Identifying Explicit Features for Sentiment Analysis in Consumer Reviews

Nienke de Boer, Marijtje van Leeuwen, Ruud van Luijk,
Kim Schouten, Flavius Frasincar, and Damir Vandic

Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands
{nien_de_boer,marijtje93}@hotmail.com
ruudvanluijk91@gmail.com
{schouten,frasincar,vandic}@ese.eur.nl

Abstract. With the number of reviews growing every day, it has become more important for both consumers and producers to gather the information that these reviews contain in an effective way. For this, a well performing feature extraction method is needed. In this paper we focus on detecting explicit features. For this purpose, we use grammatical relations between words in combination with baseline statistics of words as found in the review text. Compared to three investigated existing methods for explicit feature detection, our method significantly improves the F_1-measure on three publicly available data sets.

1 Introduction

In the last decade, the World Wide Web has changed enormously. E-commerce is expanding at a rapid pace as more and more people have access to the Internet nowadays. Because of this, the amount of reviews given on products and services is also increasing. Some products and services have hundreds of reviews, scattered over many websites. These reviews are a valuable source of information for both consumers [4] and producers [15]. Since the amount of reviews is large, it is hard, if not impossible, to read all of them and to keep track of all expressed opinions on the different features of the product or service. Selecting a few reviews to read may give a false impression of the product or service, so it is clear that more advanced and automated methods for processing and summarizing reviews are needed [10].

Reviews contain characteristics of products or services, so-called features. The literature reports several works on extracting features from texts. Some of these methods concentrate on finding explicit features while others focus on extracting implicit features. In this paper, we propose a new method to extract explicit features on which reviewers have expressed their opinions, by employing and adapting various techniques that proved to be useful in previous works. Where existing approaches use either frequency counts or grammatical relations, we propose to use both. First, we use grammatical relations between words to

find feature candidates. Then, we check whether these feature candidates occur more often than expected with respect to a general corpus, before annotating them as actual features. We hypothesize that these two techniques complement each other in such a way that combining them will yield higher precision and recall.

The paper is organized as follows. We start by discussing some of the related work in Sect. 2. In Sect. 3, we present the proposed method. In Sect. 4, the data that is used to test the method is discussed and then in Sect. 5 the evaluation results are presented. Last, our conclusions and suggestions for further research are given in Sect. 6.

2 Related work

In this section we discuss some of the related work that has been done on finding features in customer reviews. Due to space limitations, we investigate only three existing methods which are representative for their classes: a frequency-based approach, a statistical approach, and a (grammatical) relation-based approach.

One of the most well known methods to find explicit features is proposed by Hu and Liu in [7]. This method first extracts the features that are frequently mentioned in the review corpus. Since it is assumed here that explicit features are most likely to be nouns or noun phrases, these are extracted from all sentences and are included in a transaction file. In order to find features that people are most interested in, Association Rule Mining [1] is used to find all frequent item sets. In this context, an item set is a set of words or phrases that occur together. When the final list of frequent features is known, the method extracts all opinion words that are nearby the frequent features. To that end, it exploits the fact that opinion words are most likely to be adjectives. The found opinion words are used to find infrequent features, based on the idea that people often use the same opinion words for both frequent and infrequent features. If a sentence does not contain a frequent feature but does contain one or more opinion words, the proposed method extracts the noun nearest to that opinion word and this noun is stored in the feature set as an infrequent feature. A disadvantage of this method is that nouns or noun phrases that are more used in general will also be annotated as feature, favoring false positives. In the next paragraph we present a method which alleviates this issue.

In the paper by Scaffidi et al. [17], a method is proposed that uses baseline statistics of words in English and probability-based heuristics to identify features. The main idea is that nouns that occur more frequently in the review corpus than in a random section of English text are more likely to be features. Since reviewers focus on a specific topic, the relevant words for that topic will occur more in the review than in a normal English text. The probability that a certain noun or noun phrase (a maximal length of 2 is used for a noun phrase) occurs in a normal text as much as it does in the review is calculated. If this probability is small, it is more probable that the noun or noun phrase is a feature. This method has a high precision but the recall is low. For this method, it is important that the

English text that is used for determining the baseline is in the same language and roughly the same style as the reviews. Using baseline statistics thus addresses the discussed disadvantage of the previous method.

