Knowledge Injection for Aspect-Based Sentiment Classification

Romany Dekker, Danae Gielisse, Chaya Jaggan, Sander Meijers, and Flavius Frasincar[0000−0002−8031−758X] (✉)

Erasmus University Rotterdam, Burgemeester Oudlaan 50, 3062 PA Rotterdam, the Netherlands
529424rd@student.eur.nl, 539908dg@student.eur.nl, 538604aj@student.eur.nl, 530192sm@student.eur.nl, frasincar@ese.eur.nl

Abstract. Since the increase of Web reviews of products and services, Aspect-Based Sentiment Classification (ABSC) has become more important to determine the sentiment of online opinions. Useful information extracted from these reviews can then be used by companies themselves, but can also be applicable by consumers. In the recent literature on ABSC, hybrid methods, which combine knowledge-based and machine learning approaches, are becoming more popular as well. However, in this work, instead of following a two-step procedure, we attempt to improve the model accuracy by proposing to directly inject the information from a domain ontology in a state-of-the-art neural network model, more precisely LCR-Rot-hop++. Furthermore, by using soft-positioning and visible matrices we aim to prevent that the injected knowledge hinders the semantics of the original sentences. To evaluate the accuracy of our model, LCR-Rot-hop-ont++, we use the standard SemEval 2015 and SemEval 2016 datasets for ABSC. We conclude that knowledge injection in the neural network is effective for sentiment classification, especially if the amount of labeled data is limited.

Keywords: LCR-Rot-hop-ont++ · ABSC · Knowledge Injection

1 Introduction

In an era where life seems to be moving more and more towards the online dimension, information has as well moved to the online domain. Taking form in Web reviews of products and services, enormous amounts of useful information are up for grabs. When the size of this information gets enormously large, the sheer size of data requires automatic approaches in order to retrieve people's opinion from their reviews. One proposal for an automatic approach for gathering these opinions is called sentiment analysis [6], which determines the sentiment expressed in a text.

A discipline within sentiment analysis is Aspect-Based Sentiment Analysis (ABSA) [15]. Here, the overall sentiment of an entity is determined by identifying the sentiment of the aspects of that entity. ABSA usually consists of two steps:

finding the aspects present in the sentences and determining the sentiments associated to these aspects. The sentiments are often defined as positive, neutral, or negative. For example, in the sentence “I was very disappointed with this restaurant.” the aspect is “restaurant” and the sentiment is classified as negative. In this research, we assume the aspect to be given and therefore focus only on the sentiment classification part of the ABSA, also known as Aspect-Based Sentiment Classification (ABSC) [3]. To efficiently and accurately predict the sentiment of the given aspects, knowledge-based methods and machine learning approaches are generally used. Because a combination of both techniques outperforms the models that only use one technique [16], a hybrid approach is considered [2].

One hybrid approach is HAABSA++ [17] which increased the accuracy of sentiment classification w.r.t. its predecessor the Hybrid Approach for Aspect-Based Sentiment Analysis (HAABSA) [19] by introducing two extensions. First, in HAABSA++ the non-contextual GloVe [11] word embeddings are replaced by deep contextual word embeddings like BERT [4]. Furthermore, a hierarchical attention is implemented ensuring that the high-level input sentence representations are tuned to each other. HAABSA++ is a two-step model which first uses a domain ontology to estimate the sentiment. When this turns out to be inconclusive, a back-up algorithm is used taking form as the state-of-the-art neural network LCR-Rot-hop++ which has a rotatory attention mechanism based on LCR-Rot [23] and a hopping (repetition) mechanism defined by its successor LCR-Rot-hop [19].

Different than in [17] where the ontology is used before the back-up neural network, in this work we plan to inject the knowledge of the domain ontology in the neural network itself. We call the new model LCR-Rot-hop-ont++, referring to the ontology now being incorporated in the neural network LCR-Rot-hop++. We know that such an approach has been used for contextual word embeddings, i.e., K-BERT [8], for language representation, but very little is known about applying this method to the neural models (in our case LCR-Rot-hop++) in the context of ABSC. We approach our knowledge injection inspired by K-BERT, where the target text is first augmented with domain knowledge before being passed on to the neural network, providing for a richer context. Furthermore, our research contributes to the previous work in the following way. We make use of a domain ontology instead of a knowledge graph to inject knowledge. Furthermore, we follow a multi-hop approach where different concepts, which are semantically close to the original words, from the domain ontology are injected into the sentences. The Python source code of our model has been made publicly available at `https://github.com/DanaeGielisse/LCR-Rot-hop-ont-plus-plus`.

