Curriculum Learning for a Hybrid Approach for Aspect-Based Sentiment Analysis

Nana Lange[a], Flavius Frasincar[a], Maria Mihaela Truşcă[b]

[a]*Erasmus University Rotterdam, P.O. Box 1738, 3000DR, Rotterdam, the Netherlands*
[b]*KU Leuven, Blijde Inkomststraat 21, 3000 Leuven*

Abstract

In the past years, the amount of unstructured online review data has grown exponentially. Many people express their opinions about different aspects of goods and services on the Web. Aspect-Based Sentiment Analysis (ABSA) automatically extracts the sentiments with respect to aspects given in a sentence. We improve the training procedure of the state-of-the-art Hybrid Approach for Aspect-Based Sentiment Analysis with deep contextual word embeddings and hierarchical attention (HAABSA++). In this method, a domain sentiment ontology is used as a main classifier, and if it is not conclusive, a neural network is employed as a back-up. We extend the training of the neural network by incrementally adding more difficult instances, also known as curriculum learning. Restaurant reviews obtained from the SemEval-2015 and SemEval-2016 datasets are used to evaluate the effect of implementing curriculum learning. Using *baby steps* curriculum learning and a specific curriculum strategy, the accuracy of HAABSA++ is improved from 86.3% to 87.5%.

Keywords: aspect-based sentiment analysis, baby steps curriculum learning, one-pass curriculum learning, online reviews

1. Introduction

In the past years, the amount of unstructured online review data has grown exponentially. Many people express their opinions about different aspects of goods and services on the Web. Since review data is very unstructured and the number of reviews can be very large, it can be expensive, time-wise and price-wise, to extract the consumer's opinions manually from the reviews, especially for big companies. That is why sentiment analysis is often used. Sentiment analysis is a discipline that is able to detect sentiments in a full review, a sentence, or with respect to an aspect (Liu, 2015). Given a sentence in a review, it is possible that in the same sentence an opinion about two or more aspects is given. The sentiments towards these aspects can be unequal, even though they are

Email addresses: nanalange@planet.nl (Nana Lange), frasincar@ese.eur.nl (Flavius Frasincar), mariamihaela.trusca@kuleuven.be (Maria Mihaela Truşcă)

mentioned in the same sentence. Aspect-Based Sentiment Analysis (ABSA) makes it possible to find the sentiment of different aspects in the same sentence. In this way, the sentiments of all aspects in a sentence are determined, instead of only the overall sentiment of the complete sentence (Schouten & Frasincar, 2016). Focusing on the individual aspects rather than full sentences gives a more profound insight into the different aspects of products or services. A sentiment can be positive, negative, or neutral, and can give a company useful information about their products. ABSA is therefore a very important tool for companies to get an insight into how people evaluate certain aspects of their products and which product features could be improved.

In order to accurately predict the sentiment of a given aspect, both machine learning methods and knowledge-based methods can be used. However, Schouten & Frasincar (2018) have shown that using both methods in a hybrid approach yields the best performance. In this approach, first, the sentiment of an aspect is predicted by a domain sentiment ontology, and a support vector machine is used as a back-up. Wallaart and Frasincar have extended this method and have shown that this approach works best with a neural network with a rotatory attention mechanism and multiple hops (LCR-Rot-hop) (Wallaart & Frasincar, 2019). This approach will from now on be referred to as Hybrid Approach for Aspect-Based Sentiment Analysis (HAABSA). In addition to this, Trusca et al. (2020) have shown that the accuracy of HAABSA is improved by using deep contextual word embeddings and hierarchical attention (LCR-Rot-hop++), resulting in the HAABSA++ model.

The domain sentiment ontology approach is able to predict the sentiment of around 60% of the samples (Wallaart & Frasincar, 2019), with an accuracy of 86.8%. The neural network is thus needed to predict the sentiment of the remaining sentences in the dataset. The sentiment of the remaining sentences is predicted by the neural network with an accuracy of 81.5%. While it is clear why the backup neural network is of great importance, the domain sentiment ontology is required due to its capacity to provide better results than the neural network. The training of a deep learning model, however, can be a very time-consuming job. Bengio et al. (2009) discovered that presenting the training data to a model in a predetermined order, usually from the easiest to the most difficult, can improve its learning speed. This so-called curriculum learning can help the training process by converging faster to better solutions (Bengio et al., 2009). It is shown that the use of curriculum learning can be effective when applied to deep learning in text (Tsvetkov et al., 2016; Cirik et al., 2016). Cirik et al. (2016) show that ordering the samples based on their lengths, leads to improved sentiment analysis results. Recently, Rao et al. (2020) have specialized the sorting of data by proposing a curriculum strategy that is focused specifically on sentiment analysis, instead of a more general strategy. They show that ordering the training data while accounting for the downstream task leads to better results than when the data is ordered utilizing a general approach. Hence, the application of curriculum learning in sentiment analysis has shown promising results in recent studies. However, little research has been done into how curriculum learning can improve the

results of the model for ABSA.

In this research, we focus on the sentiment analysis of aspects and we do not conduct any research into the detection of these aspects. That is why data with already annotated aspects is used in our research. We aim to extend the state-of-the-art HAABSA++ model given by Trusca et al. (2020) and improve their results. To be more precise, we focus on improving the training of the neural network of the HAABSA++ approach in order to obtain faster and possibly more accurate results.

Since curriculum strategies designed specifically for the downstream tasks lead to better results than general strategies (Nagatsuka et al., 2021; Zhu et al., 2021), we aim to find a specific curriculum strategy for ABSA. Even though studies have shown that applying a curriculum strategy can be very successful in sentiment analysis (Rao et al., 2020), it is not clear what the possible effects are when applied to ABSA. To achieve this aim, we use curriculum learning for training the neural network of the HAABSA++ model (LCR-Rot-hop++). Besides the already demonstrated benefits of the curriculum learning for the NLP tasks, another reason to apply this learning strategy with the hybrid approach is the presence of the attention mechanism in the neural network. Starting from the statement of Vijjini et al. (2021) according to which curriculum learning enhances the effectiveness of the attention layers, we need to confirm this hypothesis with respect to the neural network of the hybrid approach. All of the above leads us to the following research question:

How can curriculum learning improve an approach for aspect-based sentiment analysis?

To answer this question, we use the curriculum strategy proposed by Rao et al. (2020) for sentiment analysis and alter this strategy to make it appropriate for ABSA. We compare the results of the model trained with and without curriculum learning and investigate different ways of implementing our curriculum strategy. The data we use consists of a collection of restaurant reviews and all the methods are implemented in *Python*. The source code can be found via: `https://github.com/NanaLange/CL-HAABSA`.

The remainder of this paper is structured as follows. To begin with, in Section 2, a discussion of the relevant literature into ABSA and curriculum learning is given. Section 3 gives an overview of the methods we use. Subsequently, the used datasets and the evaluation measures are discussed in Section 4. The results of our proposed methods are given in Section 5. Finally, we end our paper by giving a conclusion in Section 6.

2. Related Work

An overview of previous research into ABSA is provided in Section 2.1. After this, in Section 2.2, we discuss the concept of curriculum learning and give an overview of studies on curriculum learning in sentiment analysis.

2.1. Aspect-Based Sentiment Analysis

ABSA is broadly covered in previous research (Weichselbraun et al., 2017; Meskele & Frasincar, 2020; Yadav et al., 2021; Cheruku et al., 2024; Han et al., 2025; Xu et al., 2025). The study of Schouten & Frasincar (2016) provides an overview of research into ABSA, focusing on both the detection of aspects and the sentiment analysis of these aspects. As the data we use in our research already provides us with the aspects, we focus solely on the classification of the sentiment belonging to the aspects, by assigning a positive, negative or neutral sentiment label.

Schouten & Frasincar (2016) state that the ABSA approaches can in general be classified as knowledge representation-based or as machine learning-based. In addition to this, their study also proves that these two methods are actually complementary and that a hybrid approach using both methods outperforms approaches that rely on only one. According to Wallaart & Frasincar (2019), HAABSA consists of an ontology reasoner and a neural network. The ontology is domain-specific and sentiment-relevant and in the performed experiment it is designed for restaurant reviews. In cases where the prediction of the ontology is indecisive (conflicting sentiment or missing sentiment), a neural network is used.

In the HAABSA method, the neural network introduced by Wallaart & Frasincar (2019) is based on an extension of the Left-Center-Right separated neural network with Rotatory attention (LCR-Rot) proposed in (Zheng & Xia, 2018). LCR-Rot has shown to be very successful in predicting the sentiment of an aspect. The LCR-Rot model divides the sentence into three parts: the left context, the target phrase, and the right context. The target phrase contains the words that form the aspect. The model consists of three separate Long Short-Term Memory (LSTM) models, each LSTM for one of the parts of the sentence. Next to this, a rotatory attention mechanism is used to better model the interaction between the target and the left and right context. In this way, the LCR-Rot model uses the different parts of the sentence to capture the most important sentiment words.

