Data Augmentation in a Hybrid Approach for Aspect-Based Sentiment Analysis

Tomas Liesting
Erasmus University Rotterdam
Rotterdam, the Netherlands
tomas.liesting@gmail.com

Flavius Frasincar
Erasmus University Rotterdam
Rotterdam, the Netherlands
frasincar@ese.eur.nl

Maria Mihaela Trușcă
Bucharest Univ. of Economic Studies
Bucharest, Romania
maria.trusca@csie.ase.ro

ABSTRACT

Data augmentation is a way to increase the diversity of available data by applying constrained transformations on the original data. This strategy has been widely used in image classification but has to the best of our knowledge not yet been used in aspect-based sentiment analysis (ABSA). ABSA is a text analysis technique that determines aspects and their associated sentiment in opinionated text. In this paper, we investigate the effect of data augmentation on a state-of-the-art hybrid approach for aspect-based sentiment analysis (HAABSA). We apply modified versions of easy data augmentation (EDA), backtranslation, and word mixup. We evaluate the proposed techniques on the SemEval 2015 and SemEval 2016 datasets. The best result is obtained with the adjusted version of EDA, which yields a 0.5 percentage point improvement on the SemEval 2016 dataset and 1 percentage point increase on the SemEval 2015 dataset compared to the original HAABSA model.

CCS CONCEPTS

• **Information systems → Sentiment analysis**; *Information extraction*; Web mining;

KEYWORDS

aspect-based sentiment analysis, data augmentation

ACM Reference Format:
Tomas Liesting, Flavius Frasincar, and Maria Mihaela Trușcă. 2021. Data Augmentation in a Hybrid Approach for Aspect-Based Sentiment Analysis. In *The 36th ACM/SIGAPP Symposium on Applied Computing (SAC '21), March 22–26, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3412841.3441958

1 INTRODUCTION

Nowadays, there are many online platforms to compare and review products, locations, and services. Research has demonstrated that 91% of the people regularly read these online reviews, and around 84% trust them [2], implying that these opinionated texts influence the decisions of the majority of the consumers. Therefore, it is not surprising that the analysis of these opinionated texts awoke much interest in the past years. As the available information is enormous (e.g., Tripadvisor by itself already has over 435 million reviews [18]), the manual analysis is nearly impossible, meaning that the interest in automatic sentiment analysis has increased a lot as well.

In this research, the focus lies on aspect-based sentiment analysis (ABSA), where the goal is to predict the sentiment of a certain aspect [13]. ABSA has three crucial components. First, the target words that the sentiment is about have to be extracted (target extraction). Second, the aspects have to be detected depending on the discussed product or service (aspect detection). The third step is sentiment classification, in which the polarity of the opinion is determined (e.g., positive, negative, or neutral).

Currently, almost all state-of-the-art (SOTA) methods in ABSA rely on supervised learning due to its high rates of effectiveness. However, the major downside of supervised training is the unavailability of large annotated datasets. To mitigate this problem many researchers opt for transfer learning from other similar knowledge domains. As this approach might be too costly because it requires the access to external resources, another solution to extend the current training dataset is by using data augmentation, mainly employed in analyzing images [9]. The idea of data augmentation is increasing the available information by doing semantically constrained transformations on the training data. The idea is intuitive in image classification, as using the same image but shifted, zoomed, rotated, cropped, and many other transformations lead to more information improving performance [9], but can also be applied in natural language processing (NLP) [21].

In this paper, we explore how data augmentation techniques can improve the sentiment classification task. For this, we are not only interested in extending the training set but also in keeping a good quality of it. The baseline model we choose to evaluate our data augmentation techniques achieves high accuracy in sentiment classification and is proposed in [20] as a hybrid model with two steps. First, an ontology-based reasoner attempts to determine the polarity of a given aspect in text. If the ontology is not conclusive, the task is solved by a neural network (NN). The backup model is based on the LCR-Rot neural network [24]. The model splits the sentences into three parts, left context, target, and right context, used as input for three LSTMS cells and a rotatory attention. The model furthermore uses word embeddings from GloVe [8] and obtains SOTA results for several ABSA sentiment evaluation tasks. The source code of our work can be found on GitHub at https://github.com/tomasLiesting/HAABSADA.

The contribution of our paper can be summarised as follows:

(1) We introduce an NLP data augmentation framework together with a set of variations suitable for the sentiment classification task.

SAC '21, March 22–26, 2021, Virtual Event, Republic of Korea
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8104-8/21/03...$15.00
https://doi.org/10.1145/3412841.3441958

(2) Taking as example two widely used ABSA datasets, we prove that our techniques can boost the accuracy of the baseline model up to 2 percentage points.

