Knowledge Injection from a Domain Sentiment Ontology in an Attention Neural Network for Aspect-Based Sentiment Classification

Charlotte Visser and Flavius Frasincar(✉) [0000−0002−8031−758X]

Erasmus University Rotterdam, Burgemeester Oudlaan 50, 3062 PA Rotterdam, the Netherlands
471231cv@student.eur.nl, frasincar@ese.eur.nl

Abstract. Over the last years, enormous amounts of opinions have become available on the Web, which causes the interest in the task of Aspect-Based Sentiment Classification (ABSC) to rise. Hybrid models for ABSC, which combine a knowledge base with a machine learning algorithm, have gained popularity because of their superior performance. LCR-Rot-hop-ont++ is such a hybrid model, which injects knowledge from a domain sentiment ontology into the well-performing attention neural network LCR-Rot-hop++. In this work, we extend the LCR-Rot-hop-ont++ model in two ways. First, we inject additional knowledge from a domain sentiment ontology into the neural network. Second, we apply a novel weighting mechanism to the injected tokens to control the influence of additional knowledge. Using the SemEval 2015 and SemEval 2016 datasets for evaluation, we find that knowledge injection improves accuracy for datasets with a limited number of observations. Furthermore, we find that the proposed weighting mechanism leads to improved predictive performance of the neural network model.

Keywords: Neural network · Knowledge injection · Aspect-based sentiment classification

1 Introduction

Over the years, the volume of opinionated texts on the Web has been tremendously increasing. This trend causes the interest in the prediction of sentiment from text to rise. Sentiment Analysis (SA) is the process of gathering and analyzing the sentiment and opinions towards entities, such that it can be used by various decision-makers such as businesses and researchers [7]. Especially for large datasets, sentiment analysis is a useful tool as obtaining customers' opinions can become costly and time-consuming. Whereas SA is concerned with determining the sentiment for an entire sentence or a document, the sub-category Aspect-Based Sentiment Analysis (ABSA) has the task of identifying aspects and computing the sentiments towards these [4]. For example, in the sentence "The hotel has a great location, but the rooms are small and outdated", we observe two aspects, location and rooms. Here, the sentiment towards the location

is positive, but the sentiment towards the rooms is negative. ABSA is able to detect the different opinions towards the two aspects, whereas SA is not capable of differentiating between the conflicting sentiments.

Aspect-Based Sentiment Classification (ABSC) methods can be divided in three classes: knowledge-based, machine learning, and hybrid approaches [4]. By combining knowledge-based and machine learning approaches into a hybrid approach, the advantages of both methods can be leveraged. It has been shown that hybrid approaches outperform knowledge-based and machine learning methods [14]. For example, [16] proposes HAABSA++, a two-step hybrid method. First, a domain sentiment ontology is employed to predict sentiment. When the classification is inconclusive, this is followed by a neural network. However, the neural network does not yet leverage the benefits of injecting domain knowledge. It is shown that injecting knowledge increases the performance of language representation models [8]. The question that remains is how exactly knowledge can be injected into a hybrid model. Previous work on the injection of synonyms of words in the test data has been positively evaluated [5]. In this research, we aim to get one step further by investigating the benefits of injecting additional domain knowledge.

For this research, it is assumed that the aspects are predefined and we only consider the task of sentiment classification. In this work, we aim to investigate how to inject domain knowledge from an ontology into a neural network for ABSC. The neural network is the state-of-the-art LCR-Rot-hop++ model proposed by [16]. We follow the approach of [5], where knowledge is injected into the LCR-Rot-hop++ model based on K-BERT [8]. K-BERT uses Knowledge Graphs (KGs) to add branches of additional information about a word to a sentence. Instead of using KGs, [5] uses a domain sentiment ontology for knowledge injection. Particularly, a 0-hop method to navigate in the ontology is used, such that lexical representations of concepts as defined in the ontology are added to the sentences. By injecting knowledge into the LCR-Rot-hop++ model, a new model called LCR-Rot-hop-ont++ is obtained. This method is shown to outperform the LCR-Rot-hop++ model. Furthermore, it outperforms the HAABSA++ model for smaller datasets. Our work extends the model proposed by [5] by following a multi-hop navigation approach in the ontology. That is, we inject lexical representations of subclasses of concepts (1-hop and 2-hop) in addition to word synonyms (0-hop). Furthermore, we propose to apply a novel weighting mechanism to the injected tokens to regulate the influence of the injected knowledge on the model. For the implementation of our approach, we use Python 3.10 as programming language along with PyTorch 2.0.0. The code is made publicly available at `https://github.com/charlottevisser/LCR-Rot-hop-ont-plus-plus`.

The structure of this paper is outlined as follows. In Sect. 2, we provide a review of the relevant literature in the field of ABSC and knowledge injection techniques. Section 3 explains the data used for this work. In Sect. 4, the employed methodology is presented, after which the results of the used methods are evaluated in Sect. 5. Last, Sect. 6 provides our conclusion, discusses the implications of this research, and gives suggestions for further research.

