Databases and Information Systems IX
G. Arnicans et al. (Eds.)

doi:10.3233/978-1-61499-714-6-241

Sentiment Analysis of Multiple Implicit Features per Sentence in Consumer Review Data

Nikoleta DOSOULA, Roel GRIEP, Rick DEN RIDDER, Rick SLANGEN, Ruud VAN LUIJK, Kim SCHOUTEN, and Flavius FRASINCAR [1]

Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands

Abstract. With the rise of e-commerce, online consumer reviews have become crucial for consumers' purchasing decisions. Most of the existing research focuses on the detection of explicit features and sentiments in such reviews, thereby ignoring all that is reviewed *implicitly*. This study builds, in extension of an existing implicit feature algorithm that can only assign one implicit feature to each sentence, a classifier that predicts the presence of multiple implicit features in sentences. The classifier makes its prediction based on a custom score function and a trained threshold. Only if this score exceeds the threshold, we allow for the detection of multiple implicit feature. In this way, we increase the recall while limiting the decrease in precision. In the more realistic scenario, the classifier-based approach improves the F_1-score from 62.9% to 64.5% on a restaurant review data set. The precision of the computed sentiment associated with the detected features is 63.9%.

Keywords. product reviews, implicit features, feature-based sentiment

1. Introduction

During the last years, a growing amount of retail activity is transferred from the street to the Web. Nowadays, people buy a wide range of consumer goods online using websites such as Amazon or Alibaba. These e-commerce companies often provide an easily accessible platform where consumers can share their experiences with and opinions about their purchases in the form of product reviews. As the required effort for writing these reviews becomes increasingly little, the number of product reviews on online retail shops sharply increased during the last decade. To illustrate this, in 2014 the number of reviews on Amazon exceeded 10 million [1]. Furthermore, the number of online reviewing platforms, where consumers leave behind product or service reviews, continues to grow.

Using these product reviews for decision making has become increasingly popular [2]. Where some consumers might be looking for specific comments on their potential purchase, others might only be interested in the overall sentiment or in the senti-

[1]Corresponding Author: Flavius Frasincar, Erasmus University Rotterdam, PO Box 1738, NL-3000 DR, Rotterdam, the Netherlands; E-mail: frasincar@ese.eur.nl.

ment per product aspect. However, the number of reviews can be high for some (popular) products, which makes reading all those reviews very time consuming. In order to lower these information costs, one of three pillars in the classical transaction cost model [3], an automatic assessment of the overall sentiment within consumer reviews is asked for.

The main aim of this paper is to contribute to the existing research on the detection of implicit features within consumer reviews. In particular, we seek to extend the method proposed in [4] by adding a classifier that predicts the presence of multiple implicit features within a sentence. In addition, we compute the sentiment per implicit feature in a sentence by identifying the scope of each implicit feature using a novel algorithm that considers three cases: only one implicit feature present, multiple implicit features present, and both implicit and explicit features present in a sentence. A preliminary version of this work focusing only on the detection of implicit features is given in [5].

The remaining part of this paper is organized as follows. Sect. 2 reviews the relating literature and addresses the possible shortcomings of previously proposed methods. After presenting our method in Sect. 3, we discuss the data set used in our experiments in Sect. 4. Sect. 5 then discusses the implementation of our proposed method and we evaluate its performance in Sect. 6, also by comparing it to previous work in the literature. Sect. 7 concludes this paper and proposes possible avenues for future research.

2. Related Work

This section discusses the relevant literature in the field that is concerned with the automated assignment of implicit product features within consumer reviews. Our proposed method is motivated by the shortcomings of existing approaches.

The vast majority of approaches in the literature focuses on finding the explicit features in sentences. This limited approach is understandable because often in reviews most of the features are explicitly mentioned in a sentence. However, as addressed before, features that are implicitly mentioned in reviews are equally important. In feature-based sentiment analysis the detection of implicit feature therefore plays an essential role. However, in order to obtain reliable results, sophisticated methods that can infer implicit features from sentences are required. This section addresses some of the most relevant approaches.

A method of detecting implicit features is proposed in [6]. More specifically, the method refers to a two-phase co-occurrence association rule mining approach. In the first phase, [6] mines a set of association rules from co-occurrences between opinion words and explicit features. Therefore, each opinion word is associated with a set of candidate features. In the second phase, the explicit features are clustered in order to obtain more powerful rules. If an opinion word is not linked with an explicit feature, the list of rules is checked in order to assign the most likely feature to this opinion word.

A similar approach to [6] is presented in [7]. Specifically, [7] mines as many association rules as possible between feature indicators and the corresponding features. Namely, the indicators are based on word segmentation, part-of-speech tagging, and feature clustering. As basic rules, the best rules in five different rule sets are chosen. In addition, three methods are proposed in [7] to find some set of rules: adding substring rules, adding dependency rules, and adding constrained topic model rules. In the final stage, the results of both approaches are compared where the latter one, using expanding methods, shows the best performance.