The LCR-Rot-hop model presented by Wallaart & Frasincar (2019) extends the LCR-Rot model by performing multiple hops in the rotatory attention. Therefore, information about the different parts of the sentences is hereby used and updated multiple times in the model in order to obtain better results. Finally, Trusca et al. (2020) introduced HAABSA++ by extending LCR-Rot-hop++ as a model that combines hierarchical attention with the LCR-Rot-hop model. The authors have added an extra layer to the LCR-Rot-hop model so that not only local information is used when the vectors are updated, but also information at the sentence level is considered. In addition to hierarchical attention, Trusca et al. (2020) use deep contextual word embeddings, replacing the non-contextual word embeddings of HAABSA.

In parallel, recent work has focused on integrating neurosymbolic AI into sentiment analysis, with the goal of enabling personalization (Zhu et al., 2024) or improving interpretability and the reasoning capabilities. Neurosymbolic approaches such as SenticNet 8 (Cambria et al., 2024) and

SenticVec (Zhang et al., 2024) define hybrid models that combine symbolic logic, such as rules or affective commonsense knowledge, with neural networks. These models offer new opportunities to explain and understand sentiment analysis systems where the decision is sometimes opaque (Diwali et al., 2024). By leveraging symbolic representations for knowledge-based emotional reasoning, these approaches improve explainability without affecting performance, making them suitable for real-world applications.

Considering that HAABSA++ (Trusca et al., 2020) aligns with these recent trends, as it is a hybrid model that combines symbolic reasoning with deep learning, and given its superior performance compared to other approaches, we adopt HAABSA++ as the starting point of our research.

2.2. Curriculum Learning

It is known that humans and animals learn much better when instances are presented to them from most easy to most difficult, instead of in a random order (Skinner, 1958; Krueger & Dayan, 2009). Therefore, the basic idea of curriculum learning is to first train the model on the easier parts of a task, increasing the difficulty step-by-step (Elman, 1993).

Bengio et al. (2009) have researched some basic approaches to check whether machine learning models could benefit from curriculum learning. Their results suggest that applying appropriate curriculum strategies can lead to better training results due to the faster convergence to better solutions. In addition, they show that curriculum learning regularizes, leading to lower generalization error for the equal training error.

Given the topic of our paper, we are interested in how curriculum learning can be applied to deep learning in text. This specific field of research has not been investigated to a great extent yet. Cirik et al. (2016) have investigated the effect of curriculum learning on LSTM networks. LSTMs are used in many different neural networks in the Natural Language Processing (NLP) field. In their research, the authors study the effect of two different curriculum learning regimens: the *one-pass* curriculum and the *baby steps* curriculum.

The *one-pass* curriculum is an algorithm suggested by Bengio et al. (2009), where the training instances are ordered by a curriculum. The ordered training data is then divided into k number of buckets. First, the model is trained on the simpler instances, contained in the first bucket. When the model's loss or task accuracy criteria on a held-out set do not get better after p number of epochs, the bucket is discarded and the next bucket is selected. The model is then trained with this new bucket in the same way. The training of the model stops when all buckets are used. This algorithm is called *one-pass* curriculum because of the fact that the model uses each bucket only one time for the training, making one-pass through the data.

The *baby steps* curriculum of Spitkovsky et al. (2010) extends the former curriculum by incrementally adding more complex sets of training instances to the training set, instead of using every

set separately. Again, the data is divided into k buckets based on the complexity of the instances. The model is first trained on the bucket with the easiest data and when the model does not get any better after p epochs, the next bucket is added to the training data. In this algorithm, the amount of training instances presented to the model is thus incrementally increased based on the difficulty of the instances, hence the name. The training stops when all the data is used. The difference between the *one-pass* curriculum and the *baby steps* curriculum is that the later curriculum strategy cumulates the buckets once the difficulty is increased in the training process, while the former strategy discards them.

Xu et al. (2020) split the training data into k buckets and train a BERT model for each of them. Next, each BERT model is tested for the remaining $k-1$ buckets meaning that there are $k-1$ results for each training instance. In the end, the difficulty arrangement of the training instances is done based on their average performance obtained over the $k-1$ buckets. The approach is tested using *one-pass* curricula with a BERT model for different tasks included in the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019). Since the topic of our paper is related to sentiment analysis, we observed that the curriculum usage of Xu et al. (2020) for this topic does not outperform the baseline without the curriculum strategy.

Cirik et al. (2016) order their data from shorter to longer sentences, based on the assumption that shorter sequences are easier to learn than longer ones (Spitkovsky et al., 2010). The authors use the *one-pass* and *baby steps* curricula to study the effect of curriculum learning on LSTM networks, where they pay extra attention to sentiment analysis. They found out that the *baby steps* curriculum performs significantly better than the other approaches. They show that the *baby steps* curriculum works particularly well in case of contrasting conjunctions, where two contrary signals come from two different directions, i.e., the left and right phrases.

Similar to the work of Cirik et al. (2016), Nagatsuka et al. (2021) create multiple buckets based on the length of the input instances. The new data buckets are used for pre-training RoBERTa, introduced as a variant of BERT by Liu et al. (2019). Considering again the GLUE downstream tasks, the curriculum strategy improves the performance of the fine-tuned RoBERTa model in most cases. Additionally, Nagatsuka et al. (2021) examine the anti-curriculum learning as a strategy where the buckets are organised from the most difficult to the easiest (i.e., from longer instances to the shortest instances). Even if, the anti-curriculum strategy is less effective than the conventional learning, it outperforms the baseline without any curriculum strategy.

Given the past research, it is clear that curriculum learning can be very effective in sentiment analysis. Rao et al. (2020) claim that the curriculum strategies that have been proposed so far are all based on the difficulty of the dataset, irrespective of the task that the data will be used for (in our case sentiment analysis). That is why the authors propose a curriculum strategy that uses information about the sentiment level of the samples in order to sort them based on the *baby steps*

srategy. Their task relied on the lexical resource SentiWordNet that assigns positivity, negativity, and objectivity scores to every synset (Esuli & Sebastiani, 2006; Baccianella et al., 2010) found in the WordNet lexical database (Miller, 1995).

In order to determine the effect of the SentiWordNet strategy, Rao et al. (2020) also perform a different curriculum strategy, where the ordering is based on sentence length (as is done by Cirik et al. (2016)). Based on this setup, they show that the SentiWordNet strategy performs better than the strategy based on sentence length. They state that this result can be explained by the fact that the sentence length strategy defines the complexity of sentences more generally, instead of focusing specifically on sentiment analysis. The SentiWordNet strategy, on the other hand, ranks the training data based on how difficult it is to assign a sentiment and is thus much more specific. However, the study of Rao et al. (2020) focuses on sentiment analysis, whereas our study focuses on ABSA. Given the fact that ABSA gives deeper insight into the characteristics of a product than general sentiment analysis, it is therefore very interesting to find out whether this curriculum strategy leads to improved results as well when we focus on aspects rather than sentences.

The aforementioned curriculum learning algorithms all have in common that they begin with the simpler examples and gradually extend to the more difficult ones. In order to decide the best curriculum learning approach, Cirik et al. (2016) and Rao et al. (2020) have analyzed and compared different algorithms. Tsvetkov et al. (2016) deviate from this approach of testing and comparing a handful of curricula, by searching for an optimal curriculum utilizing Bayesian optimization. In their research, the authors order the paragraphs in the dataset based on their paragraph score, which is given by the linear function $\mathbf{w}^\top \phi(X)$. In this function, X represents the training corpus consisting of n paragraphs and $\phi(X)$ is a vector that contains a value for each of the following features for every paragraph: diversity, simplicity, and prototypicality. Finally, $\mathbf{w}$ represents the weights learned for these features. The feature weights are optimized using Bayesian optimization. In this way, curriculum learning proposed by Tsvetkov et al. (2016) is thus treated as an optimization problem, instead of shuffling the data according to human intuitions.

In the work of Tsvetkov et al. (2016), the curriculum using Bayesian optimization is compared with other curricula created using certain heuristics. The authors evaluate these approaches using four different well-known NLP tasks, including sentiment analysis. They show that using a curriculum leads to significantly better results than when no curriculum is used, which is in line with previous research. Next to this, the optimized curricula for the different NLP models are investigated to see which features are most important. They found out that for sentiment analysis, the best curriculum is sorting by prototypicality features. This is a set of semantic features that uses information from child language acquisition and cognitive linguistics. The authors first compute these features for every word after tokenization, and then average them over the sentences.