2 Related Work

This section provides an overview of the relevant literature. First, Sect. 2.1 gives a description of hybrid methods for ABSC. Then, Sect. 2.2 discusses knowledge injection as a method to enhance neural networks with domain knowledge.

2.1 Hybrid Models for ABSC

ABSC can be divided into three types of models: knowledge-based, machine learning, and hybrid models [4]. Knowledge-based algorithms make use of a knowledge base to compute sentiment. Knowledge-based approaches have the advantage that the interpretability is relatively straightforward. However, the construction of a knowledge base can be very time-consuming. In contrast to knowledge-based algorithms that require the existence of a knowledge base, machine learning models use a training dataset to learn and classify sentiment, thereby reducing the need for external knowledge. However, these models are more difficult to interpret and require a large labeled dataset for training. Particularly when datasets are small, which is often the case for ABSC, these models might not deliver adequate results. This problem can be solved by combining the two approaches into a hybrid model, such that it takes advantage of the properties of both approaches [4].

There are various approaches for combining a knowledge-driven approach with a machine learning method. One commonly used method involves using a knowledge base that is leveraged by the machine learning algorithm for sentiment prediction [15]. Alternatively, a knowledge-based classifier is used together with a machine learning classifier, which are implemented either sequentially or simultaneously [1]. Among others, [10, 15, 16, 18] have developed hybrid models which outperform knowledge-based or machine learning models.

[18] introduces a two-stage hybrid model, called Hybrid Approach for Aspect-Based Sentiment Analysis (HAABSA). In the first stage, a domain ontology is used for sentiment prediction. If the result is ambiguous, the machine learning method LCR-Rot-hop is used as a backup model. LCR-Rot-hop is an extension of LCR-Rot [20] with multi-hops for attention. [16] further extends HAABSA in two directions and introduce HAABSA++, where the neural network is called LCR-Rot-hop++. The model is enhanced by replacing GloVe word embeddings [11] by BERT word embeddings [6], allowing to capture contextual information of each word based on its surroundings. Furthermore, hierarchical attention is applied by using a high-level attention layer. By doing this, the flexibility of the model is increased. [16] finds that applying attention weighting separately for the context and target vector pairs in each iteration of the rotatory attention mechanism is most beneficial.

[5] also employs the LCR-Rot-hop++ model. However, instead of a two-step approach, a domain sentiment ontology is used simultaneously with the LCR-Rot-hop++ model, by directly injecting knowledge given by synonyms of the current words. During the testing phase, BERT embeddings are replaced by K-BERT embeddings [8], obtaining a new model called LCR-Rot-hop-ont++.

It was found that this model achieves a higher test accuracy than the LCR-Rot-hop++ model. Furthermore, the results of the LCR-Rot-hop-ont++ model show that this approach outperforms HAABSA++ for smaller datasets in terms of test accuracy.

In our work, we adopt a similar approach as [5] by injecting additional knowledge from a domain sentiment ontology into the LCR-Rot-hop++ model during the testing phase, resulting in an extended version of LCR-Rot-hop-ont++. We use HAABSA++ and LCR-Rot-hop++ as benchmark models for performance evaluation.

2.2 Knowledge Injection

Recently, incorporating knowledge into neural networks has gained popularity due to its effectiveness. External knowledge types include among others linguistic, semantic, factual, and domain-specific knowledge. In one solution, external knowledge can be injected during training. However, this requires researchers to train a model by themselves, and this is a computationally expensive and time-consuming process, making it unfeasible for many applications. Furthermore, even if training is feasible, a large domain-specific corpus is needed for this purpose. Alternatively, knowledge can be injected, for example by using a Knowledge Graph (KG), during the testing phase of trained models. This method has the advantage of reduced cost of training, as well as increased interpretability since the KG can be easily interpreted. However, one of the challenges of knowledge injection is Knowledge Noise (KN). Injecting too much knowledge might lead to a deviation from the original meaning of a sentence [8].

[8] proposes Knowledge-enabled BERT (K-BERT), a framework based on the Transformer [17]. K-BERT follows the approach of injecting knowledge obtained from a KG into the BERT model. K-BERT injects knowledge during the testing phase, removing the need for users to conduct their own training. Additionally, K-BERT uses soft-positioning and a visibility matrix to regulate the scope of knowledge, thereby alleviating the issue of KN. This model has shown to outperform BERT on domain-specific tasks.

The LCR-Rot-hop-ont++ model proposed by [5] employs K-BERT for knowledge injection. Rather than a KG, a domain sentiment ontology devised by [15] is used to inject knowledge. The advantage of a domain sentiment ontology is that it captures the concepts and their semantic relations relevant to the domain of interest. This knowledge is often only partially available in the more general KGs. [5] injects lexical representations, which are synonyms of words present in sentences.

