DCWEB-SOBA: Deep Contextual Word Embeddings-Based Semi-automatic Ontology Building for Aspect-Based Sentiment Classification

Roos van Lookeren Campagne, David van Ommen, Mark Rademaker,
Tom Teurlings, and Flavius Frasincar(✉)

Erasmus University Rotterdam, PO Box 1738, 3000 DR Rotterdam, The Netherlands
{471085rl,483808do,467978mr,482163tt}@student.eur.nl,
frasincar@ese.eur.nl

Abstract. In this paper, we propose the use of deep contextualised word embeddings to semi-automatically build a domain sentiment ontology. Compared to previous research, we use deep contextualised word embeddings to better cope with various meanings of words. A state-of-the-art hybrid method is used for aspect-based sentiment analysis, called HAABSA++, to evaluate our obtained ontology on the SemEval-2016 restaurant dataset. We achieve a prediction accuracy of 81.85% for the hybrid model with our ontology, which outperforms the hybrid model with other considered ontologies. Furthermore, we find that the ontology obtained from our proposed domain sentiment ontology builder, called DCWEB-SOBA, on itself improves the accuracy for the conclusive cases from 83.04% to 84.52% compared to the ontology builder based on non-contextual word embeddings, WEB-SOBA.

Keywords: Ontology learning · Contextual word embeddings · Aspect-based sentiment analysis · Hybrid method

1 Introduction

Aspect-based sentiment analysis (ABSA) determines the sentiment relating to the aspects (features) of products or services in (Web) text [15]. ABSA provides businesses more insight into which specific aspects of a product or service need to be improved upon [17].

ABSA involves two steps. First, the aspects are identified and categorised (aspect detection), thereafter the sentiments for the identified aspects are gauged (sentiment analysis) [17]. In this paper we perform these steps with state-of-the-art hybrid models for sentiment classification, using benchmark data in which the aspects are given. These models use a domain sentiment ontology to predict sentiment. Whenever the ontology reasoner is unable to predict the sentiment, a neural attention model serves as a backup solution [16,19,20].

P. Groth et al. (Eds.): ESWC 2022, LNCS 13261, pp. 183–199, 2022.
https://doi.org/10.1007/978-3-031-06981-9_11

There are various methods to obtain a domain sentiment ontology. The most general method is to manually build one [16,17]. Although this method has good performance, lexicalisations are limited and building the ontology is time-consuming. Additionally, the ontology is manually constructed for a specific domain and therefore hard to transfer to another domain. Automatically constructed ontologies, suggested by [2], shorten building time, but unsupervised building results in less accuracy. [22] proposes to semi-automatically build a domain sentiment ontology from a domain-specific corpus, which could produce more extensive ontologies in a time-efficient manner. For these ontologies, user input is required to control the builder on mistakes. [9] extends the work proposed in [22] by using non-contextual word embeddings instead of word co-occurrence for ontology building and achieves a higher accuracy in a hybrid model.

The objective of this research is to improve the performance of the sentiment classification by semi-automatically building a domain sentiment ontology from a domain-specific sentiment corpus based on deep contextual word embeddings. These deep contextual word embeddings allow the model to cope with polysemy, which is not considered by non-contextual word embeddings. Two forms of deep contextual word embeddings are BERT [5] and ELMo [13]. Compared to ELMo, BERT is easily applicable for a wide range of Natural Language Processing (NLP) tasks and can consider both left and right contexts simultaneously, while ELMo can only take the concatenation of the left and right context representations. Unlike other research on contextual word embeddings, our approach can serve as a refined way of aspect-based sentiment analysis by accounting for word meanings in context.

We adopt the ontology structure posed in [9] so that our work contributes to previous research in four ways. First, we use deep contextualised word embeddings obtained from BERT to deal with polysemy. Second, our ontology builder considers adverbs to carry a sentiment. For example, ‘carefully’ conveys a positive sentiment in the context of ‘carefully prepared’. Additionally, we use an extra set of BERT word embeddings that are sentiment-aware instead of using external data sources to make word embeddings sentiment-aware. Last, we select relevant words for the domain sentiment ontology using a novel threshold function. These four extensions will allow the ontology to be built time-efficiently without the cost of losing accuracy. We use the ontology obtained from our ontology builder in a state-of-the-art hybrid approach for aspect-based sentiment analysis (HAABSA++) [19], using LCR-Rot-hop++ as a backup model to obtain a sentiment when our ontology is inconclusive.

This paper is structured as follows. In Sect. 2, we briefly discuss some related work, followed by an overview of the used datasets in Sect. 3. In Sect. 4 first, the structure of the ontology is explained, followed by an explanation of our proposed approach for building a domain sentiment ontology. The performance of our proposed solution is evaluated in Sect. 5. Last, in Sect. 6, we draw our conclusions and provide suggestions for future research.The source code is available from: https://github.com/RoosVanLookeren/DCWEB-SOBA.

2 Related Work

In this section, we discuss relevant literature to our work. First, we look into hybrid approaches for ABSA in Sect. 2.1. Second, different domain sentiment ontology builders are outlined in Sect. 2.2. Last, in Sect. 2.3, we briefly describe various deep contextual word embeddings.