Considering the previous research, curriculum learning thus seems promising for our research.

We are interested in starting with the samples that are easy to classify and then gradually add the more difficult samples to the training data. Interestingly, a method exists that focuses more on the difficult instead of the easy instances. This method is called boosting and it assigns weights to the samples such that the classifiers focus on the samples that are hard to classify (Tan et al., 2005). After every boosting round, the weights of the training instances are updated and used during the next round. Multiple implementations of the boosting algorithm exist, differing on how the weights of the training samples are updated and how the predictions of the classifiers are combined. One well-known implementation is AdaBoost (Freund & Schapire, 1997).

3. Methodology

Our work is dedicated to improving the backup neural network of the HAABSA++ approach based on curriculum learning. According to the initial setup of the HAABSA++ method, an ontology is used first to predict the sentiments of the aspects, and the neural network backs up only if the ontology is indecisive. By doing so, the ontology typically classifies straightforward instances with clear positive or negative polarity. Instances with more complex, ambiguous sentiment polarities, as well as those labeled as neutral, remain unresolved by the ontology and are therefore handled by the neural network. Alternative approaches for addressing ambiguity and neutrality in sentiment analysis have been proposed by Wang et al. (2020) and Sonawane & Kolhe (2022). Wang et al. (2020) introduce a multi-level attention mechanism to capture neutrality when both positive and negative scores are either low or similar in magnitude. Similarly, Sonawane & Kolhe (2022) treat neutrality as part of the ambiguity spectrum, determining it via an ensemble of classifiers that assign the neutral class when predictions are uncertain or conflicting. While these methods represent promising research directions, our work focuses on applying curriculum learning to sentiment analysis, and thus we continue with the standard multi-label classification task that is able to detect all three sentiment labels: positive, negative, and neutral.

3.1. Ontology Reasoner

The ontology reasoner used in HAABSA++ is based on the study of Schouten & Frasincar (2018). This ontology reasoner uses predefined classes and relations between classes in order to predict the sentiment of the aspects. The tree main classes of the ontology are *SentimentValue*, *AspectMention*, and *SentimentMention.* The *SentimentValue* class contains the subclasses *Positive* and *Negative*, which are assigned to, respectively, positive and negative expressions. The neutral sentiment was not modeled in the ontology due to the inherent ambiguity of this sentiment class. The *AspectMention* class identifies the aspects in the data that are related to the discovered sentiments. Last, the *SentimentMention* class models the sentiment expressions.

Using these classes, the ontology reasoner identifies positive or negative sentiments in case that all sentiment expressions related to an aspect point to positive or negative sentiments, respectively. In the event that both sentiments are predicted for the same aspect or when no sentiment is given, the neural network is used as a backup. This model is discussed in the next section.

3.2. Multi-Hop LCR-Rot with Hierarchical Attention

For our neural network backup model, we use the LCR-Rot-hop++ proposed by Trusca et al. (2020). To define this model, we use the definitions given by Zheng & Xia (2018). To begin with, we display every sentence s with N words as $s = [w_1, w_2, .., w_N]$. Each sentence is then divided into the following three parts: the left context $[w_1^l, w_2^l, .., w_L^l]$, the target phrase $[w_1^t, w_2^t, .., w_M^t]$, and the right context $[w_1^r, w_2^r, .., w_R^r]$. L, M, and R are the lengths of the tree parts, respectively, and sum up to N. We use BERT Base word embeddings with dimension 768. From now on, the dimension of the word vectors is denoted by d.

The LCR-Rot-hop model consists of three Bi-directional Long-Short-Term-Memory (Bi-LSTM) modules, one left-, one center-, and one right-Bi-LSTM. These three Bi-LSTMs model, respectively, the left context, the target phrase, and the right context. The input of each Bi-LSTM is represented by the words of that specific part, represented as word embeddings of dimension 768×1. After feeding these word embeddings to the Bi-LSTM, three hidden states are returned: $[h_1^l, h_2^l, .., h_L^l]$ for the left context, $[h_1^t, h_2^t .., h_M^t]$ for the target phrase, and $[h_1^r, h_2^r, .., h_R^r]$ for the right context. Since we use Bi-LSTMs, the module propagates both backwards and forwards, resulting in a hidden state vector that is twice the initial size (300). Because of this fact, the dimension of the hidden state vector is 600×1.

After obtaining the three hidden states, a two-step rotatory attention mechanism is applied to these hidden states outputs to capture the most indicative words in the target phrase and the left and right contexts. In the first step, the most indicative sentiment words in the left/right context are captured, using target information. These results are then used in the second step, where the most indicative words of the target phrase with respect to the contexts are captured.

Step 1: Target2Context Attention

In this step, an average representation of the target phrase is used to obtain better representations of the two contexts. To obtain this average representation of the target phrase, an average pooling operation is utilized, as proposed by Zheng & Xia (2018):

$$\underset{2d\times 1}{r^{t_p}} = pooling([\underset{2d\times 1}{h_1^t}, \underset{2d\times 1}{h_2^t}, .., \underset{2d\times 1}{h_M^t}]). \tag{1}$$

Then, an attention function f is created that takes the average target phrase r^{t_p} and the hidden states of each word in one of the contexts. Looking at the left context for example, we define the

attention function f as:

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

where h_i^l is the ith hidden state for $i = 1, .., L$, W_c^l a weight matrix, b_c^l a bias term, and *tanh* a nonlinear function. The attention scores f are then put into a *softmax* function to obtain normalized attention scores. Again looking at the left context for example, the normalized attention scores α_i^l are computed as follows:

$$\alpha_i^l = \frac{exp(f(h_i^l, r^{t_p}))}{\sum_{j=1}^{L} exp(f(h_j^l, r^{t_p}))}. \tag{3}$$

Finally, the normalized attention scores are used to compute a weighted combination of the hidden states in order to represent the left context:

$$\underset{2d\times1}{r^l} = \sum_{i=1}^{L} \underset{1\times1}{\alpha_i^l} \times \underset{2d\times1}{h_i^l}. \tag{4}$$

The representation r^r for the right context can be retrieved in a similar way by following Equations (2)-(4).

Step 2: Context2Target Attention

In this step, the representation of the target phrase is improved by using the left and right context representations, respectively, r^l and r^r, obtained in the previous step. The target representations are computed similar to Equations (2)-(4), with the only difference that r^l and r^r are used instead of the representation vector r^{t_p}. Following the equations, we get a left-aware target representation, r^{t_l}, and a right-aware target representation r^{t_p}. Again taking the left context as example, the left-aware target representation is then given by:

$$\underset{2d\times1}{r^{t_l}} = \sum_{i=1}^{T} \underset{1\times1}{\alpha_i^{t_l}} \times \underset{2d\times1}{h_i^t}. \tag{5}$$

Since the LCR-Rot model contains a multi-hop rotatory attention mechanism, the given two steps are repeated sequentially for n times. According to Wallaart & Frasincar (2019), the optimal iteration number is three. It is important to realize that the average target vector r^{t_p} is only utilized during the first iteration and is replaced in the next iterations by r^{t_l} or r^{t_p}, depending on the context.

After performing the iterations over the rotatory mechanism, a final presentation for the sentence is obtained by concatenating the left- and right-context representations, respectively, r^l and r^r, and the left- and right-aware target representations, respectively, r^{t_l} and r^{t_r}:

$$\underset{8d\times1}{v} = [\underset{2d\times1}{r^l}; \underset{2d\times1}{r^{t_l}}; \underset{2d\times1}{r^{t_r}}; \underset{2d\times1}{r^r}]. \tag{6}$$

Since LCR-Rot-hop has the disadvantage that the four representation vectors are computed utilizing only local information, Trusca et al. (2020) propose including hierarchical attention by adding an extra layer to the model. The authors present a high-level representation of the input sentence that updates every representation vector with a relevance score computed at the sentence level.

In order to use this hierarchical attention, v is updated after every iteration Trusca et al. (2020). To do this, we begin with computing an attention function f as follows:

$$f(\underset{1\times1}{v^i}) = tanh(\underset{1\times2d}{v^{i'}} \times \underset{2d\times1}{W} + \underset{1\times1}{b}), \tag{7}$$

where v^i is the ith representation of the input sentence ($v^i \in \{r^l, r^{t_l}, r^{t_r}, r^r\}$, i=1,..,4), W a weight matrix, and b a bias. The attention function is then used to apply the attention weighting separately on the intermediate context and target vector pairs. The left and right context attention scores are given by α^1 and α^4, respectively, and can be computed as follows:

$$\alpha^{1,4} = \frac{exp(f(v^{1,4}))}{\sum_{j=1,4} exp(f(v^j))}, \tag{8}$$

with $\alpha^1 + \alpha^4 = 1$. The two target vector attention scores, α^2 and α^3, can be computed in the same way. For these scores we see as well that $\alpha^2 + \alpha^3 = 1$.