For this research, we adopt the approach of [5] as we also use the previously devised domain sentiment ontology for knowledge injection into the LCR-Rot-hop++ model. We extend this approach by following a multi-hop approach for knowledge injection. That is, we also inject lexical representations of concepts related to words by using navigation hops in the ontology into sentences in addition to synonyms.

3 Data

In this research, the datasets SemEval 2015 task 12 [12] and SemEval 2016 task 5 [13] are used for training and performance evaluation of our model. These datasets have been widely used for ABSC tasks, making them a convenient option for comparison between our proposed model and existing models. The datasets comprise restaurant reviews consisting of one or more sentences. Each sentence contains opinions about specific aspects, which are assigned to a category in advance. These aspects are also assigned a polarity, which can be positive, neutral, or negative. Figure 1 presents an example of a review in the SemEval 2015 dataset. In this example, the sentence has multiple targets, which are *menu* and *dishes*.

```
<sentence id="1632445:1">
    <text>The menu is limited but almost all of the dishes are excellent.</text>
    <Opinions>
        <Opinion target="menu" category="FOOD#STYLE_OPTIONS" polarity="negative" from="4" to="8"/>
        <Opinion target="dishes" category="FOOD#QUALITY" polarity="positive" from="42" to="48"/>
    </Opinions>
</sentence>
```

Fig. 1. An example sentence from the SemEval 2015 dataset.

Table 1 shows the frequencies and distribution of the opinions of the SemEval 2015 and SemEval 2016 datasets after preprocessing (opinions with implicit aspects are removed as the employed neural network requires explicit aspects). In terms of the number of reviews, the SemEval 2015 and SemEval 2016 datasets consist of a total of 1279 and 1880 reviews for training, and 597 and 650 reviews for testing, respectively. Consequently, the SemEval 2015 dataset is comparatively smaller for both the training and testing phases. The majority of opinions in both the training data and test data are characterized by positive polarities, followed by negative and neutral polarities.

Table 1. Distribution of opinions of train and test data of the SemEval 2015 and SemEval 2016 datasets.

	Training data						Test Data					
	Negative		Neutral		Positive		Negative		Neutral		Positive	
	Freq.	%	Freq.	%	Freq.	%	Freq.	%	Freq.	%	Freq.	%
SemEval 2015	280	21.9	36	2.8	963	75.3	207	34.7	37	6.2	353	59.1
SemEval 2016	489	26.0	72	3.8	1319	70.2	135	20.8	32	4.9	483	74.3

In our approach, knowledge is injected by using a domain sentiment ontology. The ontology contains concepts, for which it reports their lexical representations

as well as the corresponding subclasses. For our research, we use an ontology with concepts related to restaurants [15], which contains 444 concepts in total, 576 subclass relationships, and 453 lexical representations. Figure 2 shows an example of the concept *Waiter* in the ontology, with its lexical representations and superclasses.

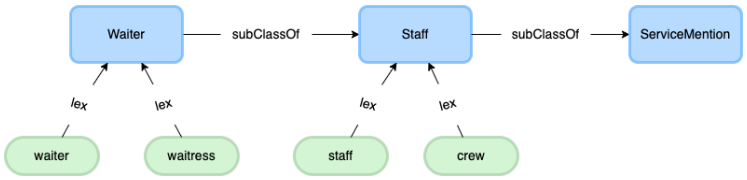

Fig. 2. An example of a concept in the domain sentiment ontology with its lexical representations and superclasses.

Last, we use pre-trained word embeddings for this research. Particularly, we use the uncased BERT [6] base model, trained on BookCorpus and English Wikipedia [19]. It has 12 transformer layers, a hidden size of 768, and 12 self-attention heads.

4 Methodology

This section describes the methodology followed for this work. First, the neural network LCR-Rot-hop++ [16] is introduced in Sect. 4.1. Then, the methods to inject knowledge into the neural network are discussed in Sect. 4.2. By injecting knowledge into LCR-Rot-hop++, we obtain the model called LCR-Rot-hop-ont++. An overview of LCR-Rot-hop-ont++ is given in Fig. 3.

4.1 LCR-Rot-hop++

For each opinion, the sentence S of N words is divided into three parts; the left context $[s_1^l, s_2^l, \ldots, s_L^l]$, the target phrase $[s_1^t, s_2^t, \ldots, s_M^t]$, and the right context $[s_1^r, s_2^r, \ldots, s_R^r]$. L, M, and R represent the length of the three parts, respectively.

Each word is embedded using BERT [6] for the training data, and K-BERT [8] for test data. Three Bi-LSTMs take the embeddings as input, and return hidden states $[h_1^l, h_2^l, \ldots, h_L^l]$ for the left context, $[h_1^r, h_2^r, \ldots, h_R^r]$ for the right context, and $[h_1^t, h_2^t, \ldots, h_M^t]$ for the target phrase, as initial representations.

Next, a rotatory attention mechanism is employed. The rotatory attention mechanism consists of two steps. First, target2context attention generates context representations by using target representations. Second, in context2target attention, left-aware and right-aware target representations are generated by using the left and right contexts, respectively. To obtain a representation of the left and right context, we initially use an average representation of the target $r^{t_p} = \text{pooling}([h_1^t, h_2^t, \ldots, h_M^t])$.