2.1 Hybrid Models

The research of [16] is among the first showing that a hybrid approach to aspect-based sentiment analysis algorithms outperforms other state-of-the-art approaches. As input for sentiment prediction, [16] proposes to manually construct a domain sentiment ontology. In case the ontology does not specify a sentiment value for an aspect a Bag-of-Words model trained with a multi-class Support Vector Machine (SVM) is used to complement the ontology. This work started more research for optimising the machine learning-based backup models.

[20] introduces a hybrid approach for aspect-based sentiment analysis, called HAABSA, using the domain sentiment ontology of [16] and a Left-Center-Right neural network with an iterative rotatory attention mechanism (LCR-Rot-hop) as a backup method. By iterating the rotary attention mechanism multiple times, the model focuses better on the relevant sentiment words related to a given aspect. This results in a better accuracy as the interaction between aspect and relevant context is better captured.

Following the HAABSA model, [19] introduces HAABSA++, which extends HAABSA in two directions. First, non-contextual GloVe word embeddings are replaced by deep contextual word embeddings, BERT [5] and ELMo [13], to deal with word semantics in the text. Second, the authors introduce hierarchical attention by adding an extra attention layer to the attention mechanism of [20], enabling the model to distinguish the importance of the high-level input sentence representations. The adjustment of the rotatory attention mechanism to a hierarchical architecture is called LCR-Rot-hop++. This paper shows that exploiting BERT word embeddings in LCR-Rot-hop++ results in better performance compared to ELMo word embeddings and outperforms LCR-Rot-hop.

In our paper we introduce the next evolution of hybrid models by using deep contextual word-embeddings to build the domain sentiment ontology. We implement our obtained domain sentiment ontology in HAABSA++ using LCR-Rot-hop++ as a backup model, due to the proven success of this method [19].

2.2 Ontology Building Approaches

Besides improvements of the backup model in hybrid approaches, research has been done in refining the first part of hybrid models, more precisely the ontology. Ideally an ontology is built in a time-efficient manner and is as accurate as possible for the considered task. However, accuracy and time-efficiency do not always go hand in hand. Therefore a good ontology needs to be built time-efficiently while remaining able to predict the sentiments accurately.

One solution for building ontologies is to integrate existing knowledge resources as proposed by OntoSenticNet 2 [6]. While such an approach is able to model commonsense sentiment, it provides the `Domain` concept that can be used to extend the ontology with domain sentiment representations. For building domain sentiment representations, often a data-driven approach is used due to the availability of domain text, which is also the approach pursued here.

To decrease the building time of a domain sentiment ontology, [4,22] suggest to build a domain sentiment ontology semi-automatically, which requires manual input of the user to control for possible mistakes made by the ontology builder. [22] proposes SOBA, in which word co-occurrence is used to deal with word semantics. Besides word co-occurrence, [4] also employs synsets in an ontology builder, called SASOBUS, to enable a fair and reliable comparison of words, while simultaneously capturing their meaning. These two methods give comparable results to the manually constructed ontology, yet a significant reduction in constructing time is achieved.

Better performance is attained when using non-contextual word embeddings for the automated part of the domain sentiment ontology building. In this method, named WEB-SOBA [9], word2vec [12] word embeddings are exploited. The CBOW model is used, which can detect syntactic and semantic word similarities. When the ontology obtained from WEB-SOBA is used in HAABSA, it performs better than the manually constructed domain sentiment ontology. WEB-SOBA reduces both computing and user time required to construct the ontology compared to SOBA and SASOBUS, given that the non-contextual word embeddings for a specific domain are already made. For this research, we use the manually constructed ontology, SOBA, and WEB-SOBA as benchmarks. We do not compare our ontology with SASOBUS as [9] already showed that WEB-SOBA outperforms SASOBUS on accuracy and time-efficiency.

2.3 Deep Contextual Word Embeddings

Word embeddings are vector word representations. In this paper we make a clear distinction between non-contextual word embeddings and deep contextual word embeddings. The difference is that deep contextual word embeddings will include the context of the word in the representation, while non-contextual word embeddings will not. This means that while non-contextual word embeddings (e.g., word2vec or GloVe) have difficulty coping with polysemy, deep contextual word embeddings (e.g., ELMo or BERT) can capture different meanings of a single word, as they assign a unique vector per instance of a word in its context.

Recently, two new game-changing types of machine-learning models that create deep contextual word embeddings were introduced. These are deep contextual word embeddings, in the sense that they are a function of many internal layers of the neural model. The first model is ELMo, the model captures information about the entire input sentence using multiple bidirectional Long Short-Term Memory (LSTM) layers. A big advantage of the second model, BERT [5], is that it can pre-train deep bidirectional vector representations from an unlabelled text by considering both left and right context simultaneously, while

ELMo can only take the concatenation of the separate contexts-based representations. BERT is an open-sourced model and a version of the pre-trained model trained on a massive dataset is publicly available. Next to that, BERT can efficiently be fine-tuned for a wide range of tasks with only one additional output layer, this makes BERT applicable in many situations. For our research, we therefore propose to use BERT to construct deep contextual word embeddings.

3 Data

To build a domain sentiment ontology we need a domain-specific corpus to determine the domain-specific terms. Earlier built ontologies use a restaurant domain dataset as domain-specific corpus. We opt to use such a dataset as well since it simplifies the comparison of our ontology with other ontologies. The dataset we use is the Yelp Open Dataset, which consists of 8,635,403 reviews of different businesses [21]. After filtering out the restaurant reviews, around 4,700,000 reviews of more than 500,000 restaurants remain in the dataset. We use the BERT base model (uncased), which is trained on the BooksCorpus (800M words) and Wikipedia (2500M words) [5] to create pre-trained word embeddings for 2000 reviews containing 200.000 unique words due to computational limitations.