These attention scores are then used to compute the new scaled left- and right-context representations and the two target representations:

$$\underset{2d\times1}{v^i} = \underset{1\times1}{\alpha^i} \times \underset{2d\times1}{v^i}\ . \tag{9}$$

Trusca et al. (2020) have shown that applying the weighting in each iteration of the rotatory mechanism separately on the target vectors leads to the best results. That is why, in our study, we use hierarchical attention in the same way. In total, we iterate three times over the rotatory mechanism.

After completing the iterations, one uses the final sentence representation vector v to predict the corresponding sentiment. This vector is then fed to a *softmax* function in order to compute the sentiment of the target:

$$\underset{|C|\times1}{p} = softmax(\underset{|C|\times8d}{W_c} \times \underset{8d\times1}{v} + \underset{|C|\times1}{b_c}), \tag{10}$$

where p is a conditional probability distribution, C the set of sentiment categories, W_c a weight matrix, and b_c a bias. The vector p gives a probability for every sentiment category in C. The final predicted sentiment is given by the highest probability of p.

3.3. Curriculum Learning

As mentioned in Section 2.2, using a curriculum strategy when training the model instead of presenting the samples in a random order, can significantly improve the accuracy and efficiency of a model. During the training of our model, we use both the *baby steps* and the *one-pass* curriculum, which are now discussed.

3.3.1. Baby Steps Curriculum

We use the formal definition given by Rao et al. (2020) to explain the *baby steps* algorithm in ABSA. For every sentence s_i in our dataset D, the sentiment is described as $y_i \in \{1, .., C\}$, where $i \in \{1, .., n\}$ for n sentences in D, and C is the number of sentiment categories. The order in which the sentences are fed to the model is determined by the curriculum strategy $S(s_i)$, where S defines how difficult a sample s_i is. The easiest sample gets the lowest S score and the most difficult sample the highest S score. We then use these S scores to order all the sentences in our dataset D from easy to difficult, which results in the ordered dataset D'.

With the *baby steps* algorithm, we then divide the ordered dataset D' into k buckets. We start by taking the bucket with the easiest instances, D^1, and train the model with these instances until convergence or until p epochs have passed. After this, the next bucket of data, containing more difficult samples, is added to the training data and we repeat the training process until convergence or until p epochs have passed. We keep adding more difficult batches to the training data until all the sentences of D are included and the model is trained on the complete training dataset. A visualization of the *baby steps* algorithm is given by the pseudocode in Algorithm 1.

Data: M, the model; D, the training data; S the curriculum score
begin
  $D' = \text{sort}(D,S)$
  $\{D^1, D^2, .., D^k\} = D'$, where $S(d_a) < S(d_b)$, $d_a \in D^i$, $d_b \in D^j$, $\forall i < j$
  $D^{train} = \emptyset$
  for s = *1,..,k* **do**
    $D^{train} = D^{train} \cup D^s$
    while *not converged for p epochs* **do**
      train(M, D^{train})
    end
  end
end

Algorithm 1: The *baby steps* algorithm

3.3.2. One-Pass Curriculum

Next to the *baby steps* algorithm, Cirik et al. (2016) also use the *one-pass* algorithm in their study. As mentioned in Section 2.2, the authors show that a model trained with the *baby steps* algorithm outperforms the *one-pass* algorithm. However, their conclusions are for sentiment analysis, while we

focus on ABSA. That is why, in this research, we use both the *baby steps* algorithm and the *one-pass* algorithm to implement our curriculum strategy.

The *one-pass* curriculum is similar to the *baby steps* curriculum, with the difference that we discard every bucket of data after training until convergence or until p epochs have passed. The final model is obtained after the training on the final bucket is completed. A visualization of the *one-pass* algorithm is given in Algorithm 2.

Data: M, the model; D, the training data; S the curriculum score
begin
  $D' = \text{sort}(D,S)$
  $\{D^1, D^2, .., D^k\} = D'$, where $S(d_a) < S(d_b)$, $d_a \in D^i$, $d_b \in D^i$, $\forall i < j$
  for s = *1,..,k* **do**
    while *not converged for p epochs* **do**
      train(M, D^s)
    end
  end
end

Algorithm 2: The *one-pass* algorithm

3.4. Curriculum Strategy

In our research, the curriculum strategy relies on the SentiWordNet sentiment lexicon proposed by Rao et al. (2020). The authors have shown that this strategy has good results for sentiment analysis of sentences. Given this fact, in our research, we investigate whether these improved results are also visible in case of ABSA. We follow the research of Rao et al. (2020), by using SentiWordNet features to order the instances and subsequently use the *baby steps* and *one-pass* curriculum strategy to train our model.

For every word in a sentence, SentiWordNet gives a positivity, a negativity, and an objectivity score (Baccianella et al., 2010). The objectivity score is computed as $1 -$ *positivity score - negativity score*. Important to note here, is that the SentiWordNet scores are for a synset and not for a word, which means that one word possibly has multiple scores. That is why in our research we make use of word sense disambiguation to determine which particular synset of the word is meant, given its context. We perform word sense disambiguation by implementing the Simplified Lesk algorithm (Lesk, 1986). This algorithm considers all the different definitions from every possible synset of a word and determines the number of overlapping words between the word's context sentence and the definitions. The synset with the most overlapping words is then returned as the most appropriate meaning of the word in the given sentence. We improve the performance of Simplified Lesk by including the part-of-speech tag of the word we disambiguate. The part-of-speech tag can be for example noun, verb, or adjective. We determine the part-of-speech tag using the NLTK platform (Bird et al., 2009).

After applying the Simplified Lesk algorithm, we end up with all the appropriate synsets for the given sentences. We then proceed with these disambiguated words and compute the associated SentiWordNet scores for all the words in our dataset. Then, for each sentence, we sum up all the positivity scores of the words in the sentence, leading to the feature Net Positivity Score, denoted by P. In a similar way we compute the features Net Negativity Score, denoted by N, and Net Objectivity Score, denoted by O. In addition to these scores, we also compute the Absolute Difference Score, denoted by AD. This feature is the absolute difference between the Net Positivity and the Net Negativity Score. Next to these features, we also include the total amount of words in the sentence, denoted by l. This feature is included since it is shown that shorter sequences are easier to train than longer ones (Spitkovsky et al., 2010).

Rao et al. (2020) use the above features to determine the difficulty of the sentences. However, their research is focused on sentiment analysis of sentences, while our study focuses on aspects rather than sentences. This means that there can be sentences that contain multiple aspects. When a sentence contains more than one aspect, we include multiple samples of the same sentence in our dataset, every sample having a different aspect marked as target. It is, however, imaginable that it is more difficult for the model to predict the sentiment of an aspect when there are multiple aspects in the same sentence, since the model then has to decide which words are connected to which aspect. That is why, next to the features used by Rao et al. (2020), we also include the feature A, representing the total number of different aspects in the sentence. Furthermore, we include the context words in the sentence, denoted by W.

Also, we include the aspect category of the marked aspect in the sentence as a feature as well. We do this by creating a one-hot encoding for every sample, denoted by C. This one-hot encoding is a vector of length twelve, representing the twelve different aspect categories in the dataset. All the indices of this vector contain a zero, except for the index of the marked aspect category, which contains a one. In this way, for every sentence, the aspect category of the marked aspect is included as a feature.

An overview of the eight discussed features is given in Table 1. The first five features are proposed by Rao et al. (2020) for sentiment analysis. The remaining three features are added by us to make the set of features more appropriate for ABSA. Looking at all these features, it is likely that a longer sentence has higher scores than a shorter sentence because of the larger total number of words. That is why, we include the scaled features as well. A feature is scaled by dividing the features with the length of their associated sentence. For example, the Scaled Positivity Score is then $\frac{P}{l}$. We do not include a scaled feature for the aspect category, since this is just a one-hot encoding and does not correlate with the sentence length. In this way, next to the eight features mentioned in Table 1, six more (scaled) features are added. In total, we use fourteen features for our curriculum strategy. As proposed by Rao et al. (2020), we normalize all the features between 0 and 1, to allow them to lie

Table 1: The features used for the curriculum strategy.