The paper is organized as follows. In Section 2, relevant data augmentation techniques for NLP used by other researchers are presented. In Section 3 the datasets used for training and evaluation are introduced. Afterward, in Section 4, we explain how the techniques discussed in Section 2 can be used for ABSA, and we propose several extensions of these. In Section 5, the effect of these techniques on the accuracy of the model is given and discussed. In Section 6, we present the implications of our research and propose ideas for future research.

2 RELATED WORK

Data augmentation increases the number of data points in a training set by transforming data in a constrained manner. This technique has been prevalent in image classification, where its effectiveness has already been proven (e.g., by Perez and Wang [9]). In [9], the authors evaluate several techniques where a given image is rotated, tilted, cropped, or shifted, improving predictions. In NLP, some attention has been given to data augmentation as well, which is further discussed in this section.

First of all, Wei and Zou [21] create a generalized way to augment textual data. They propose easy data augmentation (EDA), which consists of four different data augmentation methods in NLP. First, they use Synonym Replacement (SR), where n words, which are not stopwords, are selected and replaced with synonyms. Secondly, they use Random Insertion, where they insert a random synonym of a word at a random position in a sentence. Their third method is Random Swap, where two words in a sentence are randomly swapped, and lastly, they propose Random Deletion, where random words within the sentence are deleted. They find that using 50% of the original data, they generally obtain similar results to the models not using data augmentation and that they achieve better results in all the tasks on which the data augmentation task has been performed using the full dataset. They do note that the data augmentation effect is more substantial on smaller datasets in comparison to larger datasets.

Another way to use data augmentation is introduced by Sennrich et al. [15], employed initially to improve machine translation. They translate the data into another language and afterward translate it back to obtain synthetic data. This technique is called backtranslation. They show that this method obtains better results in translation tasks. Though no research has been done on the use of backtranslation in ABSA, the method shows potential for NLP in general. Yu et al. [22] used backtranslation from English to French or German and back in order to enhance their dataset. This backtranslation initially resulted in twice as much data (every sentence is translated to French and back to English and is treated as a new sentence), which yielded an improvement on F1-scores of 0.5 percentage points on the SQuAD dataset [12]. When also using German backtranslation, obtaining three times as much data, this method resulted in another increase of 0.2 percentage points on top of the 0.5 percentage points. The authors also notice that augmenting data more than three times results in decreased performance, presumably because this backtranslated data is noisy compared to the original data. They modified the ratio between original and augmented data, and empirically found that a 3:1:1 ratio (three times the original data, one time the data backtranslated from French and one time the data backtranslated from German) resulted in the highest performance gain of 1.1-1.5 percent. This means that they used five times as much data for their best result.

Shleifer [16] used a combination of the previous methods on a pretrained multilayered AWD-LSTM model [7] on an IMDB movie dataset. In this analysis, the authors found that using EDA on the full dataset does not lead to substantial improvements on test classification, and that backtranslation has a small gain. Using a subset of the dataset does lead to improved performances with backtranslation. Wei and Zou [21] suspect that other models that use word embeddings like BERT will not benefit either from the use of EDA, as the word embeddings are contextual.

The last method used for data augmentation is the mixup, originally introduced by Zhang et al. [23] for image recognition. Their idea is to (linearly) interpolate between feature vectors of an image, which should be identical to the interpolation of the associated target vectors. The authors take two images and their corresponding target, $(x_i; y_i)$ and $(x_j; y_j)$, where x and y are the image and the target, respectively, and a value λ is drawn from the distribution $Beta(\alpha, \beta)$ distribution, where the authors set $\alpha = \beta$ as this yields a symmetric distribution, and $\alpha \in [0.1, 0.4]$, as higher values resulted in underfitting. They create a synthetic image $(\tilde{x}_{ij}; \tilde{y}_{ij})$ as given below:

$$\begin{aligned} \tilde{x}_{ij} &= \lambda x_i + (1 - \lambda) x_j \\ \tilde{y}_{ij} &= \lambda y_i + (1 - \lambda) y_j \end{aligned} \tag{1}$$

As word embeddings are essentially feature vectors, similar to images, the mixup can also be applied to NLP tasks [4]. In [4], the proposed adaptation for NLP is twofold. First, individual word embeddings can be interpolated. This is done by zero-padding the sentences to make them of the same length, after which interpolation is done on every word in the sentence. This means that the first word of sentence A is interpolated with the first word of sentence B, and so forth. Second, the model uses sentence embeddings. Here the sentences are fed to the model and are encoded by an LSTM or CNN into sentence embeddings. These representations are extracted afterward and a linear interpolation is performed on two representations. These two methods have shown to improve the accuracy of sentence classification with both CNN and LSTM significantly.