When a sentiment word is negated in its context (e.g., 'not'), the polarity of this word embedding is reversed. To ensure that the word embedding is only determined by the correct polarity of the word, we remove the sentences in which a word from the negation set appears. The set of negation words used is $\mathcal{NW}$ = {'not', 'never', 'nothing', 'don't', 'doesn't', 'didn't', 'can't', 'wouldn't'}. In 10.4% of the sentences of the dataset one of these negation words is present. Consequently, we use 89.6% of the review sentences to create our ontology.

To evaluate the ontology we use the standard dataset from SemEval-2016 Task 5 [14], containing restaurant reviews. The data is divided into a training set of 1879 sentences and a test set of 650 sentences. Every sentence contains opinions regarding specific aspects. Each aspect is labelled with a polarity from the set $\mathcal{P} = \{\text{negative, neutral, positive}\}$. Furthermore, an entity (category) E (e.g., restaurant, food, service) and attribute A (e.g., prices, quality, general) pair is attached to each aspect. All implicit aspects are not used in the analysis because the machine learning method assumes aspects to be explicitly present.

4 Methodology

In this section, we explain the methods used in our ontology builder, to which we refer as Deep Contextual Word Embedding-Based Semi-Automatic Ontology Builder for Aspect-Based Sentiment Analysis (DCWEB-SOBA). First, we explain which BERT word embeddings we use to detect word semantics in Sect. 4.1. Thereafter, we discuss the methods used in the four stages of our ontology builder. The first stage comprises the construction of the skeletal structure and the initialisation of the ontology, which is described in Sect. 4.2. Then we explain the methods used to select the relevant terms in Sect. 4.3 and to classify sentiment terms in Sect. 4.4. The last stage involves hierarchical clustering of aspect terms to assign them to ontology classes, explained in Sect. 4.5.

4.1 Word Embeddings

The BERT model [5] has been shown to properly model language and handle different tasks. The model can, after pre-training, be post-trained and fine-tuned. In this section, we evaluate to what extent the model has to be post-trained and fine-tuned. We use the dimension reduction method *t-SNE* for a two-dimensional visualisation of the word embeddings for the evaluation. Throughout this research, we run all BERT models on a 2.3 GHz Intel Core i5 CPU with 8 GB RAM in combination with a GPU Tesla V100-SXM2. We use the AdamW optimiser [8] to update the weights during post-training and fine-tuning, as it has been proven to be one of the fastest optimisers in training neural networks [11].

Polysemy-Aware Word Embeddings. The pre-trained BERT model is able to distinguish semantics and polysemous words in our corpus. Figure 1 shows the word embeddings of the word 'turkey' for two different synsets. In half of the sentences 'turkey' is used in the meaning of an animal, denoted by 'Turkey#A', and in the other half of the sentences 'turkey' is used in the meaning of country, denoted by 'Turkey#B'. It can be seen in the plot that the pre-trained model can position these words near similar words. For the comparison, we use word embeddings of 'pizza' and 'Italy'. The pre-trained BERT model can be post-trained to capture the language characteristics in a corpus. However, we find that post-training, using 50.000 reviews from the Yelp dataset, results in a worse separation of the synsets of a word, possibly due to the small size of our domain corpus, therefore, we choose to use the pre-trained word embeddings for term selection and aspect term hierarchical clustering.

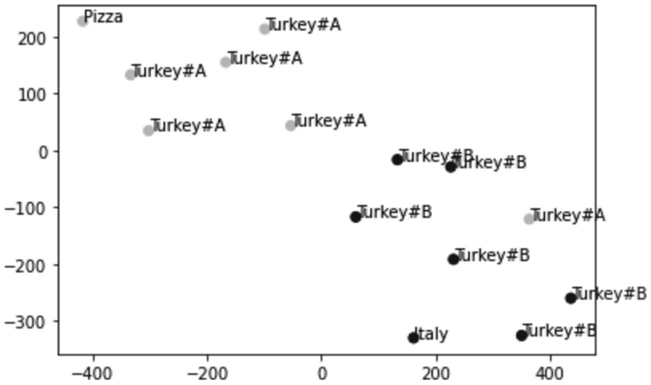

Fig. 1. Pre-trained BERT word embeddings for aspects 'turkey', 'pizza', and 'Italy': polysemous words are well-separated.

Sentiment-Aware Word Embeddings. The pre-trained BERT models position sentiment mention word embeddings based on more characteristics than only their polarity, causing words of a different polarity to be in the same vicinity. This can cause a problem for the sentiment-based part of our method. Figure 2 shows the representation of the pre-trained word embeddings of negative (black) and positive (grey) sentiment words. It can be seen that word embeddings do not separate the sentiment words very well based on polarity.