The rest of the paper is organized as follows. In Sect. 2 relevant related works regarding sentiment analysis and knowledge injection are discussed. Next, Sect. 3 presents the used data for experiments and Sect. 4 provides a detailed description of the proposed method. Furthermore, in Sect. 5 the results are presented and discussed. Last, Sect. 6 gives the conclusions of our research and provides suggestions for future work.

2 Related Work

In this section we discuss the research that is relevant to our work. First, Sect. 2.1 presents hybrid methods for ABSC. Second, in Sect. 2.2 the literature regarding knowledge injection in neural models is outlined.

2.1 Hybrid Models for ABSC

Hybrid models [2] combine the symbolic with sub-symbolic approaches in classifying sentiment. The former makes use of ontologies and lexicons to determine if words are positive, neutral, or negative, and the latter includes machine learning techniques that perform sentiment classification [5]. Several researches concluded that the use of hybrid models for ABSC leads to an increase in the performance of sentiment prediction [9], [10], [16], [19].

ALDONA [9] and its extension ALDONAr [10] are among those researches which use a hybrid approach of ABSC on a sentence-level. Both models determine the polarity of the aspect by using a lexicalized domain ontology and the statistical relations captured by a regularized neural attention model. ALDONAr improves the results of ALDONA by using BERT word embeddings among other things. Furthermore, these models also outperform state-of-the-art models such as CABASC [7].

[17] introduced the HAABSA++ model which is a two-step hybrid model for ABSC. In the first step, a domain sentiment ontology is employed in order to predict the target polarities. If this ontology leads to inconclusive results (e.g., due to conflicting sentiments), the neural network LCR-Rot-hop++ is used. LCR-Rot-hop++ extends the neural network LCR-Rot-hop of HAABSA [19] in two ways. First, deep contextual word embeddings are taken into account, namely ELMo [12] and BERT [4] (better results are obtained with BERT), in order to better deal with word semantics in text. Second, hierarchical attention is used by adding an extra attention layer to the HAABSA high-level representations. Therefore, the method is more flexible to model the input data. It was proved that the fourth method, out of three other methods, gives the highest accuracy, where for each iteration the rotatory attention weighting was applied separately on the intermediate context and target vectors pairs.

In this work, we directly include the information from the domain ontology in the neural network, which results in the model LCR-Rot-hop-ont++. Hence, the ontology is now part of the neural network and thus our model does not follow a two-step approach. More precisely, we account for our whole domain knowledge by replacing the BERT word embeddings with the word embeddings from K-BERT [8]. We also devise a novel multi-hop approach for embedding domain knowledge using K-BERT into the neural network. Last, we compare the performance of our model with HAABSA++ and LCR-Rot-hop++.

2.2 Knowledge Injection in Neural Networks

One of the first works to inject knowledge in a deep neural network is the knowledge-enhanced language representation model K-BERT [8]. This is an extension of the BERT model which is a language representation model that takes contextual information into account [4]. Both the architectures of BERT and K-BERT are based on the Transformer [18]. However, K-BERT outperforms BERT on domain-specific tasks by injecting knowledge obtained from a Knowledge Graph (KG), which could be described as a graph structure modeling a domain, into sentences. Although, this has to be done with care as too much knowledge could be injected which could change the meaning of the original input sentence. This undesired information is called Knowledge Noise (KN). K-BERT could be made more robust to KN in two ways. First, soft-positioning retains the sequential structure of the original sentence. Second, visible matrices control the visible area of each word in the sentence by blocking the words from the injected knowledge except from the corresponding subject.

[22] proposed to use a knowledge-enhanced BERT model for ABSC. The authors utilize a Sentiment Knowledge Graph (SKG) as an external source to increase the accuracy of sentiment detection by injecting domain knowledge into BERT. Like our research, [22] employs the BERT component and injects the information from the SKG into the input sentences and then uses these knowledge enhanced embeddings for ABSC. However, in [22] the knowledge is injected in the same way as in the K-BERT model, namely, in the form of candidate triples which are pairs of concepts and their associated relation. Differently, in our approach we inject the lexical representations obtained from a domain ontology. After comparing the model BERT+SKG to another external knowledge-enhanced model and several self-attention based models, the authors concluded that the BERT+SKG is effective for the ABSC task.