One pioneering method for the detection of implicit features is the one of [8], which originates from the following basic idea. A set of several selected opinion words is constructed and the reviews are scanned for so-called modification relationships between these opinions words and corresponding explicit feature words within the same sentences. In other sentences, these opinion words could appear without the presence of an explicit feature. Based on the modification frequencies, a set of candidate features is then determined for these sentences. Then, a co-occurrence matrix is built in which the numbers of co-occurrences between all notional words, i.e., also between non-opinion words and features, are calculated. Using this co-occurrence matrix, constrained by the set of candidates features, the algorithm in [8] selects features using information from all notional words within a sentence. The candidate features that are chosen have co-occurred with the corresponding opinion word before. For example, in the case of digital camera reviews, if the word 'good' appears within the same sentence as the explicitly mentioned features 'battery', 'lens', and 'material', these would be candidate features for an opinion word 'good'. From this set of candidate set of features, the implicit feature is inferred according to the associations between these candidate feature words and the rest of the notional words in the sentence, which are stored in the co-occurrence matrix.

It is important to keep the above described method in mind, since it forms an important building block of the method used by [4], on which this paper expands. The main difference in the approach by [4] is that it uses a supervised algorithm. Namely, consumers review data is used in which all implicit features are annotated. Therefore, co-occurrences can be calculated between these annotated implicit features and all words in the sentence. Based on the co-occurrences, scores are then assigned to potential implicit features, which in the case of [4] are *all* implicit features within the data set. Finally, the implicit feature with the highest score is assigned to the sentence. An advantage of [4], is that it can also be used to detect features that are not present explicitly within the data set. This is an improvement over the methods presented in [6], [7], and [8], where implicit features can only be detected when they also appear explicitly in the data set. Nevertheless, it relies on the existence of training data that is annotated with implicit features.

Furthermore, [4] improves on [8] by introducing a trained threshold in the assignment of implicit features. Where in the method presented in [8] relative low co-occurrence scores could already lead to linking an opinion word to a feature, the algorithm in [4] only assigns an implicit feature to a sentence when its score exceeds the learned threshold. The idea behind this is that when the co-occurrence frequencies are low, it is questionable whether the sentence should be linked to any feature at all. Especially in the case when there are many sentences without any implicit feature, the improvements by using such a threshold show to be large [4].

However, one apparent disadvantage of the detection procedure by [4] and [8] is that it rules out the possibility that a sentence contains two or more implicit features. This seems an unrealistic constraint, especially in the field of product reviews, where people are explicitly asked for their opinion. In fact, sentences containing two or more implicit features appear quite frequently. For instance, [9] makes the following observation in tweets that were collected from Twitter for their sentiment analysis: even short sentences may contain multiple sentiment types, concerning possibly different topics, e.g. *#fun* and *#scary* in "*Oh My God http://goo.gl/fb/K2N5z #entertainment #fun #pictures #photography #scary #teaparty*". [10] sees the same tendency in product review data. From their

Chinese restaurant review data, an intuitive example is extracted. In the sentence "the fish is great, but the food is very expensive", two obvious sentiment words can be noticed: 'great' and 'expensive'. Both these words implicitly refer to two different features which could be labeled respectively as 'quality' and 'price'.

3. Method

This section discusses our method for sentiment analysis of implicit features in product reviews. First, in subsection 3.1, we present our solution for implicit feature detection, and then, in subsection 3.2, we show our approach for computing the sentiment associated to each implicit feature.

3.1. Implicit Feature Detection

In this subsection discusses our implicit feature method that works as an extension on the algorithm developed by [4] in the sense that it allows for the extraction of multiple features per sentence. This more unrestrictive approach considers a more realistic scenario, in which sentences can be related to multiple implicit features.

We start with a short, formal description of the algorithm earlier presented in [4]. From the training data, the algorithm stores all unique annotated implicit features and all unique lemmas (which are the syntactic root form of a word) with their frequencies in list F and O. Furthermore, $|F| \times |O|$ matrix C stores the co-occurrences between all elements in F and O within sentences. Then, sentences in the test data are processed as follows. For each ith implicit feature $f_i \in F$, the sum of the ratios between the co-occurrence $c_{i,j} \in C$ of each jth word in the sentence and the frequency $o_j \in O$ of that word is calculated:

$$Score_{f_i} = \frac{1}{n} \sum_{j=1}^{n} \frac{c_{i,j}}{o_j}, \tag{1}$$

where n is the number of words in a sentence. Finally, the implicit feature with the highest score is assigned to the sentence when it exceeds a trained threshold. When there is no score that exceeds the threshold, no feature is assigned to the sentence. The training of the threshold is only based on the training data and is executed by simply finding the threshold value between 0 and 1 which yields the best performance.