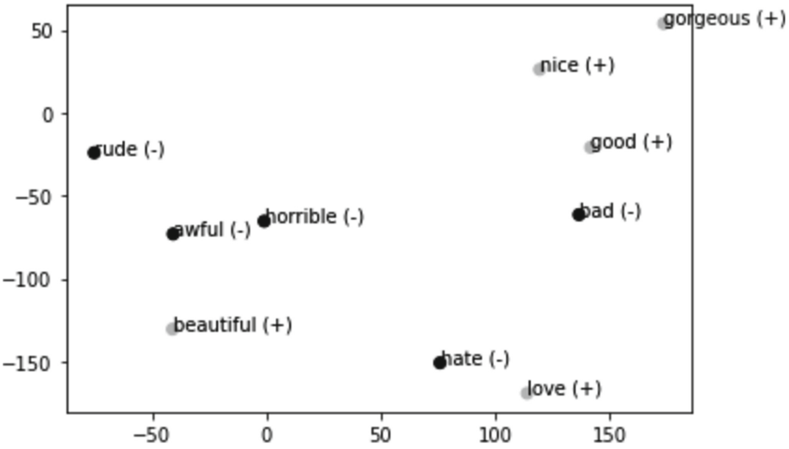

Fig. 2. Pre-trained BERT word embeddings for sentiment words: positive (+) and negative (−) words are not well-separated.

The problem can be solved by fine-tuning the pre-trained BERT model on sentiment classification. The polarity of a sentiment word will determine the position of the word embedding from the created model. This is done on the task of classifying reviews as either positive or negative. Each review in the Yelp dataset is labeled with a star rating between 1 and 5. We treat 4-star and 5-star reviews as positive reviews ($y = 1$) and 1-star and 2-star reviews as negative reviews ($y = 0$). The binary variable y denotes the label for the classification task. During fine-tuning, the model learns the sentiment value of words, and readjusts its word embeddings accordingly. The fine-tuned BERT model is trained on 100.000 reviews from the Yelp dataset.

Figure 3 shows that the fine-tuned model can better separate words on their polarity than the pre-trained model, shown in Fig. 2. Therefore we conclude that these word embeddings are sentiment-aware. In this paper, we refer to fine-tuned word embeddings when sentiment-aware word embeddings are used and refer to pre-trained word embeddings when we mention word embeddings.

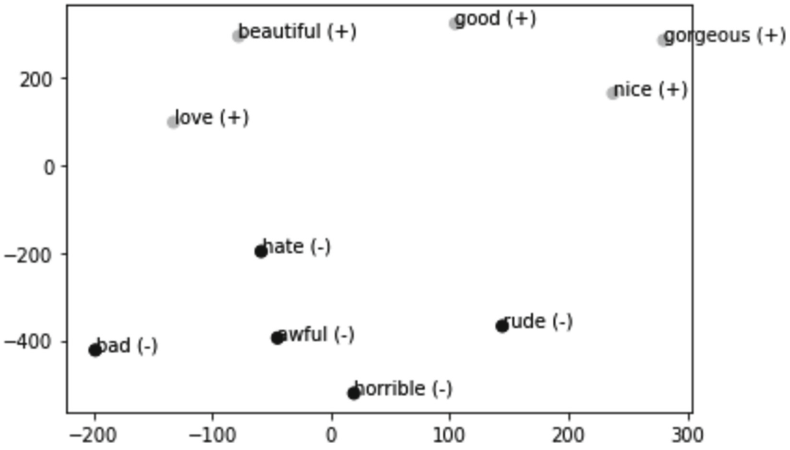

Fig. 3. Fine-tuned BERT word embeddings for sentiment words without post-training: positive (+) and negative (−) words are well-separated.

4.2 Skeletal Ontology Building

The structure of the ontology is based on [16], which contains two main classes, namely `SentimentValue` and `Mention`. The first class contains subclasses denoting sentiment. The second class is divided into subclasses distinguishing the part-of-speech of a term and the aspect. Every relevant term to our domain being a noun, verb, adjective, or adverb gets placed in, respectively, the `Entity`, `Action`, `Property`, or `Modifier` classes. We refer to these subclasses as <`Pos`>.

Each aspect has the format of `CATEGORY#ATTRIBUTE` pair. The `Mention` class has each category, denoted as <`Cat`>, and attribute, denoted as <`Att`>, as subclasses, resulting in the aspect subclasses: `Restaurant`, `Location`, `Food`, `Drinks`, `Prices`, `Experience`, `Service`, `Ambiance`, `Quality`, `Style`, and `Options`. The aspect subclasses are linked through one or multiple aspect properties to corresponding `CATEGORY#ATTRIBUTE` pairs. Table 1 shows all possible pairs.

Table 1. Combinations of categories and attributes in the ontology.

Categories	Attributes				
	General	Prices	Quality	Style&Options	Miscellaneous
Ambience	x				
Drinks		x	x	x	
Experience					x
Food		x	x	x	
Location	x				
Service	x				
Restaurant	x	x		x	

Note, the attribute *Style&Options* is split through lexical properties into two separate aspect subclasses, namely Style and Options. We use these representations instead of Style&Options as there does not exist a word embedding for *Style&Options*, which is required in the next steps of the ontology builder. Furthermore, we have attributes *Miscellaneous* and *General*, as these attributes are too generic, they do not appear as subclasses. Figure 4 presents an overview of the Mention class structure (subclasses have suffix ‘Mention’ removed to avoid cluttering). Each <Cat/Att><Pos>Mention is a subclass of <Cat/Att>Mention. For instance, ServiceEntityMention is a subclass of ServiceMention (<Cat>) and Entity (<Pos>), which in their turn are subclasses of Mention. Furthermore, each <Cat/Att><Pos>Mention has two subclasses <Cat/Att>Positive<Pos> and <Cat/Att>Negative<Pos>. These subclasses have Positive or Negative as their parent class.