A major shortcoming, shared by both [7] and [17] is that only the number of times a word occurs in all the reviews taken together matters. The methods do not take into account the grammatical structures that are within the review sentences, which could be useful to find infrequent features that appear seldom in reviews as well as in general text. In [6], Hai et al. propose a method that focuses on implicit feature identification in Chinese written reviews. However, since explicit features are used to find implicit ones, an algorithm for finding explicit features is also presented. In this method, all nouns and noun phrases that are in certain grammatical dependency relations are added as explicit features. The relations that are used here are the nominal subject relation, the root relation, the direct object relation, and object of a preposition relation. In this method, it is assumed that the sentences which contain an explicit feature are already known. The method first checks whether the sentence contains an explicit feature before it investigates the dependency relations. This makes feature extraction relatively easy and it allows for a higher recall and precision. However, in general it is not known in advance which sentences contain features and which sentences do not, so that precision is generally low when using this method. Also, this method does not use information about how often nouns and noun phrases occur at all, while this certainly is valuable information.

3 Method

In this section we propose a new method for finding explicit features by addressing the shortcomings identified in the previous work. For this purpose, we reuse and adapt techniques described in [6] and [17]. Since in the first method, precision is low but recall is relatively high, and in the second the recall is rather low but precision is high, an ensemble method that combines these two methods seems a logical next step. Therefore, we use the dependency relations of [6] to obtain a high recall and the baseline statistics of [17] to obtain a high precision.

First, the proposed method analyzes dependency relations in the review sentences to find feature candidates. This can be seen as the first step in [6]. To find feature candidates, we check the grammatical structures in each of the review sentences. If a word is in one of the used dependency relations, we first check if it is a noun or part of a noun group, before we add it to the list of feature candidates. This is because earlier research has shown that features which are explicitly mentioned are most likely to be nouns [14]. We use the Stanford Parser [11] to find the grammatical structures. Besides the dependencies mentioned by Hai et al., we also use some additional dependencies. First the dependencies which are used in [6] are explained. One of these dependencies is the 'direct object' (dobj) dependency. We explain this dependency by the following example sentence:

"He tried to clear a table for six."

If a word is the object of a verb, for instance the word "table" in the above sentence, we add this word to the feature group. Also, we add the dependent word of a 'preposition-object' (pobj) relationship. This relationship is often combined with the 'prepositional dependency' (prep). For instance, take the next sentence, in which one of the features is "brunch":

"Great for groups, great for a date, great for early brunch or a nightcap."

The dependencies prep(great, for) and pobj(for, brunch) are combined into prep_for(great, brunch). Therefore, we use both the dependencies 'pobj' and all the combined dependencies such as prep_for. Two other dependencies that are used in [6] are the 'root' and the 'nominal subject' relationships. We explain these dependencies with the help of another sentence example:

"The food is very good too but for the most part, it's just regular food."

The dependencies between the words of this sentence are shown in Fig. 1. Additional information regarding the various dependencies mentioned in this figure can be found in [12]. We add nouns to the feature group which are the 'root' (root) of the sentence, the word the whole sentence relates to. In the example, the root is "good", which is not a noun, so in this case we do not add this word to the feature group. We also add a noun to the feature group if it is the dependent word in the 'nominal subject' (nsubj) relationship. In our example sentence, this means we add the second word of the sentence, the word "food", to the feature group.

The dependencies we discussed so far are the original dependencies that were used in the method which was proposed in [6], but that method is based on Chinese written reviews. Since the data sets we use contain English reviews, it is useful to add extra dependencies. One of the extra dependencies we add is the dependency 'conjunction' (conj). This is a relation between two words which are connected by words like "and" or "or". In the above example sentence, the words "good" and "food" are connected by the word "but". Because of this, "food" is added to the list of feature candidates. Furthermore, we examine the 'noun compound modifier' (nn). This is a dependency between two nouns, in which first noun modifies the meaning of the second noun. The next sentence shows an example of this dependency:

"A wonderful jazz brunch with great live jazz."

Here, the noun "jazz" modifies the meaning of the noun "brunch", so "brunch" is added to the list of feature candidates. The last dependency we use is the 'appositional modifier' (appos): a noun that serves to define or modify the meaning of another noun, which is located at the left of the first noun. We explain this with the help of another sentence:

"A gentleman, maybe the manager, came to our table, and without so much as a smile or greeting asked for our order."

Here the noun “manager” defines the meaning of the noun at the left of it, the noun “gentleman”. In this case, “manager” is added to the list of feature candidates. The pseudocode of the total process of retrieving the feature candidates is shown in Algorithm 1.

