Progress in Artificial Intelligence (2018) 7:295–306
https://doi.org/10.1007/s13748-018-0163-7

REGULAR PAPER

Review-aggregated aspect-based sentiment analysis with ontology features

Sophie de Kok[1] · Linda Punt[1] · Rosita van den Puttelaar[1] · Karoliina Ranta[1] · Kim Schouten[1] · Flavius Frasincar[1]

Received: 31 January 2018 / Accepted: 2 September 2018 / Published online: 12 September 2018

Abstract
With all the information that is available on the World Wide Web, there is great demand for data mining techniques and sentiment analysis is a particularly popular domain, both in business and research. Sentiment analysis aims to determine the sentiment value, often on a positive–negative scale, for a given product or service based on a set of textual reviews. As fine-grained information is more useful than just a single overall score, modern aspect-based sentiment analysis techniques break down the sentiment and assign sentiment scores to various aspects of the product or service mentioned in the review. In this work, we focus on aspect-based sentiment analysis for complete reviews, as opposed to determining sentiment for aspects per sentence. Furthermore, we focus on semantic enrichment by employing ontology features in determining the sentiment value of a given pair of review and aspect. Next to that, we compare a pure review-level algorithm with aggregating the sentiment values of individual sentences. We show that the ontology features are important to correctly determine the sentiment of aspects and that the pure review-level algorithm outperforms the sentence aggregation method.

Keywords Aspect-based sentiment analysis · Ontologies · Sentiment aggregation · Sentiment reasoning

1 Introduction

The rapid growth of the World Wide Web has led to an explosion in the amount of information that is available on this platform [10]. As a result, the recognition of information retrieval as a value-added field has increased correspondingly. The Web has made it possible for consumers to share their opinions and experiences about products and services and they love to do so, according to a survey among more than 2000 American individuals [8]. The demand for user information is one of the major driving forces behind the interest in opinion mining [12], where the goal is to determine the opinion of a group of people regarding a topic [7]. However, research states that information about goods and services is often missing or confusing, and the amount of it can be overwhelming [8]. An improved way of accessing consumer opinions is thus needed to aid businesses and consumers alike.

A common way to extract information from review texts is to perform sentiment analysis, also referred to as opinion mining. This approach is defined as finding the quadruple (g, s, h, t), where g represents the sentiment target, s the sentiment, h the opinion holder, and t the time at which the opinion was expressed [12]. A general approach is to take a whole sentence or review for g. A downside to this approach is that it only assigns a single polarity value to a sentence or a review. Consequently, it can not capture different sentiments within one segment of text. Rather than finding only the general sentiment of a document or a sentence, aspect-based sentiment analysis captures different aspects of the discussed entity and the sentiments expressed about these. For example, when dealing with restaurant reviews, a consumer can be positive about the service and be negative about

✉ Kim Schouten
schouten@ese.eur.nl

Sophie de Kok
408695sk@student.eur.nl

Linda Punt
409135lp@student.eur.nl

Rosita van den Puttelaar
414662rp@student.eur.nl

Karoliina Ranta
413163kr@student.eur.nl

Flavius Frasincar
frasincar@ese.eur.nl

[1] Erasmus University Rotterdam, P.O. Box 1738, NL-3000, DR, Rotterdam, The Netherlands

the food quality. Therefore, aspect-based sentiment analysis allows for a more detailed analysis that utilizes more of the information provided by the available text [21].

Most aspect-based sentiment analysis approaches are concerned with two tasks, namely aspect detection and aspect sentiment classification [21]. Aspect detection is defined as determining the different aspects of an entity in a particular part of the text, like a sentence or a review. For example, in the sentence "Service is not what one would expect from a joint in this price category," 'service' is an aspect represented by this review. Sentiment classification assigns a sentiment to the aspects found in the text. In this example, the sentiment expressed about the aspect 'service' is negative.

A method that has been shown to perform well for aspect detection and sentiment classification is the Support Vector Machine (SVM) model [15]. When an SVM model is used, a feature vector of values is created for every instance to be classified and the model is taught using training data to interpret these values. Machine learning methods such as SVM models need a substantial amount of training data to obtain acceptable accuracy, as individual elements have little predictive value [2]. This reveals the need for large training data, but large amounts of annotated data are often not available for a product or service. To address this problem, we choose to use external information in the form of an ontology [4], which is hypothesized to lessen the need for training data. Ontologies use a shared vocabulary for a certain domain and include axioms to define the relationships between different domain concepts [6]. These axioms can help derive information that is only implicitly stated. For this reason, the employment of an ontology is expected to improve the performance of sentiment analysis [16] and lessen the need for training data.

In our paper, we focus on the second sub-problem of aspect-based sentiment analysis, namely sentiment classification. Many solutions to the first task have already been provided, for example [9,20]. We focus on review-level sentiment classification [17], meaning that each aspect is assigned a sentiment based on information from the whole review. This is in contrast to the regular sentence-level setting, where aspects are assigned a sentiment score for each individual sentence they appear in. While sentences usually contain just a single sentiment value for a given aspect, this is less the case when looking at a complete review. Hence, various sentiment values will need to be combined to get to a final classification. As such, we investigate two approaches for review-level aspect-based sentiment analysis, one at the review level and one that aggregates sentence level sentiment label predictions. We expect that the review-level approach gives better results, because reviewers tend to write in interconnected sentences.

In this paper, we hypothesize that review-level aspect-based sentiment analysis using an ontology gives better results than methods which do not include the use of an ontology. By using an ontology as a knowledge base, we can define concepts and relationships which could help in performing our task. For example, by knowing that someone liked the pasta, we can infer that the food was liked, as pasta is a type of food. We further choose to use SVM as our machine learning algorithm because it deals well with large amounts of features [3] and it has proven to be a robust model for text classification tasks. Furthermore, when using the linear kernel, we can use the internal feature weights to see how important each feature is in the model. Last, we consider the proposition that less training data is needed when we include an ontology in our model.

This work is an extended version of [5], published in the proceedings of the Cognitive Computing track of the Symposium on Applied Computing in 2018. Compared to that paper, this work has additional information about the investigated algorithm. For example, the optimization process is made more explicit and pseudocode is added for the grammatical word window feature. In terms of the evaluation, we included two additional comparisons: the used multi-class classifier versus a binary classifier and the used linear SVM versus an SVM with RBF kernel. This explains our decision to use the linear multi-class SVM model. Last, we added a stepwise feature type analysis, where starting with the base model, the performance is measured each time a feature type is added to the model.