The SentimentValue class consists of two subclasses, namely Positive and Negative, which refer to their corresponding sentiment shown in Fig. 5. Neutral sentiments are disregarded in this research. These sentiments are hard to interpret since they carry inherent ambiguity. We consider three types of sentiment words. Type-1 sentiment words (generic sentiment) carry only one polarity irrespective of the context and aspect (e.g., the term ‘bad’ is always negative). Type-1 sentiment words are assigned to the GenericPositive and GenericNegative, which are subclasses of Positive and Negative, respectively, corresponding to their polarity. Furthermore, Type-1 words are also subclasses of their corresponding part-of-speech classes, called GenericPositive<Pos> and GenericNegative<Pos>. Next, polarity consistent words that can not be used for every aspect are Type-2 sentiment words. For example, the word ‘delicious’ is a Type-2 sentiment word only applicable for the aspects ‘food’ and ‘drinks’ and always denotes a positive sentiment. Therefore, ‘delicious’ will appear as lexicalisation in the classes FoodMention and DrinksMention. Last, the polarity of Type-3 sentiment words depends on the aspect and context. For instance, in the context of price the word ‘cheap’ carries positive sentiment, while ‘cheap’ regarding style carries negative sentiment.

User Intervention. We initialise the skeletal structure of the ontology with some Type-1 sentiment words of different part-of-speeches and various sentiment polarities (e.g., ‘good’, ‘bad, ‘poorly’, and ‘hate’). This is necessary because these generic sentiment words are less likely to be extracted in the term selection as these are not specific to a Mention class. For each generic sentiment word, we suggest the 15 most similar words to the user, based on the *cosine similarity* of the word embeddings. These can be accepted or rejected. In preliminary experiments, we find that adding the 15 closest words results in the best trade-off between the quality of the accepted words and the class lexical coverage.

Our word embedding model creates a unique vector for each word instance. For comparisons, however, it is more efficient to limit the comparisons to one representative vector for a synset of a word. Therefore, we average the vectors for each instance of an initialised generic sentiment word. Most of the generic

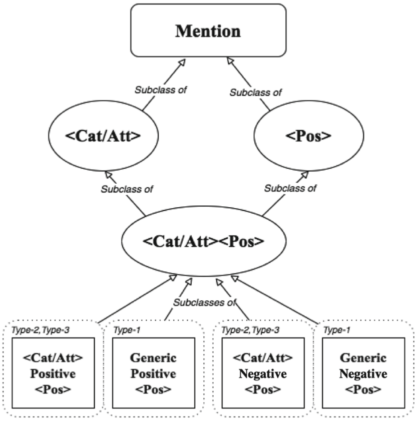

Fig. 4. `Mention` class structure.

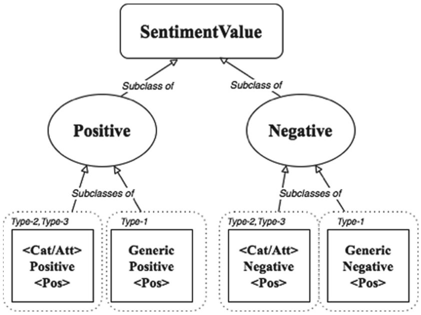

Fig. 5. `SentimentValue` class structure.

sentiment words only have one meaning (i.e., ‘hate’, ‘poorly’, ‘expensive’, ‘excellent’), so for these words we do not lose relevant information.

4.3 Term Selection

The goal of the term selection is to determine the relevant terms for our domain sentiment ontology. The part-of-speech tagger of Stanford NLP Processing Group [18] is used in our research to extract all verbs, nouns, adjectives, and adverbs from the corpus. In contrast to [9], we consider adverbs as well. After tagging, we use a pre-defined stopwords list [3] to filter out all stopwords as those do not contribute to our analysis.

First, we average all word embeddings referring to the same meaning of a word to improve computation time. This is done by averaging the vectors of the lexical representations of each aspect in the `Mention` classes, $\mathcal{A} = \{$`Restaurant`, `Location`, `Food`, `Drinks`, `Price`, `Experience`, `Service`, `Ambiance`, `Quality`, `Style`, `Options`$\}$. In our restaurant domain corpus, these aspects do not have ambiguous semantics, therefore we can average all corresponding vectors without losing semantic information.

Second, we calculate for each instance of a word the *Mention Class Similarity* (MCS) value using the *cosine similarity* as follows:

$$MCS_i = \max_{a \in \mathcal{A}} \left(\frac{v_i \cdot v_a}{\|v_i\| \cdot \|v_a\|} \right), \tag{4.1}$$

where v_i is the vector of word i in the domain-specific word embedding model and v_a is the averaged vector of all lexical representations of the aspect a in the set $\mathcal{A}$. MCS indicates the `Mention` class to which the representation of the word embedding is the closest.