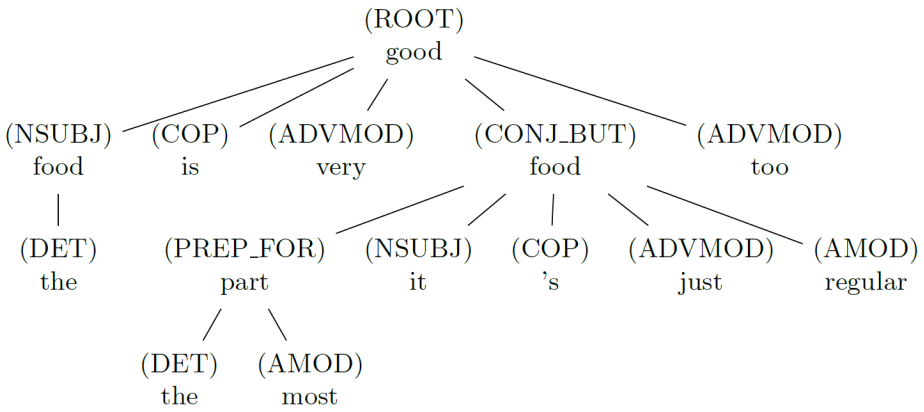


Fig. 1: Grammatical dependency relations

Now that we have a list with feature candidates based on grammatical structures, we want to improve precision. The above method is likely to find a lot of features, but it will also find a lot of non-features. If a possible feature is mentioned in many reviews, it is more likely to be an actual feature. At this point, it is important to note that some words are just more common to use in English text than others. For example, the word “lens” might not be used a lot in a normal English text, but when it is used in a review about a digital camera, it might be mentioned a lot more. Therefore, it could very well be a feature about which the writer is expressing his opinion. For this reason, we check for each of the feature candidates that we found using Algorithm 1 whether it occurs more than would be expected based on a general English text. This is the main idea that is used in [17] as well. When the word is in a list of stop words, we do not add it as a possible feature. Also, if a found noun is more likely to be used as a non-noun, it is probably not a feature, so those will not be added as features as well. For example, the word “count” can be used as a noun, but it is more probable that it is used as a verb. We check this aspect for all nouns using the frequency counts that are provided by WordNet [13]. However, for some words, the frequency counts are not given in WordNet or the word is not in the WordNet database. In such cases, we assume that the word is more likely to be used as a noun.

To explain the idea of using baseline probabilities in more detail, we start by focusing on single noun features only. To check whether the noun occurs more often than expected, one needs to check whether the number of times it occurs

in the reviews is larger than the expected amount in a generic English text of the same length. If it does occur more often, the probability that the noun occurs exactly that often is calculated. In this case, we do not look at the word itself, but we take the dictionary form of the word, the lemma, and count how many times the lemma occurs. We denote by n_x the number of times that lemma x occurs in the review corpus. Thus we calculate the probability $P(n_x)$ that lemma x occurs n_x times in a general English text with N words, where N is the total number of words in the review corpus.

To calculate this probability, the probability p_x that the lemma of a randomly selected word in a generic English text equals x needs to be known first. We use the word frequency list on conversational English from Leech et al. [9] for this. With the provided word counts, we calculate the baseline probability by dividing the count by the total amount of words in the general English text. When a review word does not occur in the general English text, we assign the average probability of words in the general English text to this word.

We can now estimate the probability that lemma x occurs n_x times in the general English text with a Poisson distribution, as we want to determine the probability of a certain count in an interval, in our case, the entire length of the review corpus [3]. However, to avoid numerical underflow, it is convenient to take the logarithm of the distribution. For this, we need to apply Stirling's approximation to estimate $\ln(n_x!)$ [16]. As a result, Eq. 1 is used to calculate the probabilities. The last term, namely $ln(\sqrt{2\pi})$ is just a constant, and thus not necessary when comparing probabilities.

$$ln(P(n_x)) \simeq (n_x - p_x N) - n_x ln(\frac{n_x}{p_x N}) - \frac{ln(n_x)}{2} - ln(\sqrt{2\pi}) \tag{1}$$

We can use a similar approach for calculating probabilities of noun phrases. In order to do this, we do need to make some assumptions. First, we assume that the occurrence of lemma x in position i is independent of whether lemma x occurs in any other position j in a sentence. We also assume that the occurrence of lemma x in position i is independent of i. Although these assumptions are generally not true, it helps to simplify the problem while they have no serious consequences for the results [19]. Because of these assumptions, the probability p_b that the bi-gram b occurs in generic English is simply p_x times p_y. In this context, a bi-gram is a noun phrase with two nouns, the probability of occurrence of the first word in the bi-gram is p_x and the probability of occurrence of the second one is p_y. We can now use Eq. 1 where n_x represents the number of occurrences of the bi-gram. To obtain the probability of a tri-gram, one must first multiply the probabilities of the single words to get p_x and use Eq. 1 again after that.