One approach to extend the algorithm to a more realistic scenario is by selecting all implicit features that exceed the trained threshold (see Sect. 2). However, when only a small proportion of the data set consists of sentences that contain more than one implicit features, the precision of the algorithm would suffer from such a crude selection mechanism. To understand this effect, one should realize that when specific words co-occur often with different implicit features, sentences in which these words are present consequently have a high score for more than one implicit feature. However, assigning more than one implicit feature to each of such sentences based on these scores might be naive when only few sentences are known to contain more than one implicit feature. Another approach to allow for multiple features is to use a classifier to determine the number of implicit features that is likely to be present within the sentence. Subsequently,

Algorithm 1 Algorithm training using annotated data.

```
Construct list F of unique implicit features
Construct list O of unique lemmas with frequencies
Construct co-occurence matrix C
for all sentence s ∈ training data do
    for all word w ∈ s do
        if ¬(w ∈ O) then
            add w to O
        end if
        O(w) = O(w) + 1
    end for
    for all implicit feature f ∈ s do
        if ¬(f ∈ F) then
            add f to F
        end if
        for all word w ∈ s do
            if ¬((w, f) ∈ C) then
                add (w, f) to C
            end if
            C(w, f) = C(w, f) + 1
        end for
    end for
end for
Train threshold for the classifier through linear search
Train threshold for the feature detection algorithm through linear search
```

the algorithm could assign features with top scores to a sentence, where now the number of assignments is based on the classifier's prediction. One should bear in mind, however, that this strategy now potentially suffers from the imperfect nature of both the classifier and the implicit feature extraction algorithm, which possibly leads to lower precision.

The method that we present works as a combination of the two above-mentioned methods such that we can utilize the advantages of both, while minimizing their disadvantages. In particular, we use a classifier in order to detect for every sentence whether there it contains more than one implicit features. If the classifier predicts more than one implicit feature, all features with a score exceeding the threshold will be assigned to the sentence. Otherwise, only the feature with the highest score could be assigned to the sentence, that is, if it exceeds the trained threshold. Hence, the classifier produces the binary result whether or not to allow for multiple features. The pseudocode describing the described method is shown in Algorithm 1 and Algorithm 2.

The classifier calculates a score based on a number of sentence characteristics that are related with the number of implicit features k_s within a sentence s. When the score for a sentence exceeds another trained threshold, the classifier predicts multiple implicit features to be present. The score function uses the following variables: (i) number of nouns ($\texttt{\#NN}_s$), (ii) number of adjectives ($\texttt{\#JJ}_s$), (iii) number of commas ($\texttt{\#Comma}_s$), and (iv) the number of 'and' words ($\texttt{\#And}_s$). In order to determine the relation between these predictor variables and the number of implicit features, we estimate the following logistic regression equation by maximum-likelihood:

Algorithm 2 Algorithm execution on new sentences in the test data.

Input: trained thresholds $kThreshold$ and $fThreshold$
Construct list NN with the number of nouns per sentence
Construct list JJ with the number of adjectives per sentence
Construct list CM with the number of commas per sentence
Construct list A with the number of 'and' words per sentence
Obtain $\hat{\beta}$'s from logistic regression using the full data set
for all sentence $s \in$ test data **do**
  $kScore = \hat{\beta}_0 + \hat{\beta}_1 NN(s) + \hat{\beta}_2 JJ(s) + \hat{\beta}_3 CM(s) + \hat{\beta}_4 A(s)$
  $currentBestFeature = empty$
  $fScoreOfCurrentBestFeature = 0$
  for all feature $f \in F$ **do**
    $fScore = 0$
    for all word $w \in s$ **do**
      $fScore = fScore + C(w, f)/O(w)$
    end for
    if $kScore > kThreshold$ **then**
      if $fScore > fThreshold$ **then**
        Assign feature f to s
      end if
    else if $fScore > fScoreOfCurrentBestFeature$ **then**
      $currentBestFeature = f$
      $fScoreOfCurrentBestFeature = fScore$
    end if
  end for
  if $\neg(kScore > kThreshold)$ **then**
    if $fScoreOfCurrentBestFeature > fThreshold$ **then**
      Assign $currentBestFeature$ to s
    end if
  end if
end for

$$Score_{k_s} = \log\left(\frac{p_s}{1-p_s}\right) = \beta_0 + \beta_1 \texttt{\#NN}_s + \beta_2 \texttt{\#JJ}_s + \beta_3 \texttt{\#Comma}_s + \beta_4 \texttt{\#And}_s, \quad (2)$$

where p_s is the probability that sentence s contains multiple implicit features. The coefficients are estimated using the full data set. The implementation of this regression approach is discussed in more detail in Sect. 5.