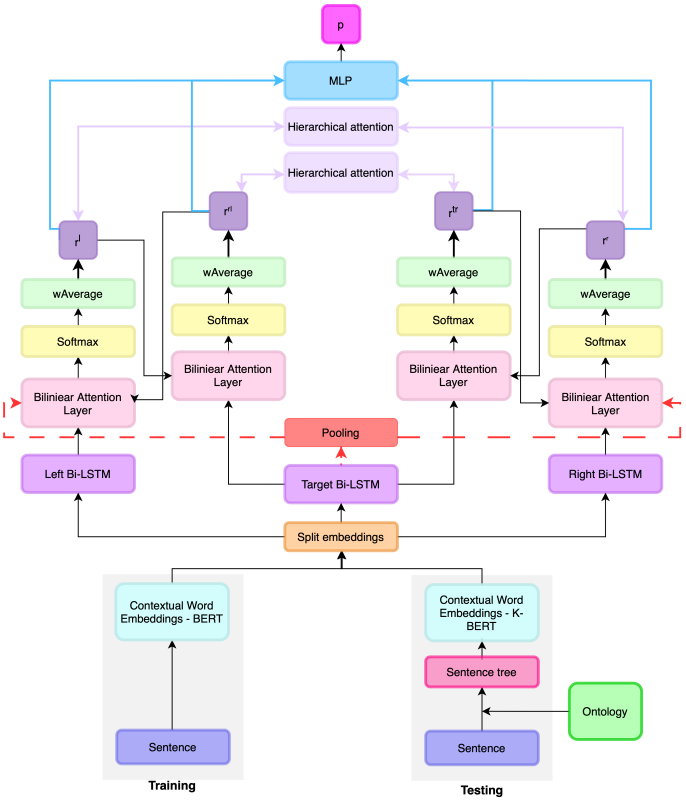

Fig. 3. An overview of LCR-Rot-hop-ont++.

Then, a scoring function f is defined as follows:

$$f(\underset{1\times1}{h_i^l}, \underset{}{r^{t_p}}) = \tanh(\underset{1\times2d}{h_i^{l\prime}} \times \underset{2d\times2d}{W_c^l} \times \underset{2d\times1}{r^{t_p}} + \underset{1\times1}{b_c^l}), \tag{1}$$

where h_i^l is the hidden state of the left context, W_c^l is the weight matrix, r^{t_p} is the average pooling layer of the target phrase, and b_c^l is the bias for the left context.

To obtain normalised attention scores a_i^l, which range between 0 and 1, a softmax function is applied, which takes the score f as input. The normalised attention score is defined as follows:

$$\alpha_i^l = \frac{\exp(f(h_i^l, r^{t_p}))}{\sum_{j=1}^{L} \exp(f(h_j^l, r^{t_p}))}. \tag{2}$$

Last, the left context representation is computed:

$$\underset{2d\times1}{r^l} = \sum_{i=1}^{L} \underset{1\times1}{\alpha_i^l} \times \underset{2d\times1}{h_i^l}. \tag{3}$$

The right context representation can be obtained in a similar manner. After computing the left and right context representations, these are used to obtain the target representation. A similar approach is taken here as in the first step of the rotatory attention mechanism. In this case, the left-aware target representation is computed by using the hidden states weighted by attention scores:

$$\underset{2d\times 1}{r^{t_l}} = \sum_{i=1}^{M} \underset{1\times 1}{\alpha_i^{t_l}} \times \underset{2d\times 1}{h_i^t}. \tag{4}$$

Similarly to the left-aware target representation, the right-aware target representation can be computed.

Hierarchical attention is then employed to overcome the limitation of relying only on local information in the LCR-Rot-hop approach. By incorporating a high-level representation of the input sentence, hierarchical attention enhances the representation of target2context and context2target vectors by incorporating relevance scores computed at the sentence level.

Attention weighting is applied separately on the context and target vector pairs in each iteration. First, we compute the attention score for each vector $v^i \in \{r^l, r^{t_l}, r^{t_r}, r^r\}$:

$$\underset{1\times 1}{f(v^i)} = \tanh(\underset{1\times 2d}{v_i{}'} \times \underset{2d\times 2d}{W_{z\in\{c,t\}}} + \underset{1\times 1}{b_{z\in\{c,t\}}}), \tag{5}$$

where W_z is a weight matrix and b_z is the bias for context vector pair c and target vector pair t. Then, the attention scores are normalized for the context and target vector pairs separately:

$$a^i = \frac{\exp(f(v^i))}{\sum_{j=1}^{2} \exp(f(v^j))}. \tag{6}$$

The scaled vectors are:

$$\underset{2d\times 1}{v_{new}^i} = \underset{1\times 1}{a^i} \times \underset{2d\times 1}{v_{old}^i}. \tag{7}$$

A hop in the rotatory attention mechanism refers to the process of computing the target and context vectors (r^l, r^r, r^{t_l}, and r^{t_r}). During hyperparameter optimization, it was found that repeating this procedure three times is optimal, which is in accordance with [18].