	Features	Description
Features proposed by Rao et al. (2020)	Sentence Length (l)	Number of words in the sentence
	Net Positivity Score (P)	Sum of all positivity scores
	Net Negativity Score (N)	Sum of all negativity scores
	Net Objectivity Score (O)	Sum of all objectivity scores
	Absolute Difference Score (AD)	Absolute difference between P and N
Added features	Aspects (A)	Number of aspects in the sentence
	Context Words (W)	Number of context words in the sentence
	Aspect Category (C)	The aspect category of the marked aspect

in the same range.

Rao et al. (2020) feed these features to an auxiliary feed-forward model Aux in order to learn which samples are the most difficult. The authors use the results of this model to determine the curriculum score of the samples of the dataset as follows:

$$S(s_i) = \sum_{j=1}^{C} (Aux(s_i)^j - y_i^j)^2, \tag{11}$$

where $Aux(s_i)^j$ is the prediction of the auxiliary model for the marked aspect in sample s_i for class j, with C the total amount of classes, and $y_i^j \in \{0, 1\}$ represents whether or not the target shows polarity j. If $S(s_i)$ is low, this means that it is easy to classify the target's sentiment. The computed scores $S(s_i)$ for every sample s_i in the training data are then used to determine the ordering of the samples before training our neural network of HAABSA++. We now discuss the auxiliary feed-forward model we use to calculate the curriculum scores.

3.4.1. Auxiliary Feed-Forward Model

The auxiliary feed-forward model is trained by minimizing a cross-entropy loss function, defined as:

$$\underset{1\times 1}{L} = -\sum_{j} \underset{|C|\times 1}{y_j} \times \log(\underset{|C|\times 1}{\hat{p_j}}), \tag{12}$$

where y_j is a vector containing the true sentiment of training instance j, $\hat{p_j}$ the predicted sentiment for that instance, and C the set of sentiment categories.

The weight matrices are initialized by the Glorot uniform initializer (Glorot & Bengio, 2010) and all biases are set to zero. Adam optimization is used to update the weights and biases (Kingma & Ba, 2015). Furthermore, during training, units are randomly dropped from the neural network (Srivastava et al., 2014). This is called the dropout technique and we apply this to all hidden layers to prevent units from co-adapting too much on the training instances.

Before training, the hyperparameters of the feed-forward model are tuned using the Hyperopt package (Bergstra et al., 2015). Hyperoptimization is used to determine the number of hidden layers

in the neural network and the number of neurons per hidden layer. Furthermore, the dropout rate and the learning rate are optimized as well. We use 80% of the training data for the hyperoptimization of the hyperparameters. The remaining 20% is used as a validation set. When the optimal values of the given hyperparameters are found, we use these parameters and train the model on the entire training set.

After applying hyperoptimization, we end up using a feed-forward neural network with two hidden layers and the following numbers of layer units: 183 and 140, respectively. The Rectified Linear Unit activation function (ReLU) is used as an activation function in our model. Inspired by the work of Rao et al. (2020), the final layer uses a softmax activation function. The accuracy of our feed-forward model is 74.0%.

Furthermore, Hyperopt is used not only for tuning of the hyperparameters of the auxiliary model, but it is also used when applying our curriculum strategies to the HAABSA++ model. In the neural network of HAABSA++ (LCR-Rot-hop++), multiple hyperparameters are optimized for the training data. However, when applying the *baby steps* or *one-pass* curriculum, we do not use the complete training data during every step. That is why we optimize the hyperparameters for every subset of the data. In this way, when during the training process a new bucket of data is selected, we simultaneously update the hyperparameters by using the optimal parameters for the currently used training data. The hyperparameter optimization is thus done for every possible combination of training data separately and we thus do not 'borrow' the optimal hyperparameters found for the HAABSA++ model, where no curriculum strategy is applied.

Last, it is important to note that in the *baby steps* and *one-pass* algorithms, we train the model with a subset of the data until convergence, before adding new data. Since we want to compare our strategy with the performance of the LCR-Rot-hop++ model of our baseline HAABSA++ method, we slightly alter the LCR-Rot-hop++ model to also take convergence into consideration. For this, before training the model, the training data is split into a training and a validation set, with, respectively, 80% and 20% of the original training data. We then train the model using this new training set and look at the performance of the model on the validation data to determine when the model has converged and, thus, when the training can be stopped. In case the model has not converged after p epochs, we also stop the training, as is done in our employed curriculum strategies as well.

4. Performance Evaluation

We now discuss the dataset that is used in our research and the different evaluation measures that we consider.

4.1. Datasets

For this research, restaurant reviews collected from the widely used datasets SemEval-2015 Task 12 Subtask 1 Slot 3 (Pontiki et al., 2015) and SemEval-2016 Task 5 Subtask 1 Slot 3 (Pontiki et al., 2016) are used. The SemEval-2015 dataset is a subset of the SemEval-2016 dataset, and therefore has the same properties as the SemEval-2016 data.

The SemEval-2015 and SemEval-2016 datasets includes restaurant reviews with one or multiple sentences, containing one or more opinions. An opinion represents the sentiment given to an aspect. The aspect has a target expression associated and an aspect category. In our research, we use the word aspect to denote the aspect category, unless otherwise specified.

The data is divided into a train and test set. We use our training data to train our neural network model. Before feeding the training data to our model, we use a curriculum strategy to order the train data. An overview of the sentiment classes in the SemEval-2015 and SemEval-2016 train and test datasets is given in Table 2, ordered on the frequency of the sentiment classes. As can be seen, the majority of the opinions is positive. Furthermore, it can be observed that the training and testing datasets are similar.

Table 2: The frequencies of the sentiments in the SemEval-2015 SemEval-2016 datasets.

		Positive		Negative		Neutral		Total	
		Freq.	%	Freq.	%	Freq.	%	Freq.	%
SemEval-2015	Train data	925	72.4	41	3.2	312	24.4	1278	100
	Test data	320	53.7	32	5.3	245	41.0	597	100
SemEval-2016	Train data	1318	70.2	489	26.0	72	3.8	1879	100
	Test data	483	74.3	135	20.8	32	4.9	650	100

The data is in XML format and is preprocessed in a similar way as done by Wallaart & Frasincar (2019). To begin with, all the opinions that are stated implicitly are removed from the dataset. This is done because the employed neural network assumes that the target is present. Implicitly stated opinions have a non-existent aspect and therefore the `target` in the sentence is defined as `NULL`. In total, we delete 24.8% of the SemEval-2016 dataset and 25.0% of the SemEval-2015 dataset for this reason. After deleting these samples, the remaining samples are processed using the NLTK platform (Bird et al., 2009). All the instances are tokenized utilizing the WordNet lexical database (Miller, 1995).

In our research, we use BERT deep contextual word embeddings, computed in the same way as in the HAABSA++ model (Trusca et al., 2020). The BERT word embeddings are obtained using the pre-trained BERT Base model (Devlin et al., 2019). The dimension of the word embedding vectors is 768.

4.2. Evaluations Measures

In this paper, the HAABSA++ model without curriculum learning is used as a baseline method. Since the first part of this model is the ontology reasoner and this is not altered by curriculum learning, we only look at the performance of the neural network of HAABSA++, which is the LCR-Rot-hop++ model. As explained in Section 3.4.1, we train this model with 80% of the training data until convergence on the validation dataset, or until p ($p = 100$) epochs have passed. The validation dataset consists of the remaining 20% of the training data and is held out during training. The results of this final model are then used as our baseline results.

In order to test the performance of implementing the *one-pass* and *baby steps* curriculum strategies, we divide our training data in k buckets, with k = 1,..,10. For the *one-pass* algorithm, we use the results of the final model that is trained with the last bucket of the training data and compare these with the results of the baseline model. For the *baby steps* algorithm, we look at the results of the final model that is trained with the complete training dataset.

Different evaluation metrics are used to compare our different strategies. First, we compute the accuracy of the model. The accuracy is computed by taking the sum of all correctly predicted sentiments, divided by the total number of samples. We look at both the in-sample accuracy and the out-of-sample accuracy. The first one is the accuracy for the complete training data and the latter one for the test data.

Furthermore, since Bengio et al. (2009) state that applying curriculum learning leads to faster convergence, we also compare the training times. This is the number of minutes the model is trained until the final model converges.

Finally, we look deeper into the progress of the out-of-sample and in-sample accuracy during the training of the model when a curriculum strategy is applied. For every k, we create a graph of the two accuracies and their course during the epochs. In this way, we can see how the accuracy changes when new buckets of data are added to the training data.

5. Results

In Table 3, the results for the SemEval-2016 and SemEval-2015 datasets are presented. The out-of-sample and in-sample accuracies are mentioned for every curriculum strategy together with the number of minutes required for the model convergence. The best results are marked in bold.