This extended algorithm is now trained in two steps, only using the training data. First, the threshold for the classifier is trained in terms of prediction performance. Second, the threshold for the feature detection algorithm, now using the prediction of the classifier, is trained (as described in the second paragraph of this section) to optimize the feature detection performance.

As a final remark, a limitation of this method is that it requires a sufficiently large data set in which the implicit features are annotated. The reason for this is that the training of the algorithm is executed on annotated implicit features. However, the benefit of this approach is that the algorithm is now able to detect all implicit features within the data set, and not only the features that are (also) *explicitly* present in the data set.

3.2. Implicit Feature Sentiment Detection

In this subsection a new method for finding the sentiment of implicit features is proposed. Shortcomings of previous methods are that they assume that there can only be one feature in a sentence and that implicit and explicit features do not co-occur in a sentence. We address both cases in our method. Our method is composed of two steps. The fist step is to identify the part of the sentence in which the feature is implied. The second step is to find the sentiment of this part of the sentence. Several techniques described in [11] and [12] are used in this method. The method for extracting explicit features using dependency relations, presented in [12], might be useful to also find the words associated with an explicit feature, if you reverse its purpose. The method for finding the sentiment of a sentence, presented in [11], is used because it is a straightforward and intuitive method. However, this method is applied for the sentence scope of the considered implicit feature.

3.2.1. Step 1: Finding the Relevant Sub-sentence

First, a method for finding the part of the sentence in which the feature is implied is discussed. To find this sub-sentence, all senses of the words in the sentence need to be identified. The most likely senses of all words in the sentence are found using the Lesk algorithm [13], using the MIT Java Wordnet Interface (JWI) and the Java API for WordNet Searching (JAWS). Hereafter, there are three possible cases of implicit and explicit features in a sentence if a sentence contains an implicit feature. To find the different cases it is assumed that all features, both implicit and explicit, are known beforehand. These three different cases are presented with an illustrating example:

1. Only one implicit feature and no explicit feature. The example is: *"It is slightly larger than an ipod."*. Here the implied feature is *size*.
2. More than one implicit feature and no explicit feature. The example is: *"It seems quite small to me and very light"*. Here the implied features are *size* and *weight*.
3. One or more implicit features and one or more explicit features. The example is: *"It has a beautiful design, very easy to use, and the battery duration is amazing"*. Here the implied feature is *user interface*. However, there are also two explicit features: *design* and *battery duration*.

These three cases all require slightly different methods for finding the sentiment of the implicit feature. The cases will be discussed in the given order, with their problems and the solutions to these problems below.

Case 1: Only One Implicit Feature. If there is only one implicit feature and no explicit feature, the implicit feature is discussed in the whole sentence. In this case, the assumption that only one feature per sentence is discussed, is not violated. Therefore the sentiment of the feature can be found by extracting the sentiment of the whole sentence.

Case 2: More than One Implicit Feature. A more challenging problem is when there is more than one implicit feature in a sentence. Clearly, the sentiment of these features cannot be found by simply determining the sentiment of the whole sentence. The features might have a different sentiment than the overall sentence sentiment. To find the sentiment of one of these features, the part of the sentence referring to this feature needs to be identified.

To identify the part of the sentence that refers to a feature, the similarity of all nouns in the sentence with the implicit features is calculated, using WordNet based meth-

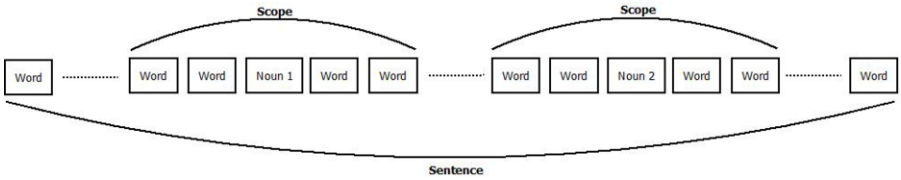

Figure 1. An example of the use of the scope to extract the sub-sentence if the scope is k=2, with 2 nouns with similarity to the implicit feature in the sentence.

ods [14]. For this purpose, the proper senses of the implicit features are manually found in WordNet. Only nouns are considered because only the similarity of words with the same part-of-speech can be calculated and features are mostly nouns. According to previous studies, the best performing algorithm to find the similarity measure of two synsets is the algorithm of Wu and Palmer [14]. A synset is a set containing information of a word, such as the meaning, the sense of the word, and a POS-tag. The formula to calculate the similarity between synset u and synset r is given in Eq. (3) [15]. In this formula, *depth* is the depth of a synset in the WordNet tree, *length* the distance between the two synsets in WordNet, and *LCS* stands for Least Common Subsumer, which is the synset with the highest depth in the WordNet tree that is a hypernym of both synsets u and r.