The four vector representations together are used for aspect-level sentiment prediction. The left- and right-context representations are concatenated with the two-side target representations to get the final sentence representation v:

$$\underset{8d\times 1}{v} = [\underset{2d\times 1}{r^l}; \underset{2d\times 1}{r^{t_l}}; \underset{2d\times 1}{r^{t_r}}; \underset{2d\times 1}{r^r}]. \tag{8}$$

Then, v is used as input for a softmax layer to obtain the sentiment probability:

$$\underset{|C|\times 1}{p} = \text{softmax}(\underset{|C|\times 8d}{W_c} \times \underset{8d\times 1}{v} + \underset{|C|\times 1}{b_c}), \tag{9}$$

where p is the conditional probability distribution, C is the number of sentiment categories, W_c is a weight matrix and b_c is the bias for the sentiment classification.

The LCR-Rot-hop-ont++ model is optimized during training by minimizing the cross-entropy loss function with L_2 regularization, given by:

$$\underset{1\times 1}{L} = -\sum_{j} \left(\underset{|C|\times 1}{y_j} \times \log \underset{|C|\times 1}{\hat{p_j}} \right) + \lambda \|\Theta\|^2, \tag{10}$$

where y_j represents the true sentiment, $\hat{p}_j$ is the predicted probability of sentiment classifications, λ is the L_2 regularization term, and Θ is a vector which contains all the parameters of the model. The model parameters are updated by using Stochastic Gradient Descent (SGD) with momentum and a dropout rate.

Prior to model training, the hyperparameters of the model are tuned by using a Tree-based Parzen Estimator (TPE). TPE is a well performing method based on Bayesian optimization [3]. To implement the TPE algorithm, the HyperOpt package [2] is employed. The training dataset is used for hyperparameter tuning, with a division of 80% for model training and 20% for model validation.

4.2 Knowledge Injection

The contextual embeddings are generated for each word by using BERT [6] for the training phase. During the test phase, knowledge is injected by using an approach based on the K-BERT model [8]. To limit the effects of KN, we use soft-positioning and construct a visibility matrix as proposed by [8].

Instead of using a KG, a domain sentiment ontology constructed by [15] is employed to inject knowledge during the testing phase. A k-hop is defined as the process of traversing k classes from the current concept in the ontology to obtain lexical representations. k ranges from 0 to $N-1$, where N represents the total number of classes. Specifically, a 0-hop indicates only adding synonyms, since the ontology does not have to be traversed. A sentence tree is constructed by inserting the tokens from the ontology as child descendant nodes of the words in the original sentence. Synonyms are inserted by adding soft branches to the sentence tree, while the other k-hops are inserted by adding hard branches. An example of a sentence tree is shown in Fig. 4.

Because of the presence of branches after the injection of the k-hops, the sentence tree cannot be used as input for BERT's embedding layer. The sentence tree is rearranged by inserting the tokens located in the branches subsequent to their originating (word) nodes. By doing this, the sentence loses its original structure. This issue can be solved by using soft-positioning. All injected tokens get the same soft-position as the original word in the sentence. However, this might lead the sentence to lose its original meaning. To solve this issue, we construct the visibility matrix M, such that the injected tokens have no influence on the hidden state values of words that are in different branches. The visibility

matrix M is defined as:

$$M_{ij} = \begin{cases} 0 & \text{if } w_i \oplus w_j \\ -\infty, & \text{if } w_i \otimes w_j, \end{cases} \tag{11}$$

where $\oplus$ denotes that w_i and w_j belong to the same branch, and $\otimes$ denotes that they are not in the same hard branch. i and j represent the hard-position indexes of the words. Words within the same hard branch are visible to each other, whereas words belonging to different hard branches are not. Synonyms from soft branches are able to only view the words that they originate from. Figure 4 shows an example of a sentence tree with the corresponding visibility matrix. Words from the corresponding hard-position indices that are visible to each other are shown in red (dark grey in black and white printing), and words that are not visible to each other are shown in light grey. All words in the original sentence form a branch and can view each other, while injected words can view only words on the same branch.

The visibility matrix M is used to mask self-attention scores. Mask-self attention is defined as follows:

$$Q^{i+1}, K^{i+1}, V^{i+1} = h^i W_q, h^i W_k, h^i W_v, \tag{12}$$

$$S^{i+1} = softmax(\frac{Q^{i+1} {K^{i+1}}^\top + M}{\sqrt{d_k}}), \tag{13}$$

$$h^{i+1} = S^{i+1} V^{i+1}, \tag{14}$$

where W_q, W_k, and W_v are model parameters, h^i is the hidden state of the i-th mask-self attention blocks, and d_k is the scaling factor given by the dimensionality of the current head (k).