When comparing the results of the two curriculum strategies, we see that both strategies give very different results. In general, training the model with the *baby steps* curriculum leads to a better out-of-sample accuracy than when no curriculum or the *one-pass* curriculum is applied, as already suggested by Vijjini et al. (2021). For both the SemEval-2016 and the SemEval-2015 datasets, the optimal number of buckets that the training data should be divided into, is seven. Using the *baby steps* algorithm with seven buckets leads to an increase in the out-of-sample accuracy of 1.2-3.0

percentage points. For the other possible numbers of buckets, we see that the accuracy is at least as good as the accuracy of the baseline model. These results thus indicate that applying the *baby steps* curriculum strategy leads, in general, to a higher out-of-sample accuracy than when no curriculum is applied.

Since Bengio et al. (2009) state that applying a curriculum strategy does not only lead to better results, but can also lead to faster convergence, we compare the times needed to train the neural network with and without the *baby steps* curriculum strategy. Interestingly, we see that applying the curriculum approach during training leads to longer training times than the baseline model when the SemEval-2016 dataset is used. However, for the SemEval-2015 dataset, we see that if the data is divided into two, four, or five buckets, the training time is shorter than for the baseline model. In these cases, the model thus converges faster when the curriculum strategy is applied. Our results thus indicate that applying the *baby steps* curriculum leads to improved results, and also to faster convergence, in a few cases.

When we look at the training time of the *one-pass* curriculum, we see that this leads to faster convergence of the model than the baseline model and the *baby steps* curriculum. This is as expected, since with the one-pass curriculum the model is trained on a small set of samples, while the baseline model uses all the samples and in the *baby steps* curriculum the number of training samples gradually increases.

At the same time, we see a very low out-of-sample and in-sample accuracy for the *one-pass*

Table 3: Accuracy results and training times for the LCR-Rot-hop++ model with different curriculum strategies.

	SemEval-2016			SemEval-2015		
	Out-of-sample acc.	In-sample acc.	Time (min.)	Out-of-sample acc.	In-sample acc.	Time (min.)
No Curriculum Learning						
Baseline model	86.3%	96.3%	1.70	77.2%	96.7%	1.34
Baby Steps Curriculum						
Two buckets	87.1%	95.4%	3.77	79.9%	96.7%	1.06
Three buckets	87.1%	95.9%	3.00	77.9%	96.7%	2.08
Four buckets	86.9%	96.0%	3.23	77.4%	96.6%	1.08
Five buckets	86.3%	95.8%	4.62	77.2%	96.7%	1.18
Six buckets	86.3%	94.2%	4.44	78.4%	96.7%	1.66
Seven buckets	**87.5%**	94.6%	3.94	**80.2%**	93.3%	1.80
Eight buckets	86.3%	95.4%	5.22	80.1%	96.9%	3.52
Nine buckets	86.3%	94.3%	5.43	79.2%	97.5%	2.78
Ten buckets	86.3%	93.9%	5.00	76.9%	95.9%	3.29
One-Pass Curriculum						
Two buckets	83.2%	88.2%	1.19	76.4%	92.4%	0.67
Three buckets	71.7%	80.7%	0.81	65.8%	74.3%	0.43
Four buckets	29.8%	35.6%	1.03	35.0%	21.9%	0.28
Five buckets	44.3%	50.7%	0.99	35.5%	22.7%	0.34
Six buckets	38.6%	44.5%	0.82	34.8%	23.1%	0.52
Seven buckets	21.7%	29.4%	0.99	34.2%	23.6%	0.63
Eight buckets	38.4%	44.2%	**0.58**	39.5%	32.4%	0.51
Nine buckets	22.0%	28.6%	1.02	36.0%	26.1%	**0.33**
Ten buckets	18.0%	25.8%	0.77	36.5%	26.8%	0.39

curriculum, compared to the baseline model and the *baby steps* curriculum. It is quite surprising that the *one-pass* curriculum leads to such a big decrease in terms of accuracy. The lowest out-of-sample accuracy is obtained when the training data is divided into ten buckets for the SemEval-2016 dataset. For this curriculum strategy, the course of the out-of-sample and in-sample accuracy is given in Figure 1. The gray lines mark that the model is trained on a new bucket of data and that the previous bucket is discarded. It can be observed that both the out-of-sample and in-sample accuracy increase for the first eight buckets, but decrease for the ninth and tenth bucket. So, looking at this graph, it seems like the accuracy of the model decreases a lot when the model is trained on more difficult training data. For every k number of buckets that we divide our training data into, a similar pattern can be observed, where training on the last one or two buckets leads to a big decrease in the accuracy.

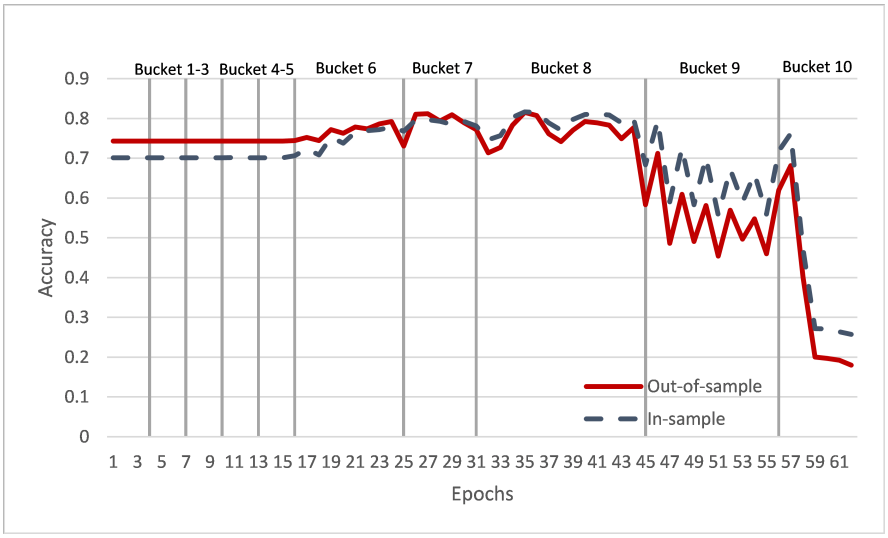

Figure 1: The out-of-sample and in-sample accuracy of the model trained with the *one-pass* curriculum with k=10 for the SemEval-2016 dataset.

The large decrease in the accuracy during the training of the model on the last few buckets of data thus leads to a final model that performs much worse than the baseline model. This remarkable result can be partly explained by looking at the composition of the ordered training data. This is the data ordered from easy to difficult based on their curriculum scores. It turns out that the majority of the easy samples has a positive sentiment, while the majority of the difficult samples has a negative sentiment. For example, when we divide the training data into ten buckets and look at the first bucket, which contains the easiest samples, we see that almost all the instances have a positive sentiment (99.6%), only one sample has a negative sentiment (0.4%), and there are no samples with a neutral sentiment. When the model is trained with this bucket of data, it always predicts a positive sentiment for the instances. Since the frequency of the positive sentiment in the test data is 74.3%, this results in a moderately high accuracy. However, in the tenth bucket, only

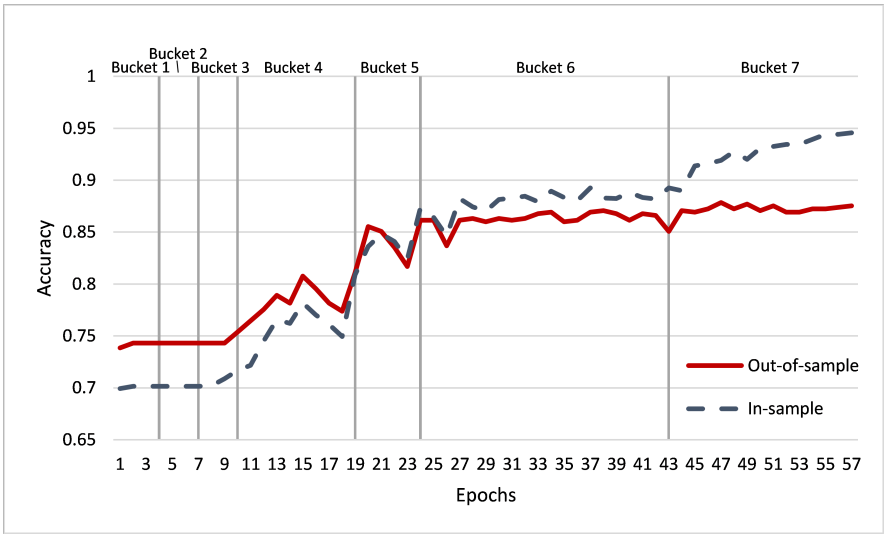

Figure 2: The out-of-sample and in-sample accuracy of the model with *baby steps* curriculum with k=7 for the SemEval-2016 dataset

1.6% of the samples have a positive sentiment, 36.4% has a neutral sentiment, and the majority of the samples has a negative sentiment (62.0%). When the model is trained with this data, it is thus more likely that the model predicts a negative or neutral sentiment, rather than a positive sentiment, given these frequencies. Therefore, it is not too surprising that the accuracy decreases when more difficult samples are used. However, the intuition behind the *one-pass* algorithm is that the model remembers what it has learned during the passed epochs and uses this knowledge during the training on the new samples. Unfortunately, for our case, the model does not seem to be good in remembering what it has learned during the training on the easier samples, which results in the low accuracies. So, in our study, the *one-pass* algorithm performs worse than the *baby steps* algorithm, which is also in line with the research of Cirik et al. (2016).