Next, all the word embeddings with an MCS value below a threshold are eliminated. Words that are allocated to the same `Mention` class according to their MCS have their embeddings averaged and named as *word<mention>*, where *word* is the currently considered word and *mention* is an element of $\mathcal{A}$. This averaging procedure is done for MCS values higher than the previous threshold. A good threshold value is pivotal, as otherwise the average vector will not accurately represent a cluster for one synset of a word. We find in preliminary research that a threshold of 0.68 gives a good balance between accuracy and coverage. We define terms as being the refined words after averaging vectors and assigning new word names to them.

After the word embeddings are reduced to relevant terms, we can efficiently determine which terms are suggested to the user. We compute again the MCS value, as this value can differ after averaging. It is used to determine the order in which the terms are suggested. In order to suggest the right amount of words to the user, the MCS values are compared against the threshold value specific to each of the four part-of-speeches. The threshold function is defined as follows:

$$TH^*_{pos} = \max_{TH_{pos}} \left(\frac{accepted}{n+1} \right), \tag{4.2}$$

where TH_{pos} is the threshold score for a <`Pos`> class, n is the number of suggested terms, and *accepted* denotes the number of accepted terms. The value of TH^*_{pos} is increasing in the number of accepted terms. We divide by $n+1$ to avoid the case of a threshold value of 1. This would happen if we divided by n and the first and only suggested term is accepted. The effect of the penalty diminishes for larger n. The threshold will be set to the optimal ratio of accepted terms for a <`Pos`> class, denoted by TH^*_{pos}.

User Intervention. For all accepted terms that are a noun or verb, the user is asked whether the term is a *Sentiment Mention* or *Aspect Mention*. We only consider adjectives and adverbs to indicate a sentiment, as these words can describe the *Aspect Mention*. When a term is a *Sentiment Mention*, the user is asked whether it is a Type-1, 2, or 3 *Sentiment Mention*. For each accepted word, we add all similar words that have a *cosine similarity* larger than a threshold to the ontology as well. Preliminary research results in a threshold of 0.7 which gives a good balance between accuracy and coverage.

4.4 Sentiment Term Clustering

Now that we have determined the relevant *Sentiment Mention* terms, we identify their polarity and class. As the sentiment-aware word embeddings separate the terms well based on polarity, we can calculate the negative and positive scores of our *Sentiment Mention* terms in the following way:

$$PS_i = \max_{p \in \mathcal{P}} \left(\frac{v_i \cdot v_p}{\|v_i\| \cdot \|v_p\|} \right) \qquad NS_i = \max_{n \in \mathcal{N}} \left(\frac{v_i \cdot v_n}{\|v_i\| \cdot \|v_n\|} \right), \tag{4.3}$$

where PS and NS are the positive and negative scores, respectively, for $term_i$. $\mathcal{P}$ and $\mathcal{N}$ are a set of, respectively, positive and negative words of different kinds of sentiment intensity. The set of positive and negative words are, respectively, $\mathcal{P} = \{$'good', 'decent', 'great', 'tasty', 'fantastic', 'solid', 'yummy', 'terrific'$\}$ and $\mathcal{N} = \{$'bad', 'awful', 'horrible', 'terrible', 'poor', 'lousy', 'shitty', 'horrid'$\}$. For each word we use the averaged vectors that are made from the sentiment-aware word embeddings. Finally, v_i denotes the vector of term i, and v_p and v_n are the vectors of a term in the positive and negative set, respectively. The largest score determines the predicted polarity of the word.

User Intervention. For each *Sentiment Mention* term the user decides whether the suggested polarity is correct. Type-2 and Type-3 sentiment words are aspect-specific and can therefore be used in multiple <`Cat/Att`>`Mention` classes. To assign these words to multiple classes, the user is asked to check whether the *Sentiment Mention* term belongs to the <`Cat/Att`>`Mention` class to which the term has the highest cosine similarity. If the user accepts this <`Cat/Att`>`Mention`, the second most similar <`Cat/Att`>`Mention` is suggested. This procedure continues until the user rejects adding a *Sentiment Mention* term to a <`Cat/Att`>`Mention` class.

4.5 Aspect Term Hierarchical Clustering

In this section, we explain the hierarchical clustering procedure for *Aspect Mention* terms in two steps. First, the *Aspect Mention* terms are allocated to their corresponding cluster of the `Mention` classes, $\mathcal{A} = \{$`Restaurant`, `Location`, `Food`, `Drinks`, `Price`, `Experience`, `Service`, `Ambiance`, `Quality`, `Style`, `Options`$\}$. This is done by representative clustering based on the *cosine similarity* between the vector of a term and a vector of the 'base'. The 'base' is made up of the averaged vectors of the lexical representation of the aspects in the `Mention` class, in the same way as previously described in Sect. 4.3.

Next, we create a hierarchy for each cluster in the 'base', using agglomerative hierarchical clustering [1]. Terms start in single clusters and throughout the clustering process they are merged together. In each iteration, clusters with the lowest *Average Linkage Clustering* (*ALC*) value get merged together. *ALC* is determined as follows:

$$ALC(A, B) = \frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} d(a, b), \tag{4.4}$$

where $d(a, b)$ is the Euclidean distance between vectors a and b, and a is in cluster A and b is in cluster B. We find by using the 'elbow' method that the optimal depth in our dendogram to create the preferred hierarchy equals 3 subclasses.