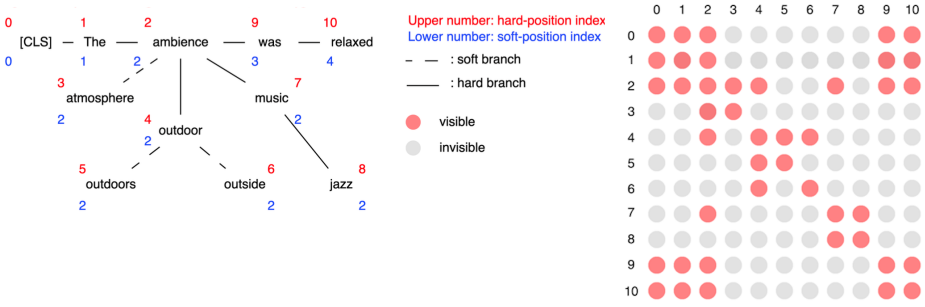

Fig. 4. An example of a sentence tree and the corresponding visibility matrix.

Hop Weight. On the one hand, increasing the number of hops in the domain sentiment ontology may cause the original meaning of the sentence to change.

This is due to the fact that injecting subclasses introduces concepts with increasingly distinct meanings as the number of hops increases. On the other hand, alternative lexical representations of present concepts might be better able to capture the sentiment because people sometimes use a variety of words, even when they all convey the same underlying meaning. Therefore, lexical representations such as synonyms can be more important than the original words due to their presence in the training data. For these reasons, we apply a hop weighting mechanism to the embedding of the injected tokens from the ontology to control the influence of the injected knowledge. The hop weight is inverse proportional to the number of hops in the ontology, such that tokens from a hop further in the ontology obtain a lower weight. The hop weight mechanism is defined as:

$$hw_i = \begin{cases} 1 & \text{if } t_i \text{ in original sentence} \\ e^{-(n_{hops}+\gamma)} & \text{otherwise,} \end{cases} \tag{15}$$

where n_{hops} is the number of hops in the ontology to reach token t_i, and γ is a hyperparameter.

5 Results

In this section, we present the results obtained from the LCR-Rot-hop-ont++ model. In Sect. 5.1, we compare the accuracy of our model with two benchmark models, namely LCR-Rot-hop++ and HAABSA++. In Sect. 5.2, the impact of soft-positioning, the visibility matrix, and hop weighting is investigated.

5.1 Knowledge Injection

In Table 2, the accuracy of the LCR-Rot-hop-ont++ model with 0, 1, and 2 hops injected from the domain sentiment ontology is presented for the SemEval 2015 and SemEval 2016 datasets, along with the accuracy of the benchmark models HAABSA++ and LCR-Rot-hop++. We give the results of the HAABSA++ model as reported by [18], since we do not replicate this method. First, we note that higher test accuracies are obtained for all models for the SemEval 2016 dataset in comparison to the SemEval 2015 dataset. This is probably due to the fact that for SemEval 2016 dataset we have more data for training. The results for the SemEval 2015 dataset show that the LCR-Rot-hop-ont++ model outperforms the LCR-Rot-hop++ model for all hop levels. Specifically, the LCR-Rot-hop-ont++ model achieves a test accuracy of 80.9%, which is the highest among all hop levels. However, the HAABSA++ model still outperforms the best LCR-Rot-hop-ont++ model by 0.8 percentage points. For the SemEval 2016 dataset, the LCR-Rot-hop++ model achieves the highest accuracy of 87.5%, outperforming both LCR-Rot-hop-ont++ and HAABSA++. Thus, the methods which use an approach containing an ontology do not improve performance for the SemEval 2016 dataset. Furthermore, HAABSA++ outperforms the LCR-Rot-hop-ont++ models for 0, 1, and 2 hops.

Using the Wilcoxon signed rank test on 100 bootstraps on the test data, we have shown that LCR-Rot-hop-ont++ (no matter the number of hops) is statistically significant better than LCR-Rot-hop++ on SemEval 2015 and LCR-Rot-hop++ is statistically significant better than LCR-Rot-hop-ont++ (no matter the number of hops) on SemEval 2016 with p-values smaller than 0.001.

We conclude that leveraging a domain sentiment ontology both in a hybrid approach and directly injecting in a neural network increases accuracy for smaller datasets. However, a hybrid approach is preferred over knowledge injection. Furthermore, an ontology does not boost performance for larger datasets, as these allow for a better trained neural model.

Table 2. Accuracy of HAABSA++, LCR-Rot-hop and LCR-Rot-hop++ for the SemEval 2015 and SemEval 2016 datasets.