[21] also makes use of a KG to enhance the language representation model. The authors use the model SAKG-BERT, which utilizes the information from Sentiment Analysis Knowledge Graph (SAKG) and the language model BERT. Contrary to our research, the authors focus on the extraction of knowledge triples from the SAKG. Then, the triples are injected as domain knowledge into the input sentences. Also different to our research, the authors have constructed their own KG that is related to online reviews. It was found that the use of a KG shows promising results for sentiment analysis tasks and sentence completion.

In our research, we use an approach based on K-BERT to inject knowledge into a neural network for ABSC instead of a general language model. We, however, inject the information from a domain ontology in the form of lexical representations (words) associated to ontology concepts. Furthermore, compared to previous work, we follow a multi-hop approach for knowledge injection. We only opt to perform knowledge injection at test time. This is because employing such an approach at training time might result in semantic loss, i.e., the model not capturing the meaning conveyed in the original sentences.

3 Data

To evaluate the performance of our model, we have used the SemEval 2015 Task 12 [14] and SemEval 2016 Task 5 [13] datasets. These datasets are often used in research relating to ABSC which makes them convenient for comparing the performance of our model to that of other models. The data contains customers reviews about restaurants which can consist of multiple sentences. Furthermore, the sentences express one or more opinions about the provided aspects. Each aspect is assigned to a predefined category that is labeled as positive, neutral, or negative. In Fig. 1 an example of a customer review from the SemEval 2016 dataset is presented. It is shown that a sentence can have multiple targets, here “food”, “drinks”, and “atmosphere”. Furthermore, the polarities and categories for each target are listed. For training we have used 1278 and 1880 reviews and

```xml
<sentence id="1609375:1">
    <text>We love the food, drinks, and atmosphere!</text>
    <Opinions>
        <Opinion from="12" to="16" polarity="positive"
            category="FOOD#QUALITY" target="food"/>
        <Opinion from="18" to="24" polarity="positive"
            category="DRINKS#QUALITY" target="drinks"/>
        <Opinion from="30" to="40" polarity="positive"
            category="AMBIENCE#GENERAL" target="atmosphere"/>
    </Opinions>
</sentence>
```

Fig. 1: Example of a sentence from the SemEval 2016 dataset.

for testing 597 and 650 reviews of the SemEval 2015 and SemEval 2016 dataset, respectively. Table 1 shows the frequencies of the positive, neutral, and negative targets of the SemEval 2015 Task 12 and SemEval 2016 Task 5 datasets. For both datasets it holds that the targets are valued the most as positive, followed by negative, and neutral.

Table 1: Polarity frequencies of the training and test data of the SemEval 2015 and SemEval 2016 datasets for ABSC.

	Training Data			Test Data		
	Positive	Neutral	Negative	Positive	Neutral	Negative
SemEval 2015	75.3%	2.8%	21.9%	59.1%	6.1%	34.8%
SemEval 2016	70.2%	3.8%	26.0%	74.3%	4.9%	20.8%

To inject knowledge into our input sentences we have used the information obtained from a restaurant domain ontology [16]. An ontology describes a set of concepts and the relations between these concepts. This ontology is divided in multiple classes and subclasses regarding the sentiments and targets. The subclasses of the targets are, for example, service, restaurant, and ambience. The sentiment class only consists of the subclasses positive, neutral, and negative. We have especially focused on the synonyms or different spellings of a certain concept (called concept lexical representations) that the ontology provides. For instance, for the concept "Outside" the ontology gives the lexical representations "outdoor", "outside", and "outdoors". However, related concepts could also be found in the ontology, such as "chef" being a direct subclass of "staff".

Furthermore, in our research we make use of the pre-trained BERT word embeddings [4]. In particular, we use the uncased BERT base model which has 12 transformer layers, 768 as hidden size, and 12 self-attention heads and was pre-trained on BookCorpus and English Wikipedia.

4 Methodology

In this section we explain our model. As an extension of the LCR-Rot-hop++ model we propose to use the information from a domain ontology to enrich the word embeddings. We call this model LCR-Rot-hop-ont++. First, in Sect. 4.1 we describe how the LCR-Rot-hop++ model works. Furthermore, in Sect. 4.2 we explain how we inject the knowledge from the domain ontology in the test sentences using K-BERT. The methodology was implemented in Python (version 3.7) with the HuggingFace's Transformer [20] and Google's TensorFlow [1] packages.