This paper is structured as follows. First, we discuss the related work in Sect. 2, followed by information about the used dataset in Sect. 3. Then, in Sects. 4 and 5, we explain the proposed methodology and analyze the performance of our algorithms. Last, Sect. 6 contains our concluding remarks and possible directions for future work.

2 Related work

In this section, we discuss work related to the field of sentiment analysis. Firstly, [21] provides a survey on aspect-level sentiment analysis. For this task, [21] considers three different types of methods: dictionary-based, supervised machine learning, and unsupervised machine learning. In this paper, we use a supervised machine learning method. We do so because we have supervised data available and supervised methods work in general better than unsupervised ones.

Various studies investigate whether the inclusion of an ontology improves results. In [20], the focus is on a knowledge-based approach that complements standard machine learning algorithms. The authors of [20] enhance the sentiment analysis using domain ontology information. By incorporating common domain knowledge into an ontology, classification performance for both aspect detection and aspect sentiment classification can be improved. The authors found words within sentences that are in the ontology and

are related to the aspect under consideration. They then provided all the superclasses of the ontology concept to the employed machine learning algorithm for the classification tasks. For both classification tasks [20], works with an existing classifier, the linear Support Vector Machine. Contrary to [20], which focuses on aspect-based sentiment analysis at the sentence level, this paper considers aspect-based sentiment analysis at the review-level. Furthermore, we enhance the ontology application using additional ontology-related features such as synonyms.

Wei and Gulla [24] propose an approach that they call HL-SOT. HL-SOT is a hierarchical learning (HL) process in combination with a sentiment ontology tree (SOT). A sentiment ontology tree has a tree-like appearance and the complete SOT consists of numerous sub-SOTs. A SOT has an attribute root node that has two leaf children that represent the positive and negative sentiment that is associated with the attribute. Each sub-SOT represents a sub-attribute and is given as a child of the root node of the parent attribute SOT. Furthermore, each sub-SOT is assigned its own classifier with its own threshold value. The ontology is used by the authors to ensure that a text is labeled to contain an attribute only if all its parent attributes have also been mentioned within the same text segment. The proposed approach, however, can be disadvantageous when considering short reviews as these may express opinions on certain attributes without the mention of parent attributes. Unlike [24], we do not use different classifiers for different attributes. However, we do take into account that the sentiment polarity of some words is dependent on the product attribute being described.

Last, in [11], a system is considered in which an individual can search for information on a specific product. The authors suggest using an ontology to improve this system. They recommend a procedure in which the ontology is used as an alternative representation of a domain-specific sentiment lexicon. The described ontology contains products, product features, sentiment words that are specific to a product feature, and the associated polarity. This sentiment ontology is then used in combination with manually crafted sentiment lexicons and NLP rules to determine the polarity of a product feature and sentiment word pair. This results in an accuracy comparable with previous works utilizing machine learning techniques. The authors of [11] compare their ontology-based approach to machine learning techniques; however, in this paper we combine both these approaches.

3 Data

In this paper, we use a dataset of restaurant reviews from SemEval 2016 [17]. The dataset consists of training data and test data. The training data is used to develop and train our machine learning algorithm, and the test data is used to evaluate the performance of our algorithm. We define a `notion` as an aspect category paired with a review (or sentence) in which it is mentioned. Each `notion` has a textual unit which contains the text of the review (or sentence). Our training data contains 335 reviews with 1435 review-level `notion` instances and 2455 sentence-level `notion` instances. The test data contains 90 reviews with 404 review-level `notion` instances and 859 sentence-level `notion` instances.

```
<sentence id="1032695:1">
<text>Everything is always cooked to
    perfection, the service is excellent,
    the decor cool and understated.</text>
<Opinions>
<Opinion target="NULL" category="FOOD#
    QUALITY" polarity="positive" from="0"
    to="0"/>
<Opinion target="service" category="
    SERVICE#GENERAL" polarity="positive"
    from="47" to="54"/>
<Opinion target="decor" category="AMBIENCE
    #GENERAL" polarity="positive" from="73
    " to="78"/>
</Opinions>
</sentence>
```

Fig. 1 A snippet from the used dataset showing an annotated sentence from a restaurant review

Our main task is to determine aspect sentiment polarities at the review level. To compare the review-based and sentence aggregation approaches, we use data that is annotated for both reviews and sentences with respect to aspect sentiment. Each review (sentence) in the dataset is annotated with its occurring aspect categories, we do not differentiate between aspects and their categories, and the corresponding sentiment polarities. A snippet of the used dataset is given in Fig. 1, showing the first sentence of a review with various sentiment annotations.

Each review in the dataset can contain multiple aspect categories and each of these is labeled as positive, neutral, negative, or conflict. An aspect is labeled as 'conflict' in the case of conflicting opinions. Each aspect mentioned in the review has a unique sentiment; however, an aspect can be mentioned multiple times in a review with different sentiments. In this case, all different sentiments are taken into account to assign an appropriate label. The sentences in the dataset can also contain multiple aspect categories; however, contrary to the review level, the aspect categories are labeled as positive, neutral, or negative. At the sentence level, the dataset is not annotated with the conflict label. This is presumably due to the small size of a sentence which makes the appearance of conflicting opinions less likely.

In Figs. 2 and 3, we show some statistics related to the aspects and sentiments in our dataset annotated for reviews. Figure 2 shows the relative frequency of each aspect category in the data. Each review is assigned an overall sentiment label