Model	SemEval 2015	SemEval 2016
HAABSA++	**81.7**%	87.0%
LCR-Rot-hop++	80.4%	**87.5**%
LCR-Rot-hop-ont++ (0-hops)	80.6%	86.8%
LCR-Rot-hop-ont++ (1-hop)	80.9%	86.8%
LCR-Rot-hop-ont++ (2-hops)	80.7%	86.9%

5.2 Ablation Experiment

The results of the ablation experiment conducted on the LCR-Rot-hop-ont++ model for the SemEval 2015 and SemEval 2016 datasets are reported in Table 3. Different components of the model, being soft-positioning, the visibility matrix, and the hop weight mechanism, are removed to investigate their impact on the performance of the model.

Considering the SemEval 2015 dataset, the test accuracy is lower for 0, 1, and 2 hops in the ontology without soft-positioning, visibility matrix, and hop weight mechanism. Furthermore, considering the SemEval 2016 dataset, removing only soft-positioning leads to an improved accuracy for all hop levels. For 0-hops, 1-hop, and 2-hops the accuracy improves with 0.7, 0.3, and 0.3, respectively. However, removing the visibility matrix results in a decrease in accuracy for all hops. Last, removing the hop weight mechanism decreases the accuracy for 0, 1, and 2 hops. This suggests that the visibility matrix and the hop weight mechanism are of greater importance for the performance of LCR-Rot-hop-ont++ than soft-positioning.

The previous result is partly in accordance with the work of [8], where the authors argue that the visibility matrix and soft-positioning are both of importance to limit KN. On the one hand, both works show that that the visibility matrix is of importance since removing it decreases performance for all datasets. On the other hand, [8] shows that excluding soft-positioning also worsens performance of all datasets, while we only find that this holds for the SemEval 2015

dataset. Interestingly, in our case, removing soft-positioning for LCR-Rot-hop-ont++ obtains better results for the SemEval 2016 dataset than HAABSA++. Based on our results, the use of soft-positioning degredates performance on larger datasets. This could be due to the fact that the heuristic used for soft-positioning introduces too much noise in a model that is already of a good quality due to the use of a large dataset for training.

Table 3. Model accuracy results for ablation experiment on SemEval 2015 and SemEval 2016 datasets.

	Model	SemEval 2015	SemEval 2016
	HAABSA++	81.7%	87.0%
	LCR-Rot-hop++	80.4%	87.5%
0-hops	LCR-Rot-hop-ont++	**80.6**%	86.8%
	LCR-Rot-hop-ont++ (w/o SP)	80.1%	87.5%
	LCR-Rot-hop-ont++ (w/o VM)	80.4%	86.2%
	LCR-Rot-hop-ont++ (w/o hop weight)	80.2%	86.6%
	LCR-Rot-hop-ont++ (w/o VM, SP, hop weight)	80.2%	**87.7**%
1-hop	LCR-Rot-hop-ont++	**80.9**%	86.8%
	LCR-Rot-hop-ont++ (w/o SP)	79.6%	**87.1**%
	LCR-Rot-hop-ont++ (w/o VM)	79.2%	84.0%
	LCR-Rot-hop-ont++ (w/o hop weight)	80.6%	86.3%
	LCR-Rot-hop-ont++ (w/o VM, SP, hop weight)	78.6%	86.5%
2-hops	LCR-Rot-hop-ont++	**80.7**%	86.9%
	LCR-Rot-hop-ont++ (w/o SP)	79.2%	**87.2**%
	LCR-Rot-hop-ont++ (w/o VM)	79.1%	83.7%
	LCR-Rot-hop-ont++ (w/o hop weight)	80.4%	86.2%
	LCR-Rot-hop-ont++ (w/o VM, SP, hop weight)	78.7%	86.0%

6 Conclusion

This research follows the model of [5] called LCR-Rot-hop-ont++. This model injects knowledge from a domain sentiment ontology into the neural network LCR-Rot-hop++. This model uses BERT [6] during the training phase. Knowledge is injected during the testing phase in a comparable manner to K-BERT [8]. However, instead of using a KG, [5] injects lexical representations of present concepts using a domain sentiment ontology. Following the methodology of K-BERT, LCR-Rot-hop-ont++ employs a visibility matrix and soft-positioning in order to deal with KN. We extend this framework by injecting lexical representations of selected subconcepts in addition to the lexical representations of the original concepts. Furthermore, we propose a hop weight mechanism, which assigns a weight to the embedding of injected tokens, to control for the additional injected knowledge. For training and evaluation of the model, we use the SemEval 2015 and SemEval 2016 datasets. We compare our proposed model with the benchmark models LCR-Rot-hop++ and HAABSA++.

The results show that the LCR-Rot-hop-ont++ model outperforms LCR-Rot-hop++ for the SemEval 2015 dataset when traversing 0, 1, and 2 subclasses from the current concept. Particularly, the highest accuracy of 80.9% is achieved by performing 1 hop. Furthermore, HAABSA++ outperforms LCR-Rot-hop++ in this case. However, LCR-Rot-hop++ achieves a higher accuracy than LCR-Rot-hop-ont++ and HAABSA++ for the SemEval 2016 dataset. Since SemEval 2015 is a smaller dataset than SemEval 2016, this shows that knowledge injection is more helpful for datasets with limited observations. Last, LCR-Rot-hop-ont++ is outperformed by HAABSA++ for both datasets. This implies that a two-step approach outperforms knowledge injection during the testing phase.