As mentioned in Section 4.2, we also take a look at the course of the out-of-sample and in-sample accuracy during the training of the model. From Table 3, we see that the best out-of-sample accuracy is obtained when we use the *baby steps* algorithm and divide the training data into seven buckets. The course of the corresponding out-of-sample and in-sample accuracy is given in Figure 2. The gray lines in this figure denote when a new bucket of data is added to the training data. We clearly see that every time that a new bucket of training data is added, the accuracy increases. The graphs also show that the model spends little time on training with the complete dataset and more time on training with intermediate buckets.

Again, a similar pattern can be observed for every k number of buckets that we divide our training data in. Remarkable is the fact that when the data is divided into a small number of buckets (k=2, 3, or 4), the largest part of the training is done with the complete training dataset, which can be seen in Figure 3. When we divide the data in a larger number of buckets, less time is spent on

training with the complete data, as can be observed from Figure 2. We thus see that incrementally adding more difficult samples to the training data leads to an increase of accuracy and also to less needed training time on the complete training dataset, which makes sense as the easy samples can be learned faster.

Finally, when looking at the course of the out-of-sample and in-sample accuracy of the *baby steps* algorithm, we see that during the training of the first buckets of data, the accuracy does not change. This could, again, be explained by the composition of the training data. The first buckets contain the easiest samples, which mainly have a positive sentiment, as is explained before. Therefore, during the first few buckets, the accuracy does not change much, as it is optimal for the model to predict a positive sentiment for the easy samples.

6. Conclusion

The aim of our research is to improve the results of a hybrid approach for ABSA by using curriculum learning based on *baby steps* and *one-pass* strategies. We use SentiWordNet features and an auxiliary model to obtain curriculum scores for the training samples. These scores are then used to order the samples and divide them into buckets for our curriculum strategies. Based on our results, only the *baby steps* curriculum strategy can improve the baseline with a margin between 1.2-3.0 percentage points. The poor results of the *one-pass* curricula might be explained by the unequal distribution of the classes over the buckets of data. Precisely, the easy samples are mainly positive and the difficult samples are mainly negative and neutral.

Given the importance of the topic of this research, we think that it would be useful if more research is done. Knowing that the data we use are restaurant reviews, which makes it likely

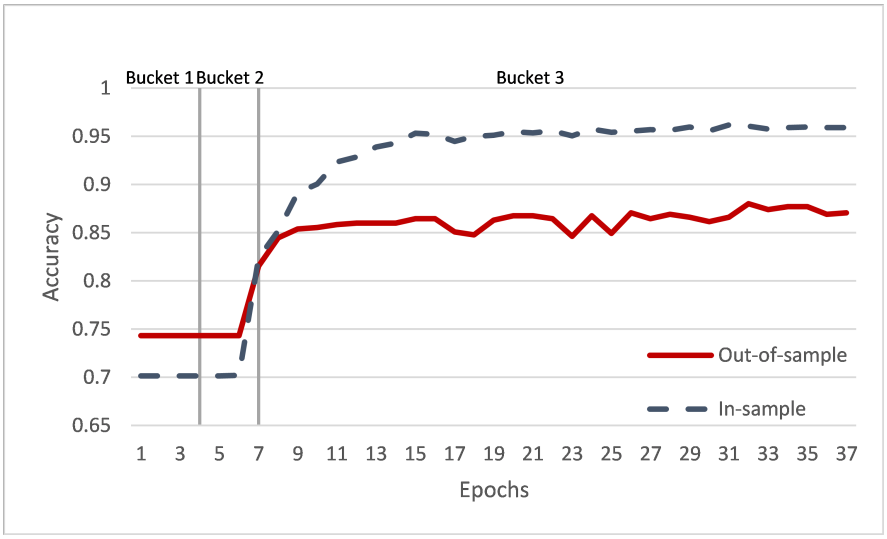

Figure 3: The out-of-sample and in-sample accuracy of the model with *baby steps* curriculum with k=3 for the SemEval-2016 dataset

that a big part of the data consists of short sentences, it would be interesting to see whether our inferences work for longer, and thus more difficult, sentences. Also, it would be interesting to see if the performance of this curriculum strategy could be improved if the sentiment classes are more equally distributed over the ordered training data. Finally, it might be interesting to investigate the effect of active learning as a closely related discipline to curriculum learning in the domain of ABSA. Unlike curriculum learning which imitates human learning, active learning is used to select new instances for training based on the information amount they bring (Settles, 2012). The most informative instances are usually picked based on the entropy or margin of prediction probabilities. Additionally, we can consider for future work the hybrid approach proposed by Jafarpour et al. (2021) as a linear combination between the active and curriculum learning strategies. Last, we would like to improve our curriculum learning strategy by making it adaptive to what the model has learned so far (Tu et al., 2024).

References

Baccianella, S., Esuli, A., & Sebastiani, F. (2010). SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In 7^{th} *International Conference on Language Resources and Evaluation (LREC 2010)*. ELRA. https://www.lrec-conf.org/proceedings/lrec2010/summaries/769.html.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In 26^{th} *Annual International Conference on Machine Learning (ICML 2009)* (pp. 41–48). ACM volume 382 of *ACM International Conference Proceeding Series*. https://doi.org/10.1145/1553374.1553380.

Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., & Cox, D. D. (2015). Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, *8*, 014008. https://doi.org/10.1088/1749-4699/8/1/014008.

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: Analyzing text with the natural language toolkit*. O'Reilly Media. https://www.amazon.com/Natural-Language-Processing-Python-Analyzing/dp/0596516495?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0596516495.

Cambria, E., Zhang, X., Mao, R., Chen, M., & Kwok, K. (2024). SenticnNt 8: Fusing emotion AI and commonsense AI for interpretable, trustworthy, and explainable affective computing. In 26^{th} *International Conference on Human-Computer Interaction (HCII 2024)* (pp. 197–216). Springer volume 15382 of *Lecture Notes in Computer Science*. https://doi.org/10.1007/978-3-031-76827-9_11.

Cheruku, R., Hussain, K., Kavati, I., Reddy, A. M., & Reddy, K. S. (2024). Sentiment classification with modified RoBERTa and recurrent neural networks. *Multimedia Tools and Applications.*, *83*, 29399–29417. https://doi.org/10.1007/s11042-023-16833-5.

Cirik, V., Hovy, E., & Morency, L.-P. (2016). Visualizing and understanding curriculum learning for long short-term memory networks. *arXiv preprint arXiv:1611.06204*.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). 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. https://doi.org/10.18653/v1/n19-1423.

Diwali, A., Saeedi, K., Dashtipour, K., Gogate, M., Cambria, E., & Hussain, A. (2024). Sentiment analysis meets explainable artificial intelligence: A survey on explainable sentiment analysis. *IEEE Transactions on Affective Computing*, *15*, 837–846. https://doi.org/10.1109/TAFFC.2023.3296373.

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, *48*, 71–99. https://doi.org/10.1016/0010-0277(93)90058-4

Esuli, A., & Sebastiani, F. (2006). SentiWordNet: A publicly available lexical resource for opinion mining. In 5^{th} *International Conference on Language Resources and Evaluation (LREC 2006)* (pp. 417–422). ELRA. https://www.lrec-conf.org/proceedings/lrec2006/pdf/384_pdf.pdf.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*, 119–139. https://doi.org/10.1006/jcss.1997.1504.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In 13^{th} *International Conference on Artificial Intelligence and Statistics (AISTATS 2010)* (pp. 249–256). JMLR volume 9 of *JMLR Proceedings*. https://proceedings.mlr.press/v9/glorot10a.html.

Han, H., Wang, S., Qiao, B., Dang, L., Zou, X., Xue, H., & Wang, Y. (2025). Aspect-based sentiment analysis through graph convolutional networks and joint task learning. *Information*, *16*, 201. https://doi.org/10.3390/info16030201.