4.1 LCR-Rot-hop++

LCR-Rot-hop++ builds further on the model proposed in [17] which uses contextual word embeddings as input for three bidirectional Long Short-Term Memory (Bi-LSTM) models. Bi-LSTMs are a particular Recurrent Neural Network (RNN) that are able to incorporate long-term dependencies in their learning. In Fig. 2 a visualization of the LCR-Rot-hop-ont++ model is presented. This figure is split into two parts. The lower part of this figure is inspired by K-BERT. On the left side the training process is displayed and on the right side the testing. For training we just use the input sentence. However, for testing we also use the input sentence and the information from the domain ontology to create a sentence tree and contextualized word embeddings. We describe this step more extensively in Sect. 4.2.

The procedure described in the upper part of Fig. 2 goes as follows. First the sentence $s = [w_1, w_2, ..., w_N]$ is split into three parts: the left context $s^l = [w_1^l, w_2^l, ..., w_L^l]$, the target phrase $s^t = [w_1^t, w_2^t, ..., w_T^t]$, and the right context $s^r = [w_1^r, w_2^r, ..., w_R^r]$. Hereby, N is the number of words in sentence s, and L, T, and R are the number of tokens in each part of the sentence, respectively. These three parts are then embedded together using BERT for training data and K-BERT for test data.

The embeddings are then used as input for three separate Bi-LSTMs and give three hidden states h for each part, namely: $[h_1^l, h_2^l, ..., h_L^l]$ for the left context, $[h_1^t, h_2^t, ..., h_T^t]$ for the target phrase, and $[h_1^r, h_2^r, ..., h_R^r]$ for the right context.

Next, the three Bi-LSTMs hidden states are used in a two-step rotatory mechanism which creates in the first step the new context representations using target information. In the first iteration, we use an average pooling mechanism to obtain the target representation $r^{t_p} = \text{pooling}[h_1^t, h_2^t, ..., h_T^t]$.

Furthermore, we calculate the attention scores for each word using a softmax function in order to compute the new context and target representations. We compute the representation of the left context for $i = 1, ..., L$ as follows:

$$f(h_i^l, r^{t_p}) = \tanh(h_i^{l'} \times W_c^l \times r^{t_p} + b_c^l) \ , \tag{1}$$

where W_c^l is a weight matrix and b_c^l is the bias for the left context l.

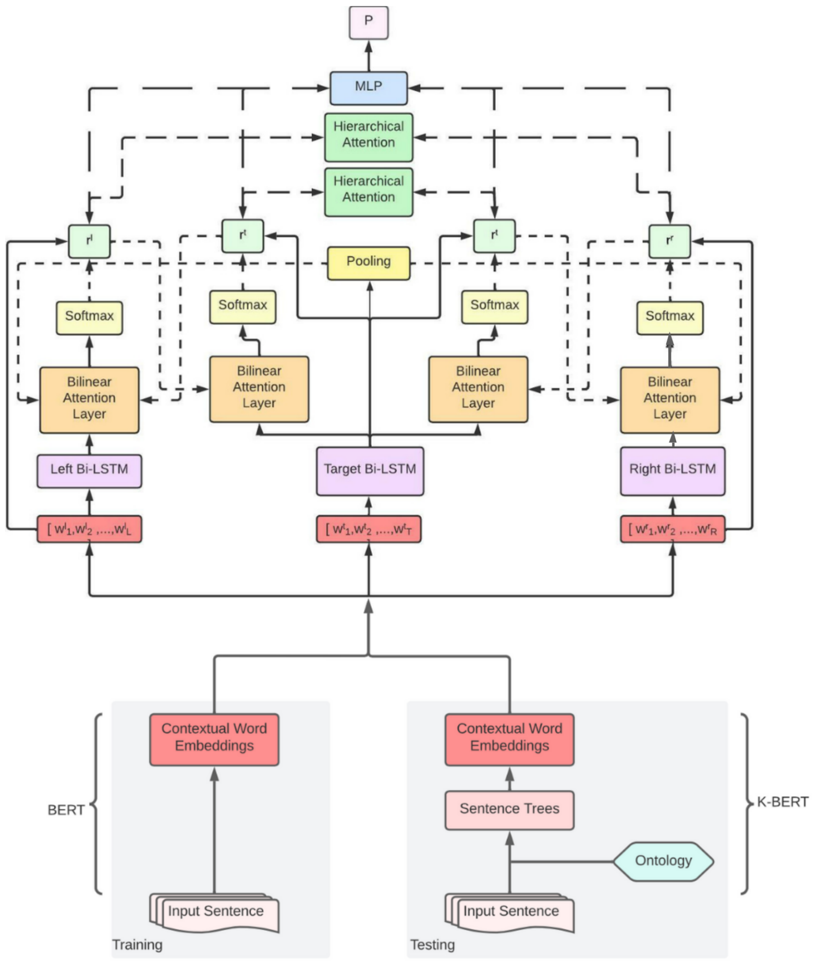

Fig. 2: Visualization of LCR-Rot-hop-ont++.