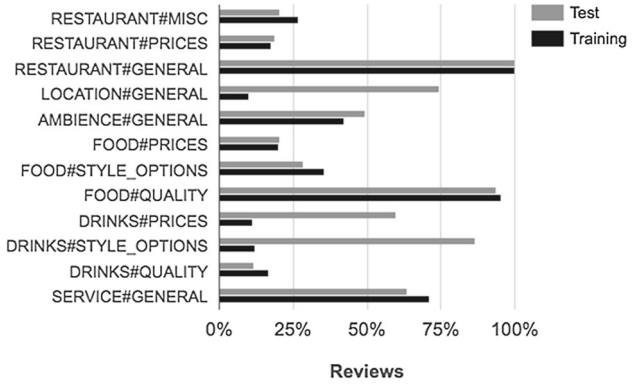

Fig. 2 Relative frequencies of each aspect category label

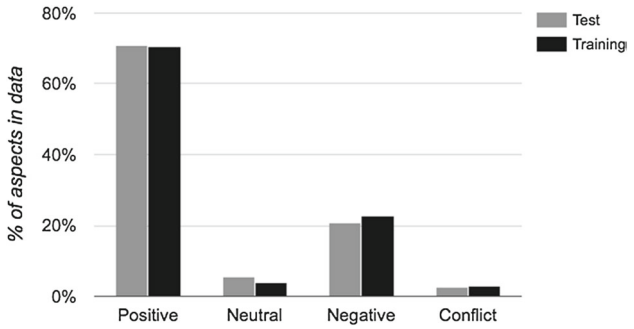

Fig. 3 Relative frequencies of each sentiment label

about the restaurant; therefore, the aspect category RESTAURANT#GENERAL has a frequency of 100%. Figure 3 shows the distribution of the sentiment values. We see that in both the training and test data the distribution of the sentiment values is unbalanced, as the positive label appears more frequently than the other sentiment labels.

4 Method

For the sentiment classification, we use a linear Support Vector Machine (SVM). For the review level, we train a multi-class SVM model with the classes: positive, negative, neutral, and conflict. For the sentence level, we train a multi-class SVM model with the classes: positive, negative, and neutral. We use an SVM model because it has shown good results for sentiment analysis in the past [15].

Before the available data can be used for aspect sentiment classification, it has to be preprocessed first. For this, we use the Stanford CoreNLP Natural Language Processing Toolkit [13]. The first step in the processing of the data is tokenization. By tokenizing the data, we can separate meaningful terms and characters in the text from each other and remove white spaces. After tokenization, Part-of-Speech tags, such as 'noun' and 'adjective,' are attached to the words of the sentences. In order to be able to recognize different forms of a word as the same, we lemmatize the words. This means that we find the dictionary form of a word. The last step is to parse the data, which determines the grammatical structure of sentences. This information can later be used to determine related words. Our proposed algorithms then use `notion` instances, an aspect category paired with a review (or sentence) in which it is mentioned, from this preprocessed data.

4.1 Ontology

The restaurant ontology[1] consists of three main classes: *Entity*, *Property*, and *Sentiment*. Our first main class is the *Entity* class. This class contains terms pertaining to the domain of restaurant reviews. Its subclasses are *Ambience, Experience, Location, Person, Price, Restaurant, Service, StyleOptions*, and *Sustenance*. Most of these classes represent one or more aspect categories, and in those cases, the *aspect* annotation connects that class to the corresponding aspect category as annotated in the dataset (e.g., FOOD#QUALITY). The mentioned subclasses of *Entity* further have their own subclasses dividing them into more specific aspect categories.

Property is our second main class, and it is divided into numerous subclasses containing descriptive terms that can be encountered in the restaurant domain. We created subclasses of terms that describe general negative and positive properties that can be related to several *Entity* subclasses (e.g., *GenericPositiveProperty*), and subclasses that represent adjectives describing characteristics of only one *Entity* subclass. For example, the class *AmbienceNegativeProperty* is a subclass of *Property* and a subclass of *Ambience*.

Our last main class is *Sentiment*. *Sentiment* is the superclass of the various polarity values, excluding conflict. This class has positive, negative, and neutral as subclasses. Subclasses of the *Property* class that represent positive (negative) properties are also a subclass of the *Positive* (*Negative*) class. Yet subclasses of the *Property* class representing adjectives that are more context specific are not subclasses of the different sentiment classes (e.g., *Cold*). Such classes, however, in combination with an entity (e.g., *WarmDrinks*), may be a subclass of one of the sentiment classes. This can be seen in Fig. 4. Here *HotTea* is a subclass of *WarmDrinks* and *Cold* is a subclass of *Property*. The intersection of *WarmDrinks* and *Cold* is a subclass of *Negative*, as warm drinks should not be cold. Thus, because *HotTea* is a subclass of *WarmDrinks*, the intersection of *HotTea* and *Cold* is a subclass of *Negative*.

The majority of our classes have the *lex* annotation attached. This annotation links the class to the associated lexicalizations. These lexicalizations can later be used to

[1] The used ontology can be downloaded at www.kimschouten.com/publications#sac2018.

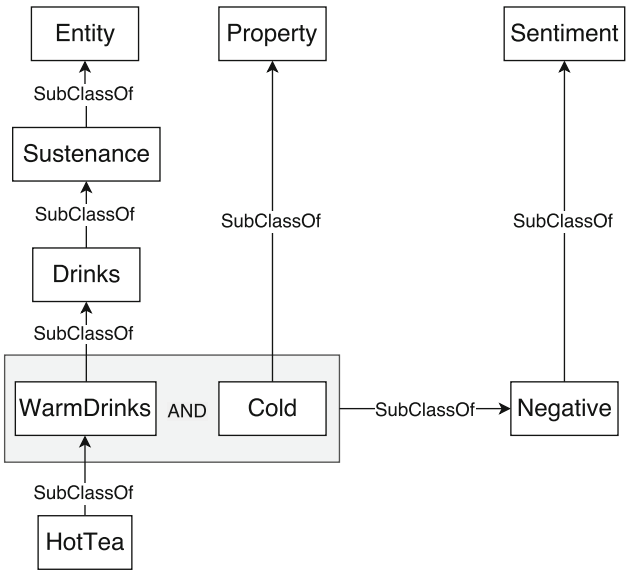

Fig. 4 Ontology snippet

search for the presence of a concept in the ontology. Because concepts are linked to words in the text, we use a so-called lexicalized ontology, combining the logical structure of an ontology with the characteristics of a lexicon.

Let us consider an example to illustrate the workings of our ontology: The word 'cramped' is related via the *lex* annotation to *Cramped* and *Cramped* is a subclass of *AmbienceNegativeProperty*. *AmbienceNegativeProperty* is a subclass of *Ambience*, and *Ambience* is related to the aspect AMBIENCE#GENERAL via the *aspect* annotation. Therefore, we know that the word 'cramped' refers to AMBIENCE#GENERAL. Furthermore, because *AmbienceNegativeProperty* is also a subclass of *Negative*, we know that 'cramped' expresses a negative sentiment about AMBIENCE #GENERAL. Therefore, if the word 'cramped' is found in the review, the aspect AMBIENCE#GENERAL is labeled negative. This can be formally represented as follows:

$$Cramped \equiv \exists lex.\{\text{“cramped”}\}$$

$$Cramped \sqsubseteq AmbienceNegativeProperty$$

$$AmbienceNegativeProperty \sqsubseteq Ambience$$

$$AmbienceNegativeProperty \sqsubseteq Negative$$

$$Ambience \sqsubseteq \exists aspect.\{\text{“AMBIENCE\#GENERAL”}\}$$

Thus 'cramped' is negative about AMBIENCE#GENERAL.