User Intervention. After a term is clustered to a 'base' cluster, the user decides whether the term is correctly placed in the `Mention` class belonging to 'base' cluster by either accepting or rejecting. Whenever a term is rejected, the user has to specify the correct `Mention` class. In this way, all terms start in the right 'base' cluster before performing the hierarchical clustering.

5 Evaluation

In this section, we evaluate our ontology on the SemEval-2016 dataset using the DRANZIERA evaluation protocol "Open" setting [7] against three benchmarks: the manual constructed ontology [16], SOBA [22], and WEB-SOBA [9]. First, in Sect. 5.1 we evaluate our ontology against the benchmarks in terms of descriptive characteristics, time-efficiency, accuracy, and conclusiveness. Thereafter in Sect. 5.2, our ontology is used in a hybrid approach to evaluate the performance for aspect-sentiment classification. The machine used for the evaluation methods is a 2.1 GHz Intel Core i3-10110U CPU with 16 GB RAM.

5.1 Ontology Building Results

We consider an ontology as good if it is accurate and conclusive. A model is conclusive in the case that it is able to make a prediction of the sentiment for an aspect. The higher the coverage of the domain ontology, the more conclusive an ontology is. The accuracy indicates the percentage at which this prediction is correct. Additionally, an ontology should be built time-efficiently.

Table 2 shows that DCWEB-SOBA requires more user time to construct an ontology than WEB-SOBA, as the user has to consider more suggested words due to the addition of adverbs and the inclusion of different synsets of a word. Consequently, more classes and lexicalisations are added to the ontology resulting in higher coverage of the domain. The overall building time (user and computing time) of an ontology with DCWEB-SOBA is shorter since the construction of the fine-tuned BERT model only takes 120 min where the construction of word embeddings in WEB-SOBA takes 300 min. To summarise, DCWEB-SOBA does well at balancing the amount of user time and attained coverage compared to other ontology builders.

Table 2. Comparison statistics for the different ontologies.

	Manual	SOBA	WEB-SOBA	DCWEB-SOBA
Classes	365	470	376	539
Lexicalisations	374	1087	348	485
User time (minutes)*	420	90	40	60
Computing time (minutes)**	0	90	30 (+300)	30 (+120)

*User time is the time spent on the user interventions.

**Computing time is the time required to construct the ontology excluding user time.

Additionally, we evaluate the performance of the ontologies obtained from the ontology builders on the SemEval-2016 test dataset. Table 3 shows that our obtained ontology has a fairly high accuracy for its conclusive cases compared to the other semi-automatically built ontologies, only SOBA performs better. Furthermore, we can see that our ontology is conclusive in 49.69% of the cases which is a significant increase from the 35.39% of WEB-SOBA. This indicates that using deep contextual word embeddings results in a more accurate and more conclusive ontology than an ontology built with non-contextual word embeddings like WEB-SOBA.

Table 3. Comparison statistics for the percentage of conclusive cases.

	Manual	SOBA	WEB-SOBA	DCWEB-SOBA
Conclusive (%)	61.85	**64.62**	35.39	49.69
Accuracy for conclusive cases (%)	**86.82**	85.48	83.04	84.52

5.2 Hybrid Setting Results

Table 4 shows that SOBA performs best on conclusiveness and the accuracy on these predictions, however, [9] shows that SOBA is outperformed by WEB-SOBA in the hybrid setting of HAABSA. Therefore, we aim to show that DCWEB-SOBA can outperform other ontology builders in the state-of-the-art HAABSA++ setting, first proposed in [19]. For LCR-Rot-hop++, the backup model in HAABSA++, the following hyperparameters can be used to reproduce our analysis: the learning rate, the keep probabilities, and the momentum, are set to 0.001, 0.7, and 0.085, respectively. The model is trained on 150 iterations on the train SemEval-2016 dataset.

To analyse the overall performance of DCWEB-SOBA in this setting, we use the weighted average of the accuracy for conclusive and inconclusive cases. We observe in Table 4 that the ontology builders using word embeddings, WEB-SOBA and DCWEB-SOBA, reach a lower accuracy for their ontology but a higher accuracy for its backup model similar to the results in the HAABSA setting in [9].

Furthermore, the DCWEB-SOBA ontology gives the highest accuracy when used in HAABSA++ compared to the other ontologies with an accuracy of 81.85%. This indicates that the use of word embeddings is essential for the HAABSA++ model. DCWEB-SOBA scores better on the accuracies of both the ontology and backup model than WEB-SOBA, this further emphasises the importance of deep contextual word embeddings in sentiment classification.

Table 4. Comparison statistics for each ontology, LCR-Rot-hop++ and the two combined.

	Ontology		LCR-Rot-hop++	Combined
	Conclusiveness	Accuracy	Accuracy	Accuracy
Manual	61.85%	86.82%	72.98%	81.54%
SOBA	64.62%	**85.48**%	73.91%	81.38%
WEB-SOBA	35.39%	83.04%	78.81%	80.31%
DCWEB-SOBA	49.69%	84.52%	**79.20**%	**81.85**%

6 Conclusion

In this research, we propose DCWEB-SOBA to construct a semi-automatically built ontology using deep contextual word embeddings for aspect-based sentiment analysis. We hypothesise that by using deep contextual word embeddings which can deal with semantics and polysemy, we can improve the performance of sentiment classification based on its accuracy and conclusiveness. The proposed methodology makes use of contextual word embeddings at its various steps: skeletal ontology building, term selection, sentiment term clustering, and aspect term hierarchical clustering.