Next, we normalize the attention scores by using a softmax function, where $i = 1, ..., L$:

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

Furthermore, we calculate the representation of the left context as:

$$r^l = \sum_{i=1}^{L} \alpha_i^l \times h_i^l \quad . \tag{3}$$

In a similar way, we obtain the representation of the right context. In the second step of this rotary mechanism, we determine the target representations w.r.t the left and right context. Here, the target representation w.r.t. the left context is given by:

$$r^{t_l} = \sum_{i=1}^{T} \alpha_i^{t_l} \times h_i^t \quad , \tag{4}$$

where the attention scores are computed between target words and previously computed representations of the left and right context. In the first iteration of the multi-hop method we use the pooled target vector. However, in the iterations thereafter we use the left and right target representations to compute the left and right context representations, respectively. We then use a hierarchical attention model to further improve the representations by including the information on sentence-level as well.

First, we calculate the normalized attention scores using the attention function in (1) for $i = 1, ..., 4$:

$$f(v^i) = \tanh(v^{i'} \times W + b) \quad , \tag{5}$$

where $v^i \in \{r^l, r^{t_l}, r^{t_r}, r^r\}$, W is a weight matrix, and b is a bias. Next, we compute the normalized attention scores as follows:

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

This is done separately for the context vectors (r^l and r^r) and target vectors (r^{t_l} and r^{r_l}). Furthermore, we determine the new representations:

$$v_{new}^i = v_{old}^i \times \alpha^i \quad . \tag{7}$$

We use $n = 3$ hops for the multi-hop approach as this was found optimal in [17]. The final representations are then concatenated, linearized, and squashed to compute the vector of probabilities of length $|C|$, where $|C|$ is the amount of sentiment categories. For this we use the vector v obtained by concatenation of four vectors:

$$v = [r^l; r^{t_l}; r^{t_r}; r^r] \quad , \tag{8}$$

Then, after applying a linear layer this is sent into the softmax function in order to predict the sentiment probability:

$$p = \text{softmax}(W_c \times v + b_c) \ , \tag{9}$$

where p is the conditional probability distribution, W_c is a weight matrix, and b_c is the bias for class c.

To obtain the model parameters that best capture the semantics in the training data, we minimize the cross-entropy loss function which measures the difference in the estimated probability and the true value of the sentiment classification.

In addition, we penalize the complexity of the system by using L$_2$-regularization. The loss formula is:

$$Loss = -\sum_j y_j \cdot \log(\hat{p_j}) + \lambda ||\theta||_2^2 \ , \tag{10}$$

where y_j is a $|C| \times 1$ vector of the true sentiment classification values, $\hat{p_j}$ is the $|C| \times 1$ vector of sentiment classification predictions, λ is the L$_2$-regularization term that penalizes the squared values of the model parameters, and θ is the vector containing all the model parameters, these are the LSTM parameters and $\{W_c^l, b_c^l, W_c^r, b_c^r, W_t^l, b_t^l, W_t^r, b_t^r, W_c, b_c, W, b\}$. The squaring of the model parameters ensures that we penalize the large model parameters more than the small model parameters.

4.2 Knowledge Injection

We inject knowledge from a domain ontology which represents concepts and relations between these concepts. Furthermore, a concept has one or more lexical representations attached which denote synonyms. We define a k-hop as navigating k classes from the current class in the ontology in order to extract lexical representations, for $k = 0, ..., N-1$ with N being the total number of classes. In this case, performing a 0-hop points to synonyms which are then injected into the test input sentence, because we do not have to navigate through the ontology.

We inject the information from the ontology according to the following procedure. First in the K-Query step, we determine for each k-hop all the lexical representations. Furthermore, $w_{i,k}$ denotes the word i in the sentence for which the lexical representations of the subclasses were obtained by navigating k-hops from the current class in ontology O (domain-specific relations from the ontology are ignored in this study due to the possibility of divergent semantics w.r.t. the current concept). Second, the words corresponding to the concepts are added to sentence s. For sentence s, the selection of these words could then be written as:

$$E = \text{K-Query}(s, O, k) \ . \tag{11}$$

The collection of a k-hop is then:

$$E = \cup_{w_i \in s}[(w_i \cup \{w_{i,0}\} \cup ...\{w_{i,k}\})] \ , \tag{12}$$

whereby $\{w_{i,n}\}$ represents the set of all words obtained from word i using n hops in the ontology.