Jafarpour, B., Sepehr, D., & Pogrebnyakov, N. (2021). Active curriculum learning. In 1^{st} *Workshop on Interactive Learning for Natural Language Processing (InterNLP 2021)* (pp. 40–45).

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio, & Y. LeCun (Eds.), 3^{rd} *International Conference on Learning Representations (ICLR 2015). arXiv preprint arXiv:1412.6980*.

Krueger, K. A., & Dayan, P. (2009). Flexible shaping: How learning in small steps helps. *Cognition*, *110*, 380–394.

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In 5^{th} *Annual International Conference on Systems Documentation (SIGDOC 1986)* (pp. 24–26). ACM. https://doi.org/10.1145/318723.318728.

Liu, B. (2015). *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Meskele, D., & Frasincar, F. (2020). Aldonar: A hybrid solution for sentence-level aspect-based sentiment analysis using a lexicalized domain ontology and a regularized neural attention model. *Information Processing & Management*, *57*, 102211. https://doi.org/10.1016/j.ipm.2020.102211.

Miller, G. A. (1995). WordNet: A lexical database for english. *Communications of the ACM*, *38*, 39–41. https://doi.acm.org/10.1145/219717.219748.

Nagatsuka, K., Broni-Bediako, C., & Atsumi, M. (2021). Pre-training a BERT with curriculum learning by increasing block-size of input text. In 12^{th} *International Conference on Recent Advances in Natural Language Processing (RANLP 2021)* (pp. 989–996). INCOMA https://aclanthology.org/2021.ranlp-1.112.

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., Al-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., Clercq, O. D., Hoste, V., Apidianaki, M., Tannier, X., Loukachevitch, N. V., Kotelnikov, E. V., Bel, N., Zafra, S. M. J., & Eryigit, G. (2016). SemEval-2016 task 5: Aspect based sentiment analysis. In 10^{th} *International Workshop on Semantic Evaluation (SemEval (2016))* (pp. 19–30). ACL. https://doi.org/10.18653/v1/s16-1002.

Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., & Androutsopoulos, I. (2015). SemEval-2015 task 12: Aspect based sentiment analysis. In 9^{th} *International Workshop on Semantic Evaluation (SemEval (2015))* (pp. 486–495). ACL. https://doi.org/10.18653/v1/s15-2082.

Rao, V. A., Anuranjana, K., & Mamidi, R. (2020). A SentiWordNet strategy for curriculum learning in sentiment analysis. In 25^{th} *International Conference on Applications of Natural Language to Information Systems (NLDB*

2020) (pp. 170–178). Springer volume 12089 of *Lecture Notes in Computer Science*. https://doi.org/10.1007/978-3-030-51310-8_16.

Schouten, K., & Frasincar, F. (2016). Survey on aspect-level sentiment analysis. *IEEE Tranactions on Knowledge and Data Engineering*, *28*, 813–830. https://doi.org/10.1109/TKDE.2015.2485209.

Schouten, K., & Frasincar, F. (2018). Ontology-driven sentiment analysis of product and service aspects. In 15^{th} *International Sematnic Web Conference (ESWC 2018)* (pp. 608–623). Springer volume 10843 of *Lecture Notes in Computer Science*. https://doi.org/10.1007/978-3-319-93417-4_39.

Settles, B. (2012). *Active learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. https://doi.org/10.2200/S00429ED1V01Y201207AIM018.

Skinner, B. F. (1958). Reinforcement today. *American Psychologist*, *13*, 94. https://doi.org/10.1037/h0049039

Sonawane, S. S., & Kolhe, S. R. (2022). Handling dimensionality of ambiguity using ensemble classification in social networks to detect multi-label sentiment polarity. *Computer Assisted Methods in Engineering and Science*, *30*, 7–26. https://doi.org/10.24423/cames.471

Spitkovsky, V. I., Alshawi, H., & Jurafsky, D. (2010). From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *2010 Conference of the North American Chapter of the Association of Computational Linguistics: Human Language Technologies (NAACL-HLT 2010)* (pp. 751–759). ACL. https://www.aclweb.org/anthology/N10-1116/.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

Tan, P., Steinbach, M. S., & Kumar, V. (2005). *Introduction to data mining*. Addison-Wesley. https://www-users.cs.umn.edu/%7Ekumar/dmbook/.

Trusca, M. M., Wassenberg, D., Frasincar, F., & Dekker, R. (2020). A hybrid approach for aspect-based sentiment analysis using deep contextual Word Embeddings and Hierarchical Attention. In 20^{th} *International Conference on Web Engineering (ICWE 2020)* (pp. 365–380). Springer volume 12128 of *Lecture Notes in Computer Science*. https://doi.org/10.1007/978-3-030-50578-3_25.

Tsvetkov, Y., Faruqui, M., Ling, W., MacWhinney, B., & Dyer, C. (2016). Learning the curriculum with bayesian optimization for task-specific word representation learning. In 54^{th} *Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. ACL. https://doi.org/10.18653/v1/p16-1013.

Tu, G., Niu, T., Xu, R., Liang, B., & Cambria, E. (2024). AdaCLF: An adaptive curriculum learning framework for emotional support conversation. *IEEE Intelligent Systems*, *39*, 5–11. https://doi.org/10.1109/MIS.2024.3411369.

Vijjini, A. R., Anuranjana, K., & Mamidi, R. (2021). Analyzing curriculum learning for sentiment analysis along task difficulty, pacing and visualization axes. In 11^{th} *Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA@EACL 2021)* (pp. 117–128). ACL. https://aclanthology.org/2021.wassa-1.13/.

Wallaart, O., & Frasincar, F. (2019). A hybrid approach for aspect-based sentiment analysis using a lexicalized domain ontology and attentional neural models. In 16^{th} *International Semantic Web Conference (ESWC 2019)* (pp. 363–378). Springer volume 11503 of *Lecture Notes in Computer Science*. https://doi.org/10.1007/978-3-030-21348-0_24.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In 7^{th} *International Conference on Learning Representations (ICLR 2019)*. OpenReview.net. https://openreview.net/forum?id=rJ4km2R5t7.

Wang, Z., Ho, S., & Cambria, E. (2020). Multi-level fine-scaled sentiment sensing with ambivalence handling. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, *28*, 683–697. https://doi.org/10.1142/S0218488520500294.

Weichselbraun, A., Gindl, S., Fischer, F., Vakulenko, S., & Scharl, A. (2017). Aspect-based extraction and analysis of

affective knowledge from social media streams. *IEEE Intelligent Systems*, *32*, 80–88. https://doi.org/10.1109/MIS.2017.57.

Xu, B., Zhang, L., Mao, Z., Wang, Q., Xie, H., & Zhang, Y. (2020). Curriculum learning for natural language understanding. In *58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)* (pp. 6095–6104). ACL. https://doi.org/10.18653/v1/2020.acl-main.542

Xu, L., Xie, H., Qin, S. J., Wang, F. L., & Tao, X. (2025). Exploring ChatGPT-based augmentation strategies for contrastive aspect-based sentiment analysis. *IEEE Intelligent Systems*, *40*, 69–76. https://doi.org/10.1109/MIS.2024.3508432.

Yadav, R. K., Jiao, L., Granmo, O., & Goodwin, M. (2021). Human-level interpretable learning for aspect-based sentiment analysis. In *35th AAAI Conference on Artificial Intelligence (AAAI 2021)* (pp. 14203–14212). AAAI Press. https://ojs.aaai.org/index.php/AAAI/article/view/17671.

Zhang, X., Mao, R., & Cambria, E. (2024). SenticVec: Toward robust and human-centric neurosymbolic sentiment analysis. In L. Ku, A. Martins, & V. Srikumar (Eds.), *62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)* (pp. 4851–4863). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.findings-acl.289.

Zheng, S., & Xia, R. (2018). Left-center-right separated neural network for aspect-based sentiment analysis with rotatory attention. *arXiv preprint arXiv:1802.00892*.

Zhu, L., Mao, R., Cambria, E., & Jansen, B. J. (2024). Neurosymbolic AI for personalized sentiment analysis. In *26th International Conference on Human-Computer Interaction (HCII 2024)* (pp. 269–290). Springer volume 15382 of *Lecture Notes in Computer Science*. https://doi.org/10.1007/978-3-031-76827-9_16.

Zhu, Q., Chen, X., Wu, P., Liu, J., & Zhao, D. (2021). Combining curriculum learning and knowledge distillation for dialogue generation. In *Findings of the Association for Computational Linguistics (EMNLP 2021)* (pp. 1284–1295). ACL. https://doi.org/10.18653/v1/2021.findings-emnlp.111.