To summarize, the usage of EDA as introduced by Wei and Zou [21] is an interesting development, though the effect of this method appears to be small when using (contextual) word embeddings. Backtranslation has proven to provide small to substantial improvements on large and small datasets, respectively. Word and sentence mixups appear to be interesting developments in both large and small datasets. In the field of ABSA, data augmentation has not yet been used to the best of our knowledge.

3 DATA

The used data is given by the SemEval (or Semantic Evaluation) datasets of 2015 and 2016. SemEval is a series of evaluation workshops that aims to extract meanings out of sentences. The datasets are annotated by human linguists, and NLP algorithms aim to be

Table 1: Frequencies of polarities in the SemEval 2015 and 2016 train and test datasets.

Dataset	Positive		Neutral		Negative		Total	
	Frequency	%	Frequency	%	Frequency	%	Frequency	%
SemEval 2015 train	963	75	36	3	280	22	1279	100
SemEval 2015 test	353	59	37	6	207	35	597	100
SemEval 2016 train	1319	70	72	4	488	26	1879	100
SemEval 2016 test	483	74	32	5	135	21	650	100

as close to the human annotator as possible. SemEval datasets have different tasks. In our case, we use task 12 subtask 1 of SemEval 2015 [10] and task 5 subtask 2 of SemEval 2016 [11]. The goal of these tasks is to predict the polarity of a sentence about a given aspect. This way, the performance benefits when applying the data augmentation can be measured.

The previously considered two datasets contain restaurant reviews with one to several sentences represented in the XML format. In each sentence, an aspect, a category, and a polarity are given. An example of a sentence in the dataset from SemEval 2016 can be seen in Figure 1. In this figure the aspect is given in the *target* field, a general category is given in the *category* field, and the polarity of the sentiment with respect to the previous category is given in the *polarity* field. The dataset of SemEval 2015 has a similar structure. The preprocessing of the data and the computation of word embeddings are done similarly to the work of Wallaart and Frasincar [20].

```
<sentence id="1004293:0">
  <text>Judging from previous posts this used to be a good place , but not any
      longer .</text>
  <Opinions>
          <Opinion target="place" category="RESTAURANT#GENERAL" polarity="
              negative" from="51" to="56"/>
        </Opinions>
</sentence>
```

Figure 1: Annotated data fragment

Table 1 shows the frequencies of polarities for the training and testing data of the employed datasets. In all training sets, neutral sentences are the least frequent, and positive are the most frequent. Table 2 gives an overview of the categories in the SemEval datasets. Food quality, general service, and general ambiance are the most represented categories, making up well over 50 percent of the data. The relative frequency of the categories in the test and train data appears similar for both datasets, though the ratio is more similar in 2016 than in 2015. This is also the case for the polarities.

4 FRAMEWORK

This section gives an overview of the framework used in this paper. Section 4.1 gives a short presentation of the ontology reasoner used in HAABSA. Afterward, Section 4.2 gives a description of the used machine learning method and the used word embeddings. Last, Section 4.3 explains existing data augmentation techniques and proposed extensions appropriate for ABSA.

4.1 Ontology Reasoner

HAABSA is a hybrid model for ABSA that consists of two stages. First of all, an ontology reasoner is employed, which is similar to the reasoner used by Schouten and Frasincar [14]. This reasoner uses a domain-specific sentiment ontology to determine the polarity of a sentence about an aspect. The domain sentiment ontology groups concepts in three classes: *SentimentValue*, *Aspect Mention*, and *SentimentMention*. The first contains two classes, namely *Positive* and *Negative*. The *AspectMention* provides lexical representation for aspect categories. An example of this is the word *fish* which is linked to the category *FOOD#QUALITY*.

Table 2: Categories in the SemEval 2015 and 2016 train and test datasets.

Category	2015		2016	
	train	test	train	test
FOOD#QUALITY	524	242	765	283
SERVICE#GENERAL	217	104	324	107
AMBIANCE#GENERAL	164	68	228	59
RESTAURANT#GENERAL	124	59	183	58
FOOD#STYLE_OPTIONS	81	33	116	51
FOOD#PRICES	41	26	71	22
DRINKS#QUALITY	32	11	44	22
RESTAURANT#MISCELLANEOUS	30	19	49	18
DRINKS#STYLE_OPTIONS	26	6	32	11
LOCATION#GENERAL