The ontology design process has been to first distinguish between Entities, which we use to model the aspects/nouns, Properties, which we use to model the sentiment expressions (i.e., adjectives), and Sentiment, which has just the Positive and Negative subclasses to indicate the sentiment value. The next step involves the domain, as we retrieve all the values for the aspect category field associated with each aspect. These are converted to ontology concepts and put in a hierarchy, where some aspect categories can be grouped into one ontology concept and others may be split. For the most frequently occurring aspect categories, we create sentiment-valued superclasses (e.g., AmbienceNegativeProperty), which are mainly convenience classes. We generate a list of frequently occurring nouns and adjectives and classify them manually in the ontology.

To extend our ontology, we use refined lists of terms pertaining to our domain and available on the Web and add these terms as subclasses of the corresponding classes in our ontology. A list of used URLs for ontology augmentation is given in Table 1.

For example, we expand our *AmbienceNegativeProperty* class with a list of negative adjectives describing ambience. We further augment our ontology by adding words that frequently occur within our training data and are relevant to the restaurant domain.

4.2 Algorithms

We distinguish between two different algorithms. The first algorithm is review based and uses a linear multi-class SVM model. The SVM determines the sentiment of the aspect categories in the review based on a feature vector. The aspect categories can be labeled as positive, neutral, negative, or

Table 1 Used URLs of online lists of terms for ontology augmentation

Ontology concept	URL
AmbienceNegativeProperty	https://quizlet.com/1627604/words-to-describe-negative-atmospheremood-flash-cards/
AmbiencePositiveProperty	https://quizlet.com/182339815/words-to-describe-postive-atmospheremood-flash-cards/
GenericNegativeProperty	https://quizlet.com/193330328/general-adjectives-negatives-flash-cards/
GenericPositiveProperty	https://quizlet.com/157842128/french-general-positive-adjectives-flash-cards/
ServiceNegativeProperty	https://quizlet.com/144266646/negative-adjectives-to-describe-people-flash-cards/
ServicePositiveProperty	https://quizlet.com/144267151/positive-adjectives-to-describe-people-flash-cards/
SustenanceNegativeProperty	http://www.macmillandictionary.com/thesaurus-category/british/tasting-bad-or-lacking-flavour
Meat	http://www.macmillandictionary.com/thesaurus-category/british/types-of-meat

conflict. Per `notion` we create a new feature vector instance. Our second algorithm is a sentence aggregation algorithm and a more refined method for the prediction of the aspect sentiments in reviews. We once again use a linear multi-class SVM, though now with the classes positive, negative, and neutral. Contrary to the review-based algorithm, we predict the sentiment of aspects in a single sentence instead of a review. Using these predictions, we use an aggregation step to sum up the predicted polarities of each aspect per sentence. This step is shown in Eq. 1, where $p_{a,r}$ is the expressed polarity of a given aspect a within a given review r, s is a sentence contained in review r, and $p_{a,s}$ is the computed polarity of aspect a in sentence s. Thus, if a review has for example five sentences, where in three of them the FOOD#QUALITY aspect appears, we sum up the predicted polarity of these three sentences. Note the difference between the neutral and conflicted cases.

$$p_{a,r} = \begin{cases} positive, & \text{if} \quad \sum_{s \in r} p_{a,s} > 0 \\ negative, & \text{if} \quad \sum_{s \in r} p_{a,s} < 0 \\ neutral, & \text{if} \quad \sum_{s \in r} abs(p_{a,s}) = 0 \\ conflicted, & \text{otherwise} \end{cases} \tag{1}$$

4.3 Model features

Our SVM models use a variety of features to determine the sentiment classification. We use several procedures to construct these features, which we can split into two groups: the *feature generators*, which each create one or more features, and the *feature adaptors*, which adjust existing features. Some of these are independent from the employed ontology, while others stem from the use of the ontology. In general, the bag model is used for features, so to encode the words in a review, we employ the bag-of-words model and encode the presence or absence of words with binary features. In a similar fashion, we encode the presence or absence of ontology concepts in the feature vector using binary features. We can optionally weight the presence of words and concepts using a TF-IDF score, in which case the features are of course no longer binary, although a zero still represents the absence of a feature.

We determine which features to include in our final models by comparing the average F_1 score for different model feature combinations using the training data with tenfold cross-validation. Furthermore, for each of the models, we optimize the SVM complexity parameter c over the range 10^{-6} to 10^{3} with steps of 10^{1}.

4.3.1 Feature generators

The following feature generators are **independent** from the ontology:

Aspect In the data, we have the aspects that are mentioned within each review (or sentence). Thus, for each `notion`, we use its corresponding aspect category as a feature in the SVM model, using dummy variables.

Sentence count This feature generator counts the number of sentences in a review-level `notion` and adds this value to the feature vector.

Lemma This feature generator keeps track of the occurrence of words within the textual unit of a `notion`. For this item, all words within the dataset are added to the SVM feature vector and the instance value is set equal to one if the word appears in the textual unit of the current `notion`, and zero otherwise (cf. bag-of-words model).

The following feature generators are **dependent** on the ontology:

Ontology concepts This feature generator is essentially a bag of concepts, where all ontology concepts are encoded as binary features in the feature vector. We inspect for each word in the textual unit of a `notion` whether it is a lexicalization of a concept in our ontology. If this is the case, we then find all superclasses of this class. If at least one of these superclasses is related to the current aspect category (e.g., SERVICE#GEN-ERAL) with the *aspect* annotation, we set all features that correspond to these superclasses in the feature vector to one. By adding all the superclasses, we can make use of implicitly stated information.

Sentiment count This feature generator counts the number of positive and negative text hits within the ontology. Thus, whenever one of the superclasses of a class associated to a word in the considered textual unit is the *Positive* or *Negative* class, we update the respective counter.