Now that the probabilities of a noun or bi-gram occurring with a certain frequency are known, we select the ones with the smallest probabilities to be features. It is important to determine how many features are added and for this a threshold value is required. If the probability of a noun or noun phrase is smaller than the threshold value, the noun or noun phrase is added to the list of features. The threshold value may differ among different data sets so this value should be trained per data set. In [17], there was only one threshold

value for single-nouns and n-grams, but there might be different optimal values for single-noun features, bi-grams, tri-grams, et cetera, so the values should be allowed to be different. This gives rise to a new problem, namely that there are not many feature candidates that consist of three or more words (as will be discussed in Sect. 4). Because of this, it is hard to train the threshold value for tri-grams and higher n-grams. Therefore, it is better to apply the approach of baseline probabilities comparison only on single-noun features and bi-grams. Algorithm 2 shows how the probability of a noun or bi-gram occurring a given number of times is computed.

In order to determine the best threshold value, we use training data only. As mentioned before, the threshold values of single-nouns and bi-grams are allowed to be different. We choose the combination of threshold values for which the F_1-value, the values that we use for evaluation, is the highest. For this, we cannot use a gradient ascent method, since the function of the F_1-values is not concave (there are some local maxima points). Therefore, a linear search method is used instead of the gradient ascent method. First, the probabilities computed with Algorithm 2 are sorted. These probabilities are the possible threshold values. We iterate over the possible threshold values by selecting the percentage of single nouns or bi-grams that we want to add as features. The step size we use is 0.01.

Now that we have all probabilities and threshold values, we iterate over each possible feature found with Algorithm 1 and check whether the corresponding probability is smaller than the threshold value. If this is the case, we add the feature to the final list of features. If the feature is a tri-gram or a n-gram of higher degree, we add this feature to the final list without considering the probabilities.

Algorithm 1 Generating a group of possible explicit features

```
Input: review sentences in the corpus
Output: a list of possible explicit features F
for each sentence s ∈ corpus do
    for each word or wordgroup w ∈ s do
        if w is in grammatical relationship of specified types then
            if each POS tag of w is noun then
                add w to F
            end if
        end if
    end for
end for
```

4 Data analysis

In this section a brief overview of the used data sets is presented. Since the performance of different algorithms depends a lot on the used data set, three

Algorithm 2 Calculating probabilities of single nouns and bi-grams

Input: D : a list of all words in the corpus. N : a count of all words in the corpus. O : a list of all unique nouns in the corpus. B : a list of all bi-grams in the corpus. p_x : a list of probabilities how often a word appears in an English text.
Output: P_S : a list with the logarithm of the probability that a single noun occurs exactly n_x times in the review corpus with x denoting the lemma of the noun. P_B : a list with the logarithm of the probability that a bi-gram occurs exactly n_b times in the review corpus with b denoting the bi-gram.

for each word $x \in D$ **do**
  if x is a noun **then**
    $count(x) + +$
    if $nextWord(x)$ is a noun **then**
      $b = concat(x, nextWord(x))$
      $countB(b) + +$
    end if
  end if
end for
for each word $x \in O$ **do**
  find baseline probability p_x
  calculate the probability that the word occurs n_x times, with $n_x = count(x)$:
  $P_S(x) = (n_x - p_x * N) - n_x * ln(\frac{n_x}{p_x * N}) - \frac{ln(n_x)}{2} - ln(\sqrt{2\pi})$
end for
for each bi-gram $b \in B$ **do**
  calculate $p_b = p_x \times p_{nextWord(x)}$
  calculate the probability that the bi-gram occurs n_b times, with $n_b = countB(b)$:
  $P_B(b) = (n_b - p_b * N) - n_b * ln(\frac{n_b}{p_b * N}) - \frac{ln(n_b)}{2} - ln(\sqrt{2\pi})$
end for

different data sets are used to train and evaluate the algorithms. The first two data sets are from the SemEval competition [2]. The first is a set containing reviews about restaurants [5]. The second data set contains reviews about laptops. The third data set contains a collection of reviews of a set of products [8]. These products include a camera, a printer, a DVD-player, a phone and an mp3. It turns out that the differences between these data sets are substantial. The characteristics of these sets will now be presented in more detail.