The second step is called K-Inject, where we create a sentence tree t by injecting the queried concepts of the k-hops to the corresponding positions. Next, we enter the embedding layer where we input the new sentence tree and get as output an embedding representation. Since the sentence tree is incompatible with the embedding layer of BERT, we first have to create a new sentence tree. We inject the words, obtained through the concepts by navigating k-hops, directly after the corresponding words in the sentence. Note that if a word from the original sentence is also listed in the ontology, we do not inject this word again in the sentence. We consider injecting synonyms by adding "soft branches" to our sentence tree. We define "hard branches" for the words obtained through the concepts of the other k-hops. Hereby, we only add branches of length one. In Fig. 3 an example of a sentence tree is shown. The original word in the sentence "staff" has a synonym, namely "crew". Furthermore, the words attached to the "hard branches" which represents the 1-hops, are "Chef", "Waiter", and "Host". Also, the words obtained by performing a 1-hop could have synonyms themselves given by corresponding "soft branches".

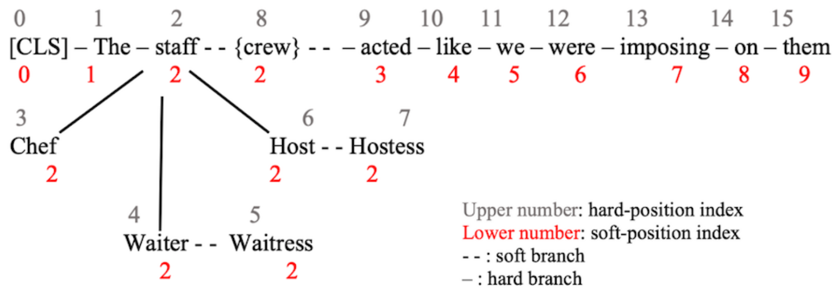
0 1 2 8 9 10 11 12 13 14 15
[CLS] – The – staff - - {crew} - - – acted – like – we – were – imposing – on – them
0 1 2 2 3 4 5 6 7 8 9
3 Chef 2
4 Waiter - - 5 Waitress
2 2
6 Host - - 7 Hostess
2 2
Upper number: hard-position index
Lower number: soft-position index
- - : soft branch
– : hard branch

Fig. 3: Example of a sentence tree.

The embedding representation is a sum of three parts: the token embedding, soft-position embedding, and the segment embedding. The token and segment embeddings are the same as in BERT. However, the main difference is in the soft-position embedding, because we cannot input the new sentence tree into the embedding layer. As mentioned, we have to place the words obtained through the concepts of performing k-hops in the original sentence at specific places which ruins the structure of the original sentence. This problem could be solved by using soft-positioning in which the lexical representations are given the same soft-position as the original word in the sentence, displayed by the red numbers (below text numbers for black and white printing) in Fig. 3. This way the structural information of the sentence is retained, however, this might also lead to false semantic information.

To solve the previously identified problem, we use the visible matrix M. This makes sure that the k-hop of the original word does not affect the hidden state value of this word that is not in the same branch of the sentence tree. We create a visible matrix by using the hard-position indexes, displayed by the gray numbers (above text numbers for black and white printing) in Fig. 3. We define the visible matrix as:

$$M_{ij} = \begin{cases} 0, & \text{if } w_i \oplus w_j \\ -\infty, & \text{if } w_i \otimes w_j \end{cases}, \tag{13}$$

where $\oplus$ means that the words are in the same branch and thus are visible to each other and $\otimes$ means the words are not visible to each other. Furthermore, i and j are the hard-position indexes. Thus, it holds that all words in the original sentence are visible to each other. Furthermore, the synonyms are only visible to the words they belong to and the words obtained from performing the k-hops are not able to see each other unless they are in the same branch. In Fig. 4 an example of a visible matrix is shown which has as input the sentence tree of Fig. 3. The numbers on the sides are the hard-position indexes, the red dots (gray dots for black and white printing) indicate that the words are visible to each other and the white dots indicate that they are invisible. For example, the word "staff" of the original sentence with hard-position 2, is visible to the words obtained by performing k-hops and the other words of the original sentence. It is, however, unable to see the synonyms "Waitress" and "Hostess" with hard-positions 5 and 7, respectively.