4.3.2 Feature adaptors

The following feature adaptors are **dependent** on the ontology:

Ontology concept score This feature adaptor influences the *ontology concepts* feature generator. We set the value for all superclasses to one, like in *ontology concepts*, however, superclasses that have the current category (e.g., SERVICE#GENERAL) as a value for the *aspect* annotation get a larger importance score. We denote this importance score with the parameter m, and we determine the value of this parameter using optimization. By assigning a larger value to the superclasses directly related to the current category, the SVM can take into account that these superclass features are more important than superclasses that are not directly related to the current category.

Negation handling It is common that reviewers make use of expressions such as 'not bad.' In order to account for this, we adapt the feature generator *ontology concepts*. If a word, that is a hit in our ontology, has a negating word directly

preceding it, we replace all positive (negative) superclasses associated with the word with their negative (positive) counterpart. In this way when expressions such as 'not bad' are used, we correctly specify it as a positive expression rather than a negative one.

Synonyms Since our ontology adds useful information to our SVM feature vectors, we want to increase the number of relevant words that occur as lexical representations of concepts in our ontology. For this, we use synonyms from WordNet [14] to complement the feature generator *ontology concepts*. If a word in the textual unit does not appear as a lexicalization in our ontology, we check if one of its synonyms is included. If this is the case, we add the superclasses associated with the synonym that does occur in the ontology to the feature vector. Word-sense disambiguation, identifying the meaning of a word within its context, is included in the design of the *synonyms* feature adaptor. Since only words that correspond to the restaurant domain-specific meaning are included in our ontology, synonyms that do not relate to the restaurant domain are ignored. For example, the word 'starter' may appear in the textual unit of a `notion`; however, 'starter' does not appear as a lexicalization in the ontology. We thus consider the synonyms of 'starter.' The word 'starter' has, among others, the synonyms 'newcomer' and 'appetizer,' yet only 'appetizer' appears in our ontology. Therefore, we select only the set of synonyms which contain at least one concept that is already in the ontology. For this to work, we assume that a word is used with only one meaning (the domain related one) in our domain text.

Weight In order to take into account that some words are less important than others, we adjust *ontology concepts* generator by determining weight scores for every word that appears in the data. In the calculation of the weight scores, we take into account that words that frequently appear in a review, but also frequently appear in all reviews, are less important than words that do not frequently appear in all reviews. This is called term frequency-inverse document frequency (TF-IDF). We use the following formula to determine the weight score of a word [19]:

$$weightScore = tf_{t,r} \cdot \log \frac{N}{df_t}, \tag{2}$$

where $tf_{t,r}$ is the frequency of the term t in the current review r, N is the total number of reviews, and df_t is the number of reviews in which the term t appears. We take the natural logarithm because words that appear ten times more often are not necessarily ten times more important. Thus, the logarithm helps to scale down the importance of the term. When the *weight* property is applied, the instance value of each superclass in *ontology concepts* is replaced by:

$$\max_i w_{s,i}, \tag{3}$$

where $w_{s,i}$ is the weight of word i, which is a lexicalization of a class in the ontology that has s as one of its superclasses. We take the maximum as we do not want a superclass to count more heavily when it appears more frequently. When *weight* is applied in combination with the *Ontology concept score* feature adaptor, the instance value of each superclass is set equal to:

$$\sum_i w_{s,i}, \tag{4}$$

where $w_{s,i}$ is as described above. However, when a superclass has the current aspect category as the value for the *aspect* annotation, the instance value described in Eq. 4 is multiplied by the importance score m. In this case, we take the summation of the weight scores because if multiple ontology concepts are related to the current aspect category, we want the ontology concept that appears more often to have a larger score.

Word window Rather than using the whole textual unit of a `notion` to create features, we determine a set of word windows. The pseudocode describing this step can be found in Algorithm 1.

We initially iterate over all the words in the textual unit and when a word (or a synonym of it, when applied in combination with the feature adaptor *Synonyms*) appears as a lexicalization in our ontology, we determine a window of related words that are at most *k+1* grammatical steps away from the original word. We determine these grammatical steps using the dependencies found during the preprocessing of the data. The value of k is optimized. We then use the word windows to create the features related to that `notion`. To illustrate the concept of a word window, consider the word 'prices' which appears in the sentence 'Prices too high for this cramped and unappealing restaurant.' The word window surrounding 'prices' is [Prices, too, high, restaurant, .], where we, for this example, assume that k is equal to one.

Parameter optimization

A single run of tenfold cross-validation is used to determine the optimal value for the complexity parameter c of the SVM model, the importance score m in *Ontology concept score* and the parameter k for the *Word window* feature adaptor. The F_1 score is calculated for each set of parameter values in order to determine the optimal values. For the feature adaptor *Ontology concept score*, the optimal importance score m is found within a range of 1.0–10.0 using steps of one. For the feature adapter *Word Window*, the k parameter is optimized over a range of 1–5 with steps of one. Last, the c parameter of the SVM model is optimized by testing values in the range 10^{-6} to 10^3 while iteratively increasing the exponent by one.

Algorithm 1: Word Window Algorithm

Let r be a review (or sentence) and a be an aspect (e.g., FOOD#QUALITY). Then we define $N_{r,a}$ to be the textual unit of a `notion` for a review (or sentence) r in combination with the aspect a. Furthermore, we define $W_{r,a}$ to be the set of word windows for the review (or sentence) r in combination with the aspect a, and w to be a word. Moreover, Syn_w is a set of synonyms for word w, and y is a synonym. Lastly, $getDepWindow(scope, k)$ is a function that returns the word window surrounding the $scope$ and where parameter k represents the maximum grammatical distance between words.

Data: $N_{r,a}$ textual unit of a `notion` for review (or sentence) r and aspect a, $Synonyms$ boolean that is true if applying $Synonyms$, k the window parameter

Result: $W_{r,a}$ set of word windows for review (or sentence) r and aspect a

```
begin
    hit ⟵ false
    for w ∈ N_{r,a} do
        scope ⟵ w
        if inOntology(w) then
            hit ⟵ true
        else if Synonyms then
            Syn_w ⟵ getSynonyms(w)
            for y ∈ Syn_w do
                if inOntology(y) then
                    hit ⟵ true
                    scope ⟵ y
        if hit then
            scope ⟵ getDepWindow(scope, k)
            W_{r,a} ∪ scope
    return W_{r,a}
```

Table 2 shows the average F_1 scores for the *ontSL* model, which is our ontology-based sentence-level model, for the different combinations of m, k, and c. We can see that the average F_1 score is the highest for $m = 5.0$, $k = 1$, and $c = 0.1$.