4.1 Restaurant data set

The restaurant data set contains 3041 sentences. In these sentences, one can find a total of 1096 unique explicit features. As can be seen in Fig. 2a, about one third of the sentences does not contain a feature, which means that two thirds of the sentences contain at least one feature. Since most sentences contain a feature, it is relatively easy to find features. The features in this data set are mostly single word features (75.44%) and bi-grams (16.84%), but 7.72% of the features consist of 3 or more words. This is illustrated in Fig. 2b. The largest feature is a composite of nineteen words, but this only appears one single time.

Most algorithms extract single-noun features and bi-gram features rather well. The longer features are harder to find as the difficulty of finding a feature increases with the amount of words it consists of. This results in most algorithms performing well on this data set.

4.2 Laptop data set

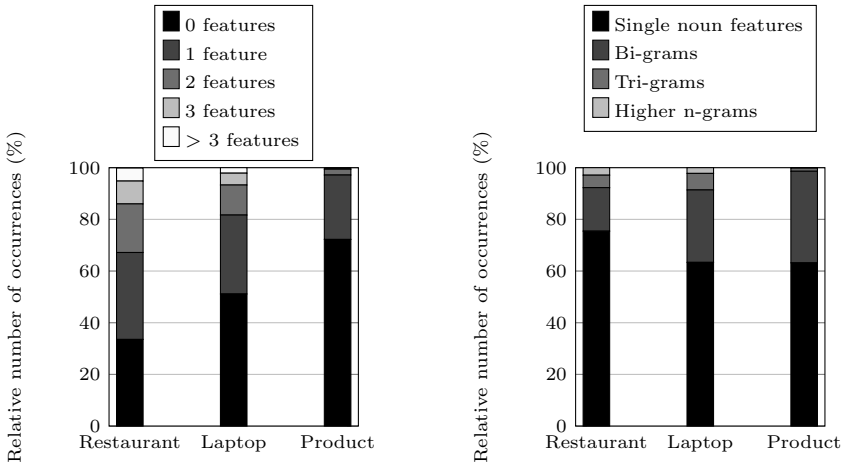
The laptop data set has only four more sentences than the restaurant data set, but there are 231 less unique features to be found. This means that the data set contains only 865 unique features. As can be seen in Fig. 2a, more than half of the sentences does not contain any feature. This is about 20%-point more than in the restaurant data set. The distribution of the sentences that contain one or more features is similar as in the restaurant set. The ratio between sentences with one feature and sentences with two, three or more features is about the same. Fig. 2b shows that there are relatively less features that consist of one single word, but there are more bi-grams and tri-grams in this data set in comparison with the restaurant data set.

Although some of the characteristics of this data set seem to be quite similar to the characteristics of the restaurant data set, it is harder to get a good performance on this data set. This is mainly caused by the fact that there are more sentences that do not contain any features. The fact that there are more features that consist of more than one word makes it also harder to get a good performance.

4.3 Product data set

The product data set has very different characteristics in comparison with the restaurant data set and the laptop data set. There are 904 more sentences in this data set than in the restaurant data set, which gives a total of 3945 review sentences. However, there are less features to be found. The number of unique features in all these sentences is only 231. There are 2850 sentences without any feature and only 1095 of the sentences contain one or more features. A visual representation of this can be found in Fig. 2a. In this data set, most features contain only one or two words. The corresponding percentages are respectively 63.20% and 35.43%. The largest feature in this data set has a size of four words. In Fig. 2b the differences between the data sets are illustrated. While the product data set has more bi-grams, there are almost no n-grams with n higher than two. Thus an algorithm that extracts bi-grams rather well is needed here. Since the amount of features is small and the amount of sentences without features is rather large, it is more difficult to get a good performance on this data set. A lower F_1 can thus be expected.

Because of the different characteristics, the use of these three sets is ideal for developing and testing an algorithm. If an algorithm performs very well on only one of the three sets, it might not be very useful in general. Performing good on all three data sets is a good indicator that it might perform well on other data sets as well.