We find that including the visibility matrix improves performance for all hops in the LCR-Rot-hop-ont++ model for both datasets. Furthermore, soft-positioning is shown to be useful for the SemEval 2015 dataset. However, the LCR-Rot-hop-ont++ model performs better than the HAABSA++ model when soft-positioning is excluded for the SemEval 2016 dataset. Last, results show that the hop weight mechanism enhances the performance of the LCR-Rot-hop-ont++ model.

For future research, an opportunity for the improvement of our proposed model is to replace BERT word embeddings by RoBERTa word embeddings [9], as RoBERTa has shown to improve performance in comparison to BERT. Furthermore, our model currently only injects knowledge in the testing phase. The effects of injecting knowledge during the training phase can also be investigated, since this approach might enhance the model's ability to learn from external information. Last, a suggestion for further research is to control the amount of knowledge injected to limit KN. In our work, we inject all lexical representations of concepts for the current hop in the domain sentiment ontology. However, by limiting the injected knowledge to a percentage of what is available in the ontology, the risk of injecting excessive or irrelevant knowledge might be reduced.

References

1. van Bekkum, M., de Boer, M., van Harmelen, F., Meyer-Vitali, A., ten Teije, A.: Modular design patterns for hybrid learning and reasoning systems. Applied Intelligence **51**(9), 6528–6546 (2021)
2. Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., Cox, D.D.: Hyperopt: a python library for model selection and hyperparameter optimization. Computational Science & Discovery **8**(1), 014008 (2015)
3. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: 24th International Conference on Neural Information Processing Systems (NIPS 2011), pp. 2546–2554. Curran Associates (2011)
4. Brauwers, G., Frasincar, F.: A survey on aspect-based sentiment classification. ACM Computing Surveys **55**(4), 1–37 (2023)
5. Dekker, R., Gielisse, D., Jaggan, C., Meijers, S., Frasincar, F.: Knowledge graph injection for aspect-based sentiment classification. In: 34th International Conference on Database and Expert Systems (DEXA 2023). LNCS, vol. 14147, pp. 173–187. Springer (2023)

6. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: 17th Conference of the Northern American Chapter of the Association for Computational Linguistics (NAACL-HLT 2019). pp. 4171–4186. ACL (2019)
7. Liu, B.: Sentiment analysis: Mining opinions, sentiments, and emotions. Cambridge University Press, 2nd edn. (2020)
8. Liu, W., Zhou, P., Zhao, Z., Wang, Z., Ju, Q., Deng, H., Wang, P.: K-BERT: Enabling language representation with knowledge graph. In: 34th AAAI Conference on Artificial Intelligence (AAAI 2020). vol. 34, pp. 2901–2908. AAAI (2020)
9. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
10. Meškelė, D., Frasincar, F.: ALDONA: A hybrid solution for sentence-level aspect-based sentiment analysis using a lexicalised domain ontology and a neural attention model. In: 34th ACM Symposium on Applied Computing (SAC 2019). pp. 2489–2496. ACM (2019)
11. Pennington, J., Socher, R., Manning, C.D.: GloVe: Global vectors for word representation. In: 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014). pp. 1532–1543. ACL (2014)
12. Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., Androutsopoulos, I.: SemEval-2015 task 12: Aspect based sentiment analysis. In: 9th International Workshop on Semantic Evaluation (SemEval 2015). pp. 486–495. ACL (2015)
13. Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., Androutsopoulos, I.: SemEval-2016 task 5: Aspect based sentiment analysis. In: 10th International Workshop on Semantic Evaluation (SemEval 2016). pp. 19–30. ACL (2016)
14. Schouten, K., Frasincar, F.: Survey on aspect-level sentiment analysis. IEEE Transactions on Knowledge and Data Engineering **28**(3), 813–830 (2015)
15. Schouten, K., Frasincar, F., de Jong, F.: Ontology-enhanced aspect-based sentiment analysis. In: 17th International Conference on Web Engineering, (ICWE 2017). LNCS, vol. 10360, pp. 302–320. Springer (2017)
16. 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: 20th International Conference on Web Engineering (ICWE 2020). LNCS, vol. 12128, pp. 365–380. Springer (2020)
17. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: 31st International Conference on Neural Information Processing Systems (NIPS 2017). pp. 5998–6008. Curran Associates (2017)
18. Wallaart, O., Frasincar, F.: A hybrid approach for aspect-based sentiment analysis using a lexicalized domain ontology and attentional neural models. In: 16th Extended Semantic Web Conference (ESWC 2019). LNCS, vol. 11503, pp. 363–378. Springer (2019)
19. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP 2020). pp. 38–45. ACL (2020)
20. Zheng, S., Xia, R.: Left-center-right separated neural network for aspect-based sentiment analysis with rotatory attention (2018), *arXiv preprint arXiv:1802.00892*