4.4 Model evaluation

To evaluate our models, we calculate the accuracy, which is equal to the F_1 score. When an instance is correctly predicted, we define this as true positive. False positives and false negatives are both found when the predicted sentiment value is incorrect. Last, to compare models with each other, we use a two-sample, two-tailed paired t test.

5 Evaluation

In this section, we present and discuss our results. We hypothesize that the use of an ontology results in an improved sentiment prediction accuracy. For this, we describe and compare the predictive capabilities of our models. We also expect that the review-based algorithm outperforms the sentence aggregation algorithm. For each model, we present the average value and standard deviation of the F_1-measure over five runs of tenfold cross-validation using only the training data. Furthermore, we report the p values of the two-sided paired t tests comparing the predictive abilities of the different models. Finally, the right half of the table contains the F_1 scores for a single run of the models using the test data. Furthermore, we hypothesize that the addition of an ontology to aspect-based sentiment analysis lessens the need for training data, and thus decreases the data size sensitivity. To analyze this we run experiments with differing training data sizes. Last, we take a look at which features are most important to our models.

Table 2 Parameter analysis for the *ontSL* model

m	k				
	1	2	3	4	5
1.0	0.8182	0.8193	0.8114	0.8138	0.8188
2.0	0.8141	0.8192	0.8129*	0.8096	0.8194
3.0	0.8220	0.8276	0.8123	0.8178	0.8112
4.0	0.8295	0.8115	0.8111	0.8121	0.8141
5.0	**0.8320**	0.8140	0.8141	0.8118	0.8145*
6.0	0.8199	0.8154	0.8128	0.8202	0.8094
7.0	0.8236	0.8249	0.8178	0.8148	0.8211
8.0	0.8221	0.8140	0.8068	0.8144	0.8102
9.0	0.8153	0.8195	0.8141	0.8173	0.8146
10.0	0.8242	0.8162	0.8159	0.8222	0.8244

*Optimal $c = 0.01$, else optimal $c = 0.1$
Bold indicates the highest value

5.1 Performance

The first experiment we want to perform is to compare the performance of the baseline method with the ontology-enhanced version for the sentence-level task. In this task, the aspects are annotated per sentence instead of per review, so there is a prediction per sentence. While this is not the focus of this paper, the sentence aggregation method that does review-level sentiment analysis actually depends on the predictions made by the sentence-level version of the proposed algorithm. Hence, it is informative to see how this method fares on this task and whether the ontology features are useful. The results of this experiment are shown in Table 3. We compare our sentence-level aspect-based sentiment analysis model, *ontSL*, to a baseline model, *baseSL*. The *baseSL* model is a combination of the feature generators *aspect* and *lemma*. The *ontSL* model is a combination of the feature generators *aspect*, *lemma*, *ontology concepts*, and *sentiment count*, and the feature adaptors *ontology concept score, negation handling, synonyms, weight*, and *word window*, all applied to the

Table 3 Model performance for the aspect sentiment classification at the sentence level

	Tenfold cross-valid.		p value	Training data	Test data
	Avg. F_1	SD		F_1	F_1
baseSL	0.7008	0.0513	–	0.8436	0.7229
ontSL	0.8217	0.0500	<0.0001	0.8811	0.7963

sentence level. The optimal value of the importance score for the *ontology concept score* feature adaptor is equal to $m = 5.0$, and the optimal value of the parameter k for the *word window* feature adaptor is $k = 1$. The *baseSL* model and the *ontSL* model have an optimal complexity parameter of $c = 1$ and $c = 0.1$, respectively.

The next step is to measure how well the baseline and ontology-enhanced method perform on the review-level task when we apply the sentence aggregation step. In Table 4, we show the model performance for review-level aspect sentiment classification using the sentence aggregation algorithm. The *baseSA* and *ontSA* models in this table, respectively, use the predictions of the *baseSL* and *ontSL* model in Table 3, in combination with the sentence aggregation algorithm. Similarly to the previous results, the model that includes the use of an ontology has significantly better performance than the baseline. The gold value mentioned in the table is an upper bound on the F_1 score for the sentence aggregation algorithm. Instead of summing up the predicted sentiment polarities with the sentence summation algorithm, we aggregate the real sentiment polarities of the aspects in sentences, as present in the annotated data.

In Table 5, we compare our *final* review-level aspect-based sentiment analysis model to a *base*line model. Our *base* model is a combination of the feature generators *aspect*, *sentence count*, and *lemma*. While the *final* model uses the feature generators *aspect*, *sentence count*, *lemma*, *ontology concepts*, and *sentiment count*, and the feature adaptors *negation handling*, *synonyms*, and *weight*. For both the *base* and the *final* models, we find the optimized complexity parameter to be $c = 0.1$.

Table 5 shows that the model that includes the use of an ontology has significantly better performance than the base case for the task of aspect sentiment classification at the review-level. This result can be seen in both the tenfold cross-validation using the training data, and in the single run using both the training and test data. The model with the ontology displays an increase in accuracy of approximately 1.0% points in comparison with the baseline.

As the performance on the test set can be used to compare between tables, the results show that the review-based algorithm outperforms the sentence aggregation algorithm for the test data. The *final* model has an accuracy that is approximately 4.0% points higher than the accuracy of the *ontSA* model. Table 6 shows the performance of our *final* model compared to SemEval submissions for this task [17]. The purpose of this table is to give some context for the reported performance measures, not to directly compare the performance of our method against the SemEval participators. In the table, it is indicated whether the submission was constrained (C), i.e., it used only the training data, or unconstrained (U), i.e., it used also additional resources. The accuracy of our *final* model differs less than 1% point from the top two submissions.

Table 4 Model performance for the aspect sentiment classification using the sentence aggregation algorithm

	Tenfold cross-valid.		p value	Test data
	Avg. F_1	SD		F_1
baseSA	0.6897	0.0616	–	0.6824
ontSA	0.8130	0.0512	<0.0001	0.7717
Gold value (upper bound)				0.9633

5.2 Data size sensitivity

Since we hypothesize that less training data is needed when we include an ontology in our model, we perform an experiment by training our *base* and *final* SVM models with a decreasing proportion of training data. The test dataset, however, remains the same for each run. In this way, we can compare the F_1 scores over all the runs. We obtain the average F_1 scores over 10 single runs with randomly generated seeds. The result, shown in Fig. 5, is a mapping between the prediction performance and the proportion of training data.