DCWEB-SOBA achieves higher accuracy compared to WEB-SOBA when we measure the sentiment prediction accuracy on conclusive cases. The accuracy increases from 83.04% to 84.52%. Additionally, DCWEB-SOBA is able to predict the sentiment for more aspects as it is conclusive in 49.69% of the cases, compared to 35.39% for WEB-SOBA. When we use the ontology obtained from DCWEB-SOBA in a hybrid approach, HAABSA++, we achieve an accuracy of 81.85% which outperforms the other used ontologies. This shows that using deep contextual word embeddings increases the performance in a hybrid approach for aspect-based sentiment analysis compared to non-contextual word embeddings.

Our research highlights the importance of the usage of deep contextual word embeddings in sentiment classification. For future work, we suggest the usage of Facebook's RoBERTa, introduced in [10], for creating contextual word embeddings. This model has been trained on a dataset more than ten times as large as BERT and is solely trained on the Masked Language Modeling task. In addition, RoBERTa makes use of dynamic masking: training by masking different words for every epoch. These factors make us believe that the usage of RoBERTa in a hybrid approach to aspect-based sentiment analysis is very promising.

Furthermore, the fine-tuned model can be improved by training on separating Type-3 sentiment words. This can be done by training the model on data with aspect ratings instead of review rating. As a consequence, we expect the fine-tuned model to be trained on less noisy data and result in a model which is able to cope better with aspect-based sentiment analysis.

References

1. Behnke, L.: (2012). https://github.com/lbehnke/hierarchical-clustering-java
2. Blaschke, C., Valencia, A.: Automatic ontology construction from the literature. Genome Inform. **13**, 201–213 (2002)
3. Bleier, S.: (2000). https://gist.github.com/sebleier/554280
4. Dera, E., Frasincar, F., Schouten, K., Zhuang, L.: SASOBUS: semi-automatic sentiment domain ontology building using Synsets. In: Harth, A., et al. (eds.) ESWC 2020. LNCS, vol. 12123, pp. 105–120. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_7
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019), pp. 4171–4186. ACL (2019)
6. Dragoni, M., Donadello, I., Cambria, E.: OntoSenticNet 2: Enhancing reasoning within sentiment analysis. IEEE Intell. Syst. **37**(1) (2022)
7. Dragoni, M., Tettamanzi, A., da Costa Pereira, C.: DRANZIERA: an evaluation protocol for multi-domain opinion mining. In: 10th International Conference on Language Resources and Evaluation (LREC 2016), pp. 267–272. ELRA (2016)
8. Gugger, S., Howard, J.: AdamW and super-convergence is now the fastest way to train neural nets. fast.ai (2018)
9. ten Haaf, F., et al.: WEB-SOBA: word embeddings-based semi-automatic ontology building for aspect-based sentiment classification. In: Verborgh, R., et al. (eds.) ESWC 2021. LNCS, vol. 12731, pp. 340–355. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77385-4_20
10. Liu, Y., et al.: RoBERTa: a robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
11. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: 7th International Conference on Learning Representations (ICLR 2019). OpenReview.net (2019)
12. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: 27th Annual Conference on Neural Information Processing Systems (NIPS 2013), pp. 3111–3119. Curran Associates (2013)
13. Peters, M.E., et al.: Deep contextualized word representations. In: 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018), pp. 2227–2237. ACL (2018)
14. Pontiki, M., et al.: SemEval-2016 task 5: aspect-based sentiment analysis. In: 10th International Workshop on Semantic Evaluation (SemEval 2016), pp. 19–30. ACL (2016)
15. Schouten, K., Frasincar, F.: Survey on aspect-level sentiment analysis. IEEE Trans. Knowl. Data Eng. **28**(3), 813–830 (2015)
16. Schouten, K., Frasincar, F.: Ontology-driven sentiment analysis of product and service aspects. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 608–623. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_39
17. Schouten, K., Frasincar, F., de Jong, F.: Ontology-enhanced aspect-based sentiment analysis. In: Cabot, J., De Virgilio, R., Torlone, R. (eds.) ICWE 2017. LNCS, vol. 10360, pp. 302–320. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60131-1_17

18. Toutanova, K., Manning, C.D.: Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP 2010), pp. 63–70. ACL (2000)
19. Truşcă, M.M., Wassenberg, D., Frasincar, F., Dekker, R.: A hybrid approach for aspect-based sentiment analysis using deep contextual word embeddings and hierarchical attention. In: Bielikova, M., Mikkonen, T., Pautasso, C. (eds.) ICWE 2020. LNCS, vol. 12128, pp. 365–380. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50578-3_25
20. Wallaart, O., Frasincar, F.: A hybrid approach for aspect-based sentiment analysis using a lexicalized domain ontology and attentional neural models. In: Hitzler, P., Fernández, M., Janowicz, K., Zaveri, A., Gray, A.J.G., Lopez, V., Haller, A., Hammar, K. (eds.) ESWC 2019. LNCS, vol. 11503, pp. 363–378. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21348-0_24
21. Yelp (2019). https://www.yelp.com/dataset
22. Zhuang, L., Schouten, K., Frasincar, F.: SOBA: semi-automated ontology builder for aspect-based sentiment analysis. J. Web Semant. **60**, 100–544 (2020)