(a) The distribution of number of features per sentence

(b) The distribution of n-grams in the data sets

Fig. 2: Characteristics of the data sets

5 Evaluation

For the evaluation of the proposed method, a 10-fold cross-validation is performed. For this, the available data is randomly divided in ten equally sized groups. Nine groups are used to train the threshold values. The sentences that are in the tenth group are used as test data. We repeat this approach ten times, with each time another group as test data. In the end, all ten groups are used as test data once, and nine times as training data. For the evaluation of the method, and to determine the best threshold values, we used the F_1-value, which is the harmonic mean of precision and recall.

Since the proposed method uses some of the ideas that were expressed in the papers [17] and [6], the performance of the proposed method is compared with the performance of these methods. Furthermore, we use the method proposed in [7] to compare the results of the new method with. The results of the earlier proposed methods on the different data sets can be found in Table 1. To show the effects of the various components of the proposed method, a stepwise evaluation is performed. The first step is to improve the two original methods of [17] and [6].

In [17], a single threshold for both single-noun features and bi-grams is used. As argued before, it can be better to allow different threshold values for single noun features and bi-gram features respectively. The performance of the method as proposed in the paper, having only one threshold value, and the performance of the extended method that allows for threshold values to be different are shown in Table 2. It can be seen that for the laptop data set and for the product data set, the F_1-score improves by more than one percentage point when the threshold values are allowed to be different. For the laptop data set, the increase

Table 1: The performance of the methods proposed in the papers by Hu and Liu [7], Scaffidi et al. [17] and Hai et al. [6]

	Restaurant data set			**Laptop data set**			**Product data set**		
Method	Hu	Scaffidi	Hai	Hu	Scaffidi	Hai	Hu	Scaffidi	Hai
Precision	0.371	0.432	0.384	0.137	0.188	0.200	0.056	0.126	0.060
Recall	0.627	0.575	0.715	0.405	0.451	0.591	0.480	0.356	0.417
F_1	0.467	0.493	0.500	0.204	0.266	0.299	0.100	0.186	0.105

is caused by an increase in precision. For the product data set, both precision and recall have increased. Results for the restaurant data set are slightly different. Although recall has increased in this case, the F_1-score is the same for both methods. Combining the results on the different data sets, we conclude that it is better to allow the threshold values to be different, since the F_1-scores of that method are the same or higher than the scores of the method that uses a single threshold.

Table 2: Performance of the method proposed by Scaffidi et al. [17] with one threshold value and the performance of the extended method that uses two threshold values on the different data sets.

	Restaurant data set		**Laptop data set**		**Product data set**	
Threshold values	One	Two	One	Two	One	Two
Precision	0.432	0.412	0.188	0.226	0.126	0.133
Recall	0.575	0.614	0.451	0.380	0.356	0.380
F_1	0.493	0.493	0.266	0.283	0.186	0.197
Diff. in F_1	**+0.000**		**+0.017**		**+0.011**	

In [6], the used grammatical relations are the nominal-subject, the root, the direct object, and the object of a preposition relations. This method was designed for Chinese written reviews and for English reviews it is better to use more types of relations. Adding the relation types 'conjunction', 'appositional modifier' and 'noun compound modifier' boosts the recall, but the precision declines. Since we want to get a high recall using the grammatical dependencies and we expect precision to grow using the baseline probabilities, we use the relation types 'conjunction', 'appositional modifier' and 'noun compound modifier' in addition to the relation types that were used in [6]. In Table 3 the performance of the method proposed in [6] with the original dependencies and the performance of the same method but with the proposed dependencies are shown.

Table 3: Performance of the method proposed by Hai et al. [6] with the original dependencies and the performance of the method with the proposed dependencies on the different data sets.

	Restaurant data set		**Laptop data set**		**Product data set**	
Dependencies	Original	Proposed	Original	Proposed	Original	Proposed
Precision	0.384	0.380	0.200	0.197	0.060	0.054
Recall	0.715	0.771	0.591	0.646	0.417	0.477
F_1	0.500	0.509	0.299	0.303	0.105	0.098
Diff. in Recall	**+0.056**		**+0.055**		**+0.060**	

Now, the performance of the proposed method is discussed. We evaluate the performance of the method using only the grammatical structures used in [6] and a single threshold value first. The results are shown in Table 4, with 'original' referring to the fact that only the grammatical structures of [6] are used. As discussed in the previous part of this section, adding more grammatical structures may improve performance and allowing for different threshold values causes an increase in performance as well. Therefore, we also tested the proposed method with the additional grammatical structures and with different threshold values. The results of the method with proposed grammatical structures are shown in Table 4 under the heading 'proposed'. The bottom row of each subtable shows the difference between the F_1-value of the used method in that column and the F_1-value of the proposed method that uses only one threshold value and the original dependencies.