$$sim(u,r) = \frac{2 \times depth(LCS(u,r))}{length(u,r) + 2 \times depth(LCS(u,r))} \tag{3}$$

The similarity is between 0 and 1, where a higher number refers to more similar synsets. Not all features are single word features. In the example of the third case, mentioned before, the feature is *battery duration* and thus a bi-gram. To calculate the similarity of an n-gram and a word, the similarity of the word of the sentence with all n parts of the n-gram is calculated. These n values are then multiplied to get one similarity score. When all similarity scores between the words in the sentence and the implicit features are calculated, the words are linked to the feature with the highest similarity. If the similarity scores of a word with the implicit features are the same, the word is assumed to refer to all implicit features. This can be illustrated by an example. In the sentence *"It's far heavier and much larger which might not seem to matter except if you use it while running or keep it in your pocket."* two features are implied, *"size"* and *"weight"*. Both features have a similarity of 0.222 to the noun *"pocket"*. Since the product might be both too heavy and too big to carry in a pocket, the noun *"pocket"* refers to both implicit features.

Once all similarities of the nouns in the sentence and the implicit features are found, the part of the sentence that refers to the implicit feature is extracted. This is done by extracting all words within a range k from all nouns with a similarity to that implicit feature. The extracting of the sub-sentences is illustrated in Figure 1.

Case 3: Both Explicit and Implicit Features. The most complex case is when there are both explicit and implicit features in a sentence. In these cases not all words carry a sentiment about the present implicit features, as they can be associated with explicit features. These words need to be filtered out of the sentence in order to find the sentiment of implicit features. This extraction is done using a technique proposed in [12].

This method uses dependencies in sentences to find words that carry a sentiment of an explicit feature. The Stanford part-of-speech parser [16] is used to get these dependencies. All words that have a significant dependency with the explicit features should not be included in the sentiment extraction for the implicit features, so these will be removed from the sentence. Words that carry sentiment of an explicit feature modify the sentiment of the explicit feature, therefore significant relations are a set of modifier types. Some examples of these modifier types are: *"adverbial modifier" (advmod)*, *"adjectival modifier" (amod)*, and *"nominal subject" (nsubj)*. These dependency relations are best explained by examples. Good example sentences containing these modifiers are: *"Genetically modified food"* with *advmod(modified, genetically)*, *"Sam eats red meat."* with *amod(meat, red)*, and *"The baby is cute."* with *nsubj(cute, baby)* [17].

The remaining sentence carries sentiment solely of the implicit features. Therefore, after removing the explicit features and their associated words from the sentence, the previously described method for Case 2 can be applied to get the sentiment of all implicit features if there are more than one implicit features. If there is only one implicit feature, the method for the first case is to be used.

3.2.2. Step 2: Extracting the Sentiment

Once the part of the sentence that refers to the implicit feature is found, the method for extracting the sentiment presented in [11] is used. In this method, all senses of the words are used to find the sentiment of the words, using SentiWordNet [18,19]. The sentiment found by using SentiWordNet is expressed by three scores: objectivity, negativity, and positivity. These scores range from 0 to 1, with a total sum of 1. The resulted score for a synset is the positivity score minus the negativity score. Adding up all these individual sentiment scores, after handling negations, give the total score of a sentence.

The default threshold for determining whether the sentence is positive or negative used in [11] is 0. This, however, does not handle neutral sentences and it might not be optimal. Therefore two thresholds for determining the sentiment of an implicit feature are used in this method, ε_1 and ε_2. If the sentence score is larger or equal than ε_2, the sentiment of the implicit feature is positive. If the sentence score is smaller than ε_1, the sentiment of the implicit feature is negative. If the sentence score is between ε_1 and ε_2, the sentiment of the implicit feature is neutral. Implicit features with a conflicted sentiment are ignored in this method.

4. Data Analysis

The data set which is used to build up and validate the method proposed in the previous section consists of a collection of restaurants reviews [20]. Every review sentence is assigned to at least one of five so-called review aspect categories: 'food', 'service', 'ambience', 'price', and 'anecdotes/miscellaneous'. These aspect categories are generally not explicitly referred to in a sentence, but can be inferred from each sentence. Therefore, these aspect categories operate as *implicit* features of the product, i.e., the restaurant. In the data set, both implicit and explicit restaurant features are labeled.