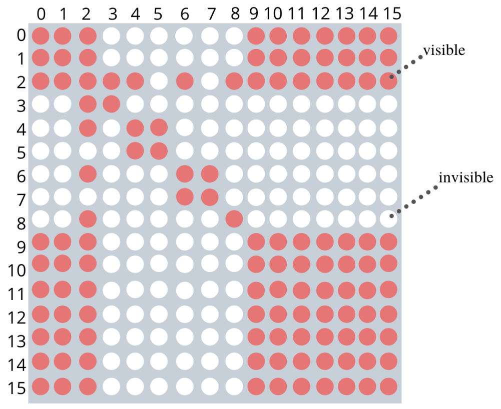

Fig. 4: Example of a visible matrix.

Next, we apply the BERT-encoder where we change the multi-head attention block by adding the visible matrix M in (15). The input of the first multi-head attention block of the first layer is $h^0 = \text{sum}(E_t, E_s, E_{seg})$, where E are the embeddings of the tokens, soft-positions, and segments, respectively. The input plus output of this block is then the input for the first normalization layer in the BERT-encoder. Furthermore, the output of this first normalization layer is then the input of the feed-forward layer. Then, the input plus output of the feed-forward layer is used as the input of the second normalization layer. The output of the second normalization layer is then used as the input for the next encoder layer. This process is repeated until we arrive at the last encoder layer and set that output as hidden states for all tokens. These steps are repeated for all twelve encoder layers. In the next three equations the multi-head attention block is given where we add the visible matrix M (13).

$$Q^{i+1}, K^{i+1}, V^{i+1} = h^i W_q, h^i W_k, h^i W_v \ , \tag{14}$$

$$S^{i+1} = \mathrm{softmax}\left(\frac{Q^{i+1}{K^{i+1}}^{\top} + M}{\sqrt{d_k}}\right) \ , \tag{15}$$

$$h^{i+1} = S^{i+1} V^{i+1} \ , \tag{16}$$

where Q, K, and V are matrices of queries, keys, and values, respectively. Furthermore, h^i and h^{i+1} denote the input and output of the multi-head attention block, respectively. In addition, W_q, W_k, and W_v are trainable model parameters and d_k is a scaling factor. Now, we are able to identify that if the words k and j are not in the same branch, S_{kj}^{i+1} is set to zero. This means that word k does not make a contribution to the hidden state of word j. This process is repeated for the whole sentence. Next, the sum of the outputs of the last four encoder layers are split into the three parts that correspond to the left context, target, and right context of the sentence. These are then each used as input for the Bi-LSTMs as was showed in Fig. 2.

It is possible to do knowledge injection at training time and at test time. However, by injecting knowledge at training time we add lexical representations to the sentences which could lead to sentences with a different structure and thus a different meaning. This in turn could lead to different embeddings and thus different predictions at test time. If the knowledge is only injected at test time then we just have to change the embeddings and hidden states in K-BERT with the use of soft-positioning and visible matrices. One can note that the model training remains unchanged, the only changes are being made for testing.

5 Results

In the section we report the results of our model LCR-Rot-hop-ont++ which makes use of soft-positioning and visible matrices. To evaluate the performance of our model we have used the SemEval 2015 Task 12 and SemEval 2016 Task 5 datasets. Furthermore, we compare the accuracy of our model with the accuracy of two benchmarks. These are HAABSA++ and LCR-Rot-hop++. In this paper we adopt a conservative approach and focus on 0-hops as these affect the least the semantics of the original sentences. By performing 0-hops, synonyms are injected into the input sentences. In Table 2 the test accuracies of the LCR-Rot-hop-ont++ model and its benchmarks are presented for both the SemEval 2015 and SemEval 2016 datasets. We do not replicate the HAABSA++ and LCR-Rot-hop++ work and thus give the results as reported in the previous work [17].

In Table 2 is given that for the SemEval 2015 dataset LCR-Rot-hop-ont++ outperforms HAABSA++ and LCR-Rot-hop++ in terms of test accuracy by 0.2% and 0.8%, respectively. In addition, for the SemEval 2016 dataset LCR-Rot-hop-ont++ outperforms LCR-Rot-hop++ by 0.2 percentage points, but LCR-Rot-hop-ont++ has a lower performance than HAABSA++ (0.1 percentage points). As SemEval 2015 is a smaller dataset than SemEval 2016 we conclude that our proposed approach helps boost the performance of a neural model where for smaller datasets knowledge injection is preferred over the hybrid approach.

Table 2: Comparison of LCR-Rot-hop-ont++ to its benchmarks using accuracy (the bold font indicates the best results).