The figure shows that the gap between the two lines remains approximately the same. Therefore, the use of an ontology does not influence the data size sensitivity. This is in line with the results reported in [20]. However, the ontology-enhanced method does, on average, perform better at all proportions of the training data when compared to the *base* model.

5.3 SVM model comparison

For this work, two alternative SVM models were considered next to a multi-class linear SVM: a multi-class SVM with an RBF kernel and a linear binary SVM. Both alternatives were compared with the linear multi-class SVM model. The *final* selection of features is used for all three models.

Table 5 Model performance for the aspect sentiment classification at the review level. *P* value is given for a paired two-sided *t* test

	Tenfold cross-valid.		*p* value versus base	Training set	Test set
	Avg. F_1	SD		F_1	F_1
Base	0.7852	0.0524	–	0.8718	0.8020
Final	0.8001	0.0506	<0.0001	0.8753	0.8119

Table 6 Ranking of proposed aspect sentiment classification methods SemEval-2016

Team	(Un)Constrained	Accuracy
UWB	Unconstrained	0.8193
ECNU	Unconstrained	0.8144
final	Unconstrained	0.8119
UWB	Constrained	0.8094
ECNU	Constrained	0.7871
bunji	Unconstrained	0.7055
bunji	Constrained	0.6658
GTI	Unconstrained	0.6411

Bolditalic indicates the presented method in a list of related works

Table 7 Comparing the linear kernel to the RBF kernel

	Avg. F_1	SD	Training data F_1	Test data F_1
Linear kernel	0.8001	0.0506	0.8753	0.8119
RBF kernel	0.7987	0.0473	0.8676	0.7921

Using a linear kernel, the optimal c is found to be 0.1. For the RBF kernel, the optimal combination of c and γ has to be determined. For γ, a range of 10^{-3} to 10^{1} is used where the exponent is increased by one in each iteration. The c parameter is optimized in conjunction using the same range as before. The optimal combination of c and γ for the *final* model is $c = 10.0$ and $\gamma = 0.001$. Table 7 shows that the SVM model using the linear kernel has better accuracy.

Table 8 Comparing binary classifier models to the multi-class model

Neutral	Conflict	Training data	Test data
Predicted as		F_1	F_1
Negative	Negative	0.9080	0.7401
Positive	Negative	0.9136	0.7401
Negative	Positive	0.9171	0.7401
Positive	Positive	0.9185	0.7451
Multi-class		0.8753	0.8119

The second alternative model, the binary classifier, only predicts the classes positive and negative. The `notions` with sentiment label neutral or conflict were set to negative or positive. Table 8 shows that the multi-class model has a better accuracy than the binary models.

5.4 Feature analysis

In Table 9, we list the ten most important features of our *final* model. The values listed represent the Information Gain (IG) [18]. The term reported next to the Information Gain states from which generator this feature originates, and next to that the name of the feature is given. For example, the first feature *numNegative* is generated by *sentiment count*, which counts the number of negative concept hits in the ontology.

As can be seen in Table 9, the majority of the most important SVM features are related to the negative sentiment class.

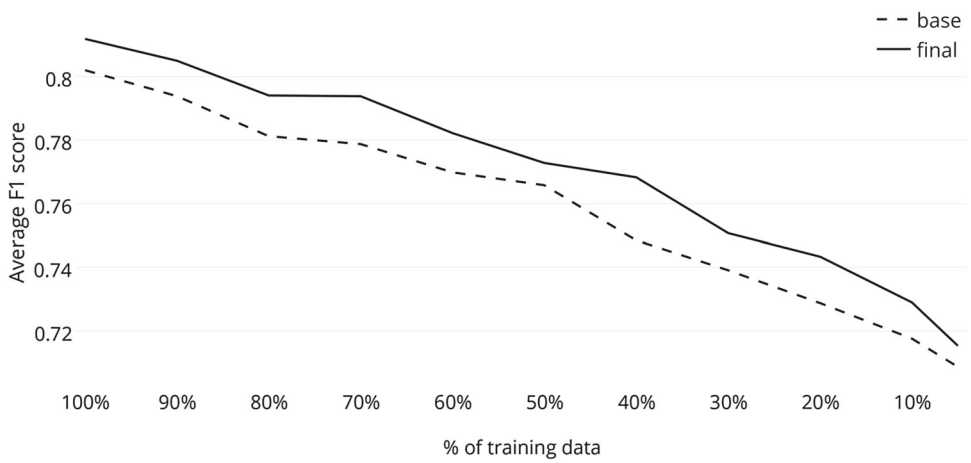

Fig. 5 Data size sensitivity analysis of an ontology-enhanced (*final*) and a non-ontology-enhanced (*base*) approaches

Table 9 Top 10 most important features for the *final* review-based model (*ontology concepts* is abbreviated to ontology)

0.2381	Sentiment count: numNegative
0.1159	Ontology: Negative
0.0821	Lemma: 'not'
0.0638	Ontology: SustenanceNegativeProperty
0.0557	Sentiment count: numPositive
0.0539	Ontology: ServiceNegativeProperty
0.0522	Lemma: 'do'
0.0517	Sentence count: numSentences
0.0515	Ontology: GenericNegativeProperty
0.0443	Lemma: 'horrible'

Table 10 Feature analysis

	Tenfold cross-valid		Training data	Test data
	Avg. F_1	SD	F_1	F_1
base	0.7852	0.0524	0.8718	0.8020
+Weight	0.7969	0.0446	0.8808	0.8020
+Sentiment count	0.7988	0.0459	0.8808	0.8045
+Negation handling	0.7992	0.0459	0.8808	0.8045
final	0.8001	0.0506	0.8753	0.8119

As most of the `notions` within the dataset are labeled as positive, features that expose the negativity of a textual unit are important. Furthermore, we also calculate the internal attribute weights of the *final* SVM model. The 80 features with the largest weight are all ontology-related features such as *Negative*, *Boring* and *Cozy*. This emphasizes the added value of an ontology.

We also look at the influence of our various feature generators and adaptors. As the **weight** feature influences for instance the **ontology concepts** feature, the latter is not separately mentioned. Table 10 shows the results of a stepwise approach where the F_1 is measured each time a feature type is added to the model. Note that for every step, the average F_1, as measured over the tenfold cross-validation results, increases, showing the benefit of each of the selected feature types.