In Table 4 it is shown that for all data sets, the F_1-measure of the method in which more grammatical relations and different threshold values are used is the highest. When comparing the F_1-values of this method with the performance of the existing methods as shown in Table 1, we find that our method improves the F_1-value of the methods proposed in [7], [6] and [17] on each of the used data sets. For the restaurant data set, the F_1-value of the proposed method is 8.1%-point higher than the method proposed in [7], 5.5%-point higher than the method proposed in [17] and 4.8%-point higher than the method proposed in [6]. For the laptop data set, these values are 11.0%-point, 4.8%-point and 1.5%-point, respectively, and for the product data set the values are 9.1%-point, 0.5%-point and 8.6%-point, respectively.

To test whether the found differences are statistically significant, we perform a t-test. We also test whether the differences between the proposed method and the extended versions of the existing methods proposed in [17] and [6] are statistically significant. In order to perform the tests, it is necessary to have multiple evaluations. Therefore, we construct 30 bootstrap samples [18] for each of the used data sets. Every bootstrap sample is expected to contain about 63.2% of the unique sentences of the original data set. By using this sampling technique, we have obtained 30 new data sets for each of the original data sets.

Table 4: The performance of the proposed method for the different combinations on the different data sets.

Restaurant data set				
Threshold values	One threshold		Two thresholds	
Dependencies	Original	Proposed	Original	Proposed
Precision	0.523	0.518	0.521	0.516
Recall	0.542	0.580	0.547	0.585
F_1	0.532	0.547	0.534	0.548
Diff. in F_1		**+0.015**	**+0.001**	**+0.016**

Laptop data set				
Threshold values	One threshold		Two thresholds	
Dependencies	Original	Proposed	Original	Proposed
Precision	0.242	0.235	0.279	0.269
Recall	0.415	0.442	0.354	0.377
F_1	0.306	0.307	0.312	0.314
Diff. in F_1		**+0.001**	**+0.006**	**+0.008**

Product data set				
Threshold values	One threshold		Two thresholds	
Dependencies	Original	Proposed	Original	Proposed
Precision	0.132	0.131	0.140	0.139
Recall	0.271	0.285	0.290	0.305
F_1	0.177	0.180	0.189	0.191
Diff. in F_1		**+0.003**	**+0.012**	**+0.014**

These new sets will be used for method evaluation. We use the data sets as input for the existing methods, the extended versions of the existing methods and for the proposed method. A one-tailed paired t-test is performed to test whether the differences between the F_1-value of the proposed method and the F_1-values of the (extended) existing methods is significantly larger than zero. The results of the test are shown in Table 5. It shows that the F_1-measure of the proposed method is significantly higher than the F_1-values of the existing methods proposed in [7], [17] and [6] for each of the three used data sets at a 1.0% significance level. Furthermore, the proposed method performs significantly better in terms of the F_1-measure than the extended versions of the existing methods proposed in [17] and [6] for the restaurant and for the laptop data set. For the product data set, only the extended version of the method proposed in [17] performs slightly better than the proposed method.

Table 5: Results of the t-test on whether the F_1-value of the proposed method is significantly higher than the F_1-values of the (extended) existing methods.

	Restaurant data set		**Laptop data set**		**Product data set**	
Statistics	Mean	P-value	Mean	P-value	Mean	P-value
Proposed method	0.547		0.313		0.191	
Hu et al.	0.470	0.000	0.207	0.000	0.104	0.000
Scaffidi et al.	0.493	0.000	0.269	0.000	0.187	0.001
Scaffidi et al. with double threshold	0.494	0.000	0.281	0.000	0.198	1.000
Hai et al.	0.501	0.000	0.300	0.000	0.099	0.000
Hai et al. with extra dependencies	0.509	0.000	0.304	0.000	0.098	0.000

6 Conclusion

In this paper, we proposed a new method that extracts explicit features from consumer reviews. We employed and adapted various techniques that were used in two existing methods. The F_1-value of the proposed method is higher than the F_1-values of the separate methods on three different data sets. The differences are statistically significant at a 1.0% significance level for a restaurant, a laptop and a product data set. For the restaurant and for the laptop data sets, the proposed method also performs significantly better than the extended versions of the existing methods. For the product data set, the difference in F_1-value is statistically significant with respect to one of the extended versions of the earlier proposed methods but not the other.