All 3,044 sentences in the restaurant data set contain at least one implicit feature. However, in order to obtain a better performance test of our classifier for the number of implicit features present in each sentence, the fifth category of 'anecdotes/miscella-

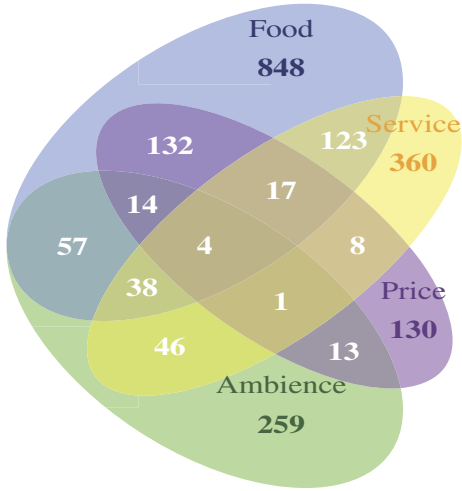

Figure 2. Co-occurrence frequencies of the four unique implicit features in our data set [5].

neous' is removed from the data set. This particular category seems most appropriate for removal, as it does not describe a unique implicit feature, but refers to the general context 'miscellaneous'. In this way, the number of implicit features in our data set has a wider distribution because part of the set now consists of sentences without an implicit feature. As consumer review sentences generally do not always contain an implicit feature, the performance of our classifier on this more realistic scenario is interesting. Furthermore, in this setting the influence of the threshold parameter in the algorithm by [4] in combination with our classifier can be measured.

As the main purpose of the method that we propose is to search for multiple implicit features in each sentence, it seems worthwhile to examine to what extent multiple features are present in one sentence. Figure 2 shows the frequency of all possible co-occurrences between the four unique implicit features. More than 4% of the sentences implicitly refer to both 'food' and 'price', and almost the same percentage corresponds to the co-occurrence of 'food' and 'service'. The remaining combinations of two implicit features appear less frequently in the same sentence in our restaurant review data set.

The implicit feature sentiment of the data set are divided into 4 semantic categories. These categories are: positive, negative, neutral, and conflicted. 2179 implicit features in this data set have a positive sentiment, 839 implicit features have a negative sentiment, 500 features are neutral, and 195 of the implicit features have a conflicted sentiment. The percentage of implicit features percentage with a positive review is 58.7%.

5. Implementation

To predict the presence of multiple implicit features, we use the score function as given in Eq. (2). We think of this score function as a general rule for categorized review data such as our restaurant review data set. In order to specify the correct score function, however, sufficient amount of this type of consumer review data is required. Constrained by resources, however, only the same restaurant review data set is available to us. Therefore,

Table 1. Coefficients of logistic regression (2) for the classifier.

Predictor Variable	Coefficient	p-value
`Constant`	−3.019479	0.0000
`#NN`$_s$	0.116899	0.0002
`#JJ`$_s$	0.335530	0.0000
`Comma`$_s$	0.216417	0.0004
`And`$_s$	0.399415	0.0000

the score function is not trained on a training part of the data set, and then tested on a test part. Instead, the full data set is used in order to maximize the information available to us.

We estimate the $Score_{k_s}$ function (2) using logistic regression. Table 1 displays the results. The p-values indicate that our variables are highly significant, i.e., for significance levels below 1%. Apart from the variables that we include, we also test implementing the number of words in a sentence and the number of grammatical subjects in a sentence. Neither of these variables yield a significant improvement. Intuitively, this can be explained because the variables for the number of nouns and adjectives already capture the relevant information that lies within the number of words within a sentence. The number of subjects possibly does not perform better than the number of nouns because often the subject in a sentence is the product instead of the feature.

The classifier predicts multiple implicit features for sentence s when $Score_{k_s}$ is larger than a certain threshold. We can therefore train the classifier by determining the optimal threshold. In order to do so, we isolate the performance of the classifier by assuming that the feature detection part of the algorithm is perfect. This way, the errors made by the classifier are isolated and can thus be minimized by means of altering the threshold.

6. Evaluation

Evaluation of the implemented method is based on 10-fold cross-evaluation. This means that the whole data set is split into two subsets: one part contains 90% of the data, the other part 10%. The algorithm is then trained on this 90% of the data set. The trained algorithm then detects the implicit features in the remaining 10% of the data. This procedure is repeated 10 times, where there is no overlap in the 10 hold-out samples. For each fold, the F_1-score is calculated and finally averaged to provide the measure for the performance of the algorithm.

The performance measure we use for predicting implicit features is the F_1-score. Using the F_1-score as the performance measure allows for easy comparison with previous work, as it is one of the standard performance measures within the literature.