	SemEval 2015	SemEval 2016
LCR-Rot-hop-ont++	**81.9%**	86.9%
HAABSA++	81.7%	**87.0**%
LCR-Rot-hop++	81.1%	86.7%

6 Conclusion

In this research, we propose the LCR-Rot-hop-ont++ model for performing Aspect-Based Sentiment Classification (ABSC) on a sentence-level of restaurant reviews. The model is formed by injecting the information from a domain ontology into the state-of-the-art neural network model LCR-Rot-hop++ in our test instances. This approach of knowledge injection is based on the language representation model K-BERT which also makes use of soft-positioning and visible matrices. These are used to control the scope of the injected knowledge and thus help to retain the meaning of the original sentences. Furthermore, the performance of the LCR-Rot-hop-ont++ model was evaluated by using the standard SemEval 2015 and SemEval 2016 datasets for ABSC. We have compared our model to its benchmarks, namely HAABSA++ (ont+LCR-Rot-hop++) and LCR-Rot-hop++.

Hereby, we conclude that LCR-Rot-hop-ont++ boosts the testing accuracy for the SemEval 2015 dataset, namely from 81.7% for HAABSA++ and 81.1% for LCR-Rot-hop++ to 81.9%. Furthermore, for the SemEval 2016 dataset we conclude that LCR-Rot-hop-ont++ outperforms LCR-Rot-hop++, but does not outperform HAABSA++. Therefore, we conclude that knowledge injection is effective for the neural network to perform sentiment classification, especially for smaller datasets.

For future work, we propose to further explore the amount of information that can be injected from the ontology into the neural network. We have only researched knowledge injection for a 0-hop, but perhaps injecting more information could boost the performance of the neural network more for sentiment classification. In addition, we could try different percentages of knowledge injection, e.g., 20%, 40%, 60%, 80%, and 100%. Furthermore, we could analyze the characteristics of sentences that can benefit the most from knowledge injection. For example, it is expected that shorter sentences could benefit the most from knowledge injection for building additional context information.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2016). pp. 265–283. USENIX Association (2016)
2. 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)
3. Brauwers, G., Frasincar, F.: A survey on aspect-based sentiment classification. ACM Computing Surveys **55**(4), 65:1–65:37 (2023)
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: 17th Conference of the North American Chapter of the Association of Computational Linguistics: Human Language Technologies (NAACL-HLT 2019). pp. 4171–4186. ACL (2019)
5. Dragoni, M., Poria, S., Cambria, E.: OntoSenticNet: A commonsense ontology for sentiment analysis. IEEE Intelligent Systems **33**(3), 77–85 (2018)
6. Liu, B.: Sentiment Analysis: Mining Opinions, Sentiments, and Emotions. Cambridge University, 2nd edn. (2020)
7. Liu, Q., Zhang, H., Zeng, Y., Huang, Z., Wu, Z.: Content attention model for aspect based sentiment analysis. In: 27th International World Wide Web Conference (WWW 2018). pp. 1023–1032. ACM (2018)
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. 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)
10. Meškelė, D., Frasincar, F.: ALDONAr: A hybrid solution for sentence-level aspect-based sentiment analysis using a lexicalized domain ontology and a regularized neural attention model. Information Processing and Management **57**(3), 102211 (2020)

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. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: 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)
13. Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., AL-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., De Clercq, O., Hoste, V., Apidianaki, M., Tannier, X., Loukachevitch, N., Kotelnikov, E., Bel, N., Jiménez-Zafra, S.M., Eryiğit, G.: SemEval-2016 task 5: Aspect-based sentiment analysis. In: 10th International Workshop on Semantic Evaluation (SemEval 2016). pp. 19–30. ACL (2016)
14. 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)
15. Schouten, K., Frasincar, F.: Survey on aspect-level sentiment analysis. IEEE Transactions on Knowledge and Data Engineering **28**(3), 813–830 (2015)
16. 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)
17. Trusca, 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)
18. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: 31st Annual Conference on Neural Information Processing Systems (NIPS 2017). vol. 30, pp. 5998–6008. Curran Associates (2017)
19. 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)
20. 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., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A.: 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)
21. Yan, X., Jian, F., Sun, B.: SAKG-BERT: Enabling language representation with knowledge graphs for Chinese sentiment analysis. IEEE Access **9**, 101695–101701 (2021)
22. Zhao, A., Yu, Y.: Knowledge-enabled BERT for aspect-based sentiment analysis. Knowledge-Based Systems **227**, 107220 (2021)
23. Zheng, S., Xia, R.: Left-center-right separated neural network for aspect-based sentiment analysis with rotatory attention. arXiv preprint arXiv:1802.00892 (2018)