Possible future work includes using domain knowledge, for example by employing ontologies, to find features, instead of purely statistical information. Also of interest are implicit features which were out of scope for this research. However, when performing aspect-level sentiment analysis, it is definitely beneficial to detect all features, not just the explicitly mentioned ones. Last, the method now assigns the average probability for words in a review text which are not in the general text. However, given the highly skewed, zipfian distribution of word frequencies, this could be improved upon. For instance, if the general text corpus is large enough, it stands to reason that all regular words are included, and that therefore words which do not appear in the text corpus, should have a very low probability associated to them, instead of the average probability.

Acknowledgment

The authors are partially supported by the Dutch national program COMMIT.

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules in Large Databases. In: Proceedings of the 20th International Conference on Very Large Databases (VLDB 1994). vol. 1215, pp. 487–499. Morgan Kaufmann (1994)
2. Androutsopoulos, I., Galanis, D., Manandhar, S., Papageorgiou, H., Pavlopoulos, J., Pontiki, M.: SemEval-2014 Task 4 (March 2014), `http://alt.qcri.org/semeval2014/task4/`
3. Bain, L.J., Engelhardt, M.: Introduction to Probability and Mathematical Statistics, 2nd edition. Duxbury Press (2000)
4. Bickart, B., Schindler, R.M.: Internet Forums as Influential Sources of Consumer Information. Journal of Interactive Marketing 15(3), 31–40 (2001)
5. Ganu, G., Elhadad, N., Marian, A.: Beyond the Stars: Improving Rating Predictions using Review Content. In: Proceedings of the 12th International Workshop on the Web and Databases (WebDB 2009) (2009)
6. Hai, Z., Chang, K., j. Kim, J.: Implicit Feature Identification via Co-occurrence Association Rule Mining. In: Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text processing (CICLing 2011). vol. 6608, pp. 393–404. Springer (2011)
7. Hu, M., Liu, B.: Mining Opinion Features in Customer Reviews. In: Proceedings of the 19th National Conference on Artifical Intelligence (AAAI 2004). pp. 755–760. AAAI (2004)
8. Hu, M., Liu, B.: Mining and Summarizing Customer Reviews. In: Proceedings of 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004). pp. 168–177. ACM (2004)
9. Leech, G., Rayson, P., Wilson, A.: Word Frequencies in Written and Spoken English: based on the British National Corpus. Longman (2001)
10. Liu, B.: Sentiment Analysis and Opinion Mining. Morgan & Claypool (2012)
11. Marneffe, M.C.D., MacCartney, B., Manning, C.D.: Generating Typed Dependency Parses from Phrase Structure Parses. In: Proceedings of International Conference on Language Resources and Evaluation (LREC 2006). vol. 6, pp. 449–454 (2006)
12. Marneffe, M.C.D., Manning, C.D.: Stanford Typed Dependencies Manual (September 2008), `http://nlp.stanford.edu/downloads/lex-parser.shtml`
13. Miller, G., Beckwith, R., Felbaum, C., Gross, D., Miller, K.: Introduction to WordNet: An On-Line Lexical Database. International Journal of Lexicography 3(4), 235–312 (1990)
14. Nakagawa, H., Mori, T.: A Simple but Powerful Automatic Term Extraction Method. In: Proceedings of the 19th International Conference on Computational Linguistics (AAAI 2004). pp. 29–35. Morgan Kaufmann Press (2002)
15. Pang, B., Lee, L.: Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval 2(1-2), 1–135 (2008)
16. Ross, S.M.: Introduction to Probability Models, 10th edition. Academic Press (2010)
17. Scaffidi, C., Bierhoff, K., Chang, E., Felker, M., Ng, H., Jin, C.: Red Opal: Product-Feature Scoring from Reviews. In: Proceedings of the 8th ACM Conference on Electronic Commerce (EC 2007). pp. 182–191. ACM (2007)
18. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley (2005)
19. Wu, H., Salton, G.: A Comparison of Search Term Weighting: Term Relevance vs. Inverse Document Frequency. In: Proceedings of the 4th Annual International ACM Conference on Information Storage and Retrieval. pp. 30–39 (1981)