6 Conclusion

In this paper, we investigated the added value of an ontology to the task of review-level aspect-based sentiment analysis. We proposed two different algorithms, a review-based algorithm and an aggregated sentence-based algorithm, which we both enhanced with the use of an ontology. For both algorithms, the accuracy is significantly higher when the ontology is used. The importance of the ontology is also supported by the results of the performed feature analysis. When comparing the two algorithms, we observe that, as expected, the review-based algorithm gives better results than the aggregated sentence-based algorithm. Furthermore, we hypothesized that the inclusion of an ontology would lessen the need for training data. However, contrary to our expectations, this is not the case. The ontology incorporating method is not less sensitive to the size of the training data than the method that does not incorporate an ontology, which is in line with earlier results [20]. A reason for this might be that the ontology features are not robust, meaning that the model needs training data to learn how to interpret them.

Since building an ontology manually is a time-consuming and labor-intensive task, we would suggest to look into automating the process of creating the ontology for future work, for example, by extracting information from text [1]. This could make the use of an ontology more efficient when considering multiple domains. However, when combining multiple domains in a single ontology, a refined word-sense disambiguation method is needed to determine the domain and decide the sense of a word given that domain.

Furthermore, in our paper, we only assign the polarity values positive, neutral, negative, and conflict. However, an opinion can often not be categorized by merely four sentiment polarities. To account for this, sentiment scores could be assigned in order to extract the strength of an opinion. Given the current trend in using deep neural networks, it would be interesting to see how methods, such as attention-based LSTMs [23] or dyadic memory networks [22], can be combined with the external knowledge and reasoning from ontologies.

References

1. Buitelaar, P., Cimiano, P., Magnini, B.: Ontology Learning from Text: Methods. Evaluation and Applications. IOS Press, Amsterdam (2005)
2. Cambria, E.: Affective computing and sentiment analysis. IEEE Intell. Syst. **31**(2), 102–107 (2016)
3. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(3), 27:1–27:27 (2011)
4. de Heij, D., Troyanovsky, A., Yang, C., Scharff, M.Z., Schouten, K., Frasincar, F.: An ontology-enhanced hybrid approach to aspect-based sentiment analysis. In: Proceedings of the 18th International Conference on Web Information Systems Engineering (WISE 2017), pp. 338–345. Springer (2017)

5. de Kok, S., Punt, L., van den Puttelaar, R., Ranta, K., Schouten, K., Frasincar, F.: Review-level aspect-based sentiment analysis using an ontology. In: Proceedings of the 33rd Symposium on Applied Computing (SAC 2018), pp. 315-322. ACM (2018)
6. Decker, S., Erdmann, M., Fensel, D., Studer, R.: Ontobroker: ontology based access to distributed and semi-structured information. In: Meersman, R., Tari, Z., Stevens, S. (eds.) Database Semantics: Semantic Issues in Multimedia Systems, pp. 351–369. Springer, New York (1999)
7. Feldman, R.: Techniques and applications for sentiment analysis. Commun. ACM **56**(4), 82–89 (2013)
8. Horrigan, J.A.: Online shopping. Pew Internet Am. Life Proj. Rep. **36**, 1–24 (2008)
9. Hu, M., Liu, B.: Mining opinion features in customer reviews. In: Proceedings of the 19th National Conference on Artificial Intelligence (AAAI 2004), vol. 4, pp. 755–760. AAAI (2004)
10. Internet World Stats. (WWW Document). www.internetworldstats.com/stats.htm. Accessed 31 Aug 2017
11. Lau, R.Y., Lai, C.C., Ma, J., Li, Y.: Automatic domain ontology extraction for context-sensitive opinion mining. In: Proceedings of the 30th International Conference on Information Systems (ICIS 2009), pp. 35–53. AIS Electronic Library (2009)
12. Liu, B.: Sentiment Analysis and Opinion Mining, Synthesis Lectures on Human Language Technologies, vol. 16. Morgan & Claypool, New York (2012)
13. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55–60. ACL (2014)
14. Miller, G., Beckwith, R., Felbaum, C., Gross, D., Miller, K.: Introduction to WordNet: an on-line lexical database. Int. J. Lexicogr. **3**(4), 235–312 (1990)
15. Mullen, T., Collier, N.: Sentiment analysis using support vector machines with diverse information sources. In: Proceedings of the 9th International Conference in Empirical Methods on Natural Language Processing (EMNLP 2004), vol. 4, pp. 412–418. ACL (2004)
16. Polpinij, J., Ghose, A.K.: An ontology-based sentiment classification methodology for online consumer reviews. In: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI 2008), vol. 1, pp. 518–524. IEEE (2008)
17. Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., Manandhar, S.: Semeval-2016 Task 5: aspect based sentiment analysis. In: Proceedings of the Tenth International Workshop on Semantic Evaluation (SemEval 2016), pp. 27–35. ACL (2016)
18. Roobaert, D., Karakoulas, G., Chawla, N.V.: Information gain, correlation and support vector machines. In: Guyon, I., Nikravesh, M., Gunn, S., Zadeh, L.A. (eds.) Feature Extraction: Foundations and Applications, pp. 463–470. Springer, Berlin (2006)
19. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, New York (1983)
20. Schouten, K., Frasincar, F., de Jong, F.: Ontology-enhanced aspect-based sentiment analysis. In: Proceedings of the 17th International Conference on Web Engineering (ICWE 2017), vol. 10360, pp. 302–3320. Springer (2017)
21. Schouten, K., Frasincar, F.: Survey on aspect-level sentiment analysis. IEEE Trans. Knowl. Data Eng. **28**(3), 813–830 (2016)
22. Tay, Y., Tuan, L.A., Hui, S.C.: Dyadic memory networks for aspect-based sentiment analysis. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM 2017), pp. 107–116. ACM (2017)
23. Wang, Y., Huang, M., Zhu, X., Zhao, L.: Attention-based lstm for aspect-level sentiment classification. In: Proceedings of the 2016 Conference on Empirical Methods on Natural Language Processing (EMNLP 2016), pp. 606–615. ACL (2016)
24. Wei, W., Gulla, J.A.: Sentiment learning on product reviews via sentiment ontology tree. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010), pp. 404–413. ACL (2010)