Because the different training and test subsamples used in the cross-evaluation are generated randomly, we run our algorithm 10 times. Figure 3 shows the results, in terms of mean F_1-scores, following from our proposed method (the blue bars). To provide more insights into our results, Figure 3 also depicts F_1-scores of the algorithm with both a perfect classifier (the red line) and with a perfect feature detection algorithm (the green

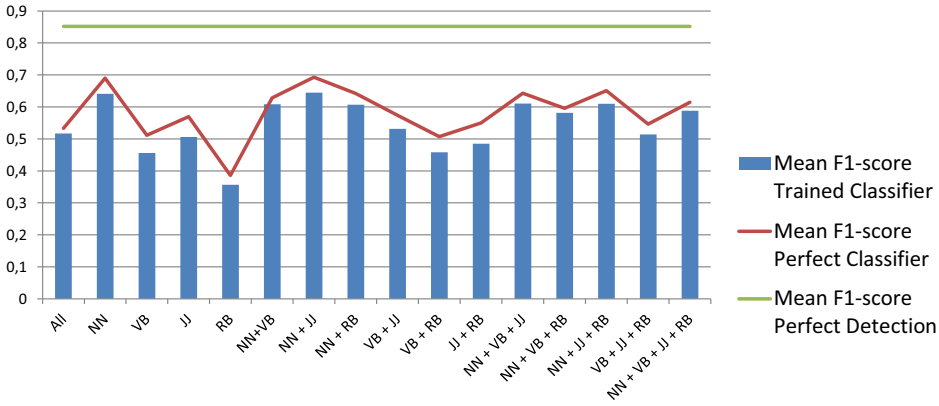

Figure 3. Mean F_1-scores with different part-of-speech filters [5].

line). The scores using a perfect classifier are computed by always passing the correct prediction (in terms of the presence of multiple implicit features) onto the feature detection algorithm. The scores with the perfect feature detection are found by, based on the prediction of the classifier, assigning a number of golden implicit features to the sentences.

Results are given for different part-of-speech filters, which are used to filter out possibly irrelevant words in the co-occurrence matrix that could be harmful to the performance of the algorithm. Figure 3 shows the scores for 16 different part-of-speech filters. The filters include only the words of types that are mentioned, where `NN` stands for nouns, `VB` for verbs, `JJ`, for adjectives and `RB` for adverbs. Examining the F_1-scores in Figure 3, we find that the best results are obtained using the `NN+JJ` part-of-speech filter. That is, filtering for nouns and adjectives, we obtain an F_1-score equal to 64.5%. We note that the F_1-score we find for using the `NN` filter is only marginally worse, namely 64.1%.

Since our proposed method extends the one presented in [4], we start by evaluating the increase in performance as a result of our extension. In order to do so, we also evaluate the unextended algorithm as presented in [4] 10 times using the `NN+JJ` part-of-speech filter. Again, we note that each evaluation provides slightly different results due to the random nature of the cross-evaluation method. We find a mean F_1-score of 62.9% for the algorithm without the classifier.[2] Hence, comparing this to our 64.5%, we find an improvement of 1.6 percentage points. We test for significance by means of a two-sample t-test. This results in a t-test statistic equal to 12.0, which indicates a significant improvement at the significance level of 1%.

After the implicit features are found, the addition of finding the scope of each implicit feature is evaluated. This is done by evaluating the precision of the method with different scope ranges for the considered implicit feature, parameter k. This is done from 0 to 5, in which 0 means that only the sentiment of the assigned noun(s) to the implicit feature is calculated. The evaluation is limited to 5 because if this number will grow larger, it is hypothesized that words with greater distances do not contribute to the sentiment. The precision of the method is presented in Table 2 (note that Case 3 does not appear in our data set). Here it is shown that the optimal k is 4, and the precision for

[2] We note that in [4], based on a number of runs, a maximum F_1-score of 63.3% is reported.

Table 2. Results of the precision using only Case 1 and Case 2 of the algorithm using different scopes.

k	0	1	2	3	4	5
	0.631	0.634	0.635	0.638	**0.639**	0.638

sentiment feature detection is 63.9%. The optimal values for ε_1 and ε_2 were found to be -0.025, which means that we do not predict the neutral class.

7. Conclusion

In our proposed method we construct a classifier that predicts the presence of multiple implicit features using a score function. The score function is based on four simple sentence characteristics: (i) number of nouns, (ii) number of adjectives, (iii) number of commas, and (iv) the number of 'and' words. The function parameters are estimated by means of logistic regression and we train a threshold for better performance. Based on the prediction of the classifier for a given review, the feature detection part of our algorithm then looks for either one or multiple implicit features.

Considered on a restaurant review data set, our approach shows small, but significant improvement with respect to the constrained method in [4]. That is, we improve the F_1-measure by 1.6 percentage points. Based on analysis of the performance of our classifier we conclude that we capture a reasonable (considering its simplicity) part of the full potential of our approach. The performance and potential of the classifier is, however, dependent on the distribution of the number of implicit features per sentence within the data set. That is, when consumer reviews frequently cover multiple implicit features per sentence, our more realistic approach is desirable. For the precision of the sentiment associated with the detected implicit features we obtain 63.9%.

