necessary for the correct determination of the sentiment polarity of a certain word.

5 Conclusion

This paper presented different experiments on classifying product reviews of domains *books* and *cameras & photos* under the categories *positive polarity* and *negative polarity* using distinctive (maximally occurring) phrases as features. For both domains best results were achieved with all extracted distinctive phrases as features. This approach outperforms slightly the 1-gram and 2-gram experiments on this data and shows that the use of phrases occurring maximally in text could be successfully applied in the classification of sentiment data and that it is worth experimenting with classifying sentiment data without necessarily relying on general predefined sentiment lexicons.

References

Ron Bekkerman and James Allan. 2004. Using bigrams in text categorization. *Technical Report IR-408*, Center of Intelligent Information Retrieval, UMass Amherst.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 440-447, Prague, Czech Republic. Association for Computational Linguistics.

Gaston Burek and Dale Gerdemann. 2009. Maximal phrases based analysis for prototyping online discussion forums postings. In *Proceedings of the workshop on Adaptation of Language Resources and Technologies to New Domains (AdaptLRTtoND),* Borovets, Bulgaria.

Kenneth W. Church and William A. Gale. 1995. Poisson mixtures. *Natural Language Engineering*, 1:163-190.

Kushal Dave, Steve Lawrence and David M. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of WWW,* pp. 519-528.

Adam Funk, Yaoyong Li, Horacio Saggion, Kalina Bontcheva, and Christian Leibold. 2008. Opinion analysis for business intelligence applications. In *Proceedings of First International Workshop on Ontology-supported Business Intelligence (OBI2008)* at the *7th International Semantic Web Conference (ISWC)*, Karlsruhe, Germany.

Slava M. Katz. 1996. Distribution of content words and phrases in text and language modelling. *Natural Language Engineering*, 2(1):15-59.

Rasmus E. Madsen, David Kauchak, and Charles Elkan. 2005. Modeling word burstiness sing the dirichlet distribution. In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 545-552.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, Vol. 2, Nos. 1-2 (2008) 1-135.

Bo Pang, Lillian Lee., and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 79-86.

Peter D. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 417-424.

Mikio Yamamoto and Kenneth W. Church. 2001. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. In *Computational Linguistics*, 27(1):1-30.