In our approach we determine a *general* relation between sentences written in consumer reviews and the number of implicit features. Nonetheless, it might be desirable to integrate the specification and estimation of this relation in the training part of the algorithm in order to make it specifically effective for a given data set. One promising path for future work is therefore to train a classifier for the number of implicit features by using more advanced machine-learning techniques, such as Support Vector Machines. Also, rule learning methods could be employed in order to determine more indicators for the presence of multiple implicit features.

The case that filters the sentiment of possible explicit features, Case 3, is not evaluated since there is no suitable data set available, yet. This should be evaluated in future work. Furthermore, it might be interesting to investigate a different, more subtle, method for determining the sentiment of a (sub-)sentence. The proposed method simply adds up the sentiment of the words in the context of an implicit feature. A possible different method, for example, is by appointing proximity weights to words in a sentence.

Acknowledgments

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

References

[1] K. Floyd, R. Freling, S. Alhoqail, H.Y. Cho, and T. Freling. How online product reviews affect retail sales: A meta-analysis. *Journal of Retailing*, 90(2):217–232, 2014.

[2] S. Senecal and J. Nantel. The influence of online product recommendations on consumers' online choices. *Journal of Retailing*, 80:159–169, 2004.

[3] C.J. Dahlman. The problem of externality. *Journal of Law and Economics*, 22(1):141–162, 1979.

[4] K. Schouten and F. Frasincar. Finding implicit features in consumer reviews for sentiment analysis. In *Proceedings of the 14th International Conference on Web Engineering (ICWE 2014), volume 8541 of Lecture Notes in Computer Science*, pages 130–144. International World Wide Web Conferences Steering Committee, 2014.

[5] N. Dosoula, R. Griep, R. den Ridder, R. Slangen, K. Schouten, and F. Frasincar. Detection of multiple implicit features per sentence in consumer review data. In *Proceedings of the 12th International Baltic Conference on Databases and Information Systems (DB&IS 2016)*, volume 615 of *Communications in Computer and Information Science*, pages 289–303. Springer, 2016.

[6] Z. Hai, K. Chang, and J. Kim. 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)*, volume 6608, pages 393–404, 2011.

[7] W. Wang, H. Xu, and W. Wan. Implicit feature identification via hybrid association rule mining. *Expert Systems with Applications*, 40(9):3518–3531, 2013.

[8] Y. Zhang and W. Zhu. Extracting implicit features in online customer reviews for opinion mining. In *Proceedings of the 22nd International Conference on World Wide Web Companion (WWW 2013 Companion)*, pages 103–104. International World Wide Web Conferences Steering Committee, 2013.

[9] D. Davidov, O. Tsur, and A. Rappoport. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 241–249, 2010.

[10] J. Zhu, H. Wang, M. Zhu, B.K. Tsou, and M. Ma. Aspect-based opinion polling from customer reviews. *IEEE Transactions On Affective Computing*, 2:37–49, 2011.

[11] A. Hogenboom, P. van Iterson, B. Heerschop, F. Frasincar, and U. Kaymak. Determining negation scope and strength in sentiment analysis. In *Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics (SCM 2011)*, pages 2589–2594. IEEE, 2011.

[12] S. Mukherjeean and P. Bhattacharyya. Feature specific sentiment analysis for product reviews. In *Proceedings of the 13th International Conference on Intelligent Text Processing Part I (CICLing 2012)*, volume 7181, pages 475–487. Springer Berlin Heidelberg, 2012.

[13] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation (SIGDOC 1986)*, pages 24–26. ACM, 1986.

[14] M. Capelle, F. Frasincar, M. Moerland, and F. Hogenboom. Semantics-based news recommendation. In *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics (WIMS 2013), Article No. 22*. ACM, 2013.

[15] Z. Wu and M. Palmer. Verb semantic and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL 1994)*, pages 133–138, 1994.

[16] M.C. De Marneffe, B. MacCartney, and C.D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of Language Resources and Evaluation Conference (LREC 2006)*, volume 6, pages 449–454, 2006.

[17] M.C. De Marneffe and C.D. Manning. Stanford typed dependencies manual, September 2008.

[18] S. Baccianella, A. Esuli, and F. Sebastiani. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the 7th International Conferance on Language Resources and Evaluation Conference (LREC 2010)*, pages 2200–2204, 2010.

[19] A. Esuli and F. Sebastiani. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th International Conferance on Language Resources and Evaluation Conference (LREC 2006)*, volume 5, pages 417–422, 2006.

[20] G. Ganu, N. Elhadad, and A. Marian. Beyond the stars: Improving rating predictions using review content. In *Proceedings of the 12th International Workshop on the Web and Databases (WebDB 2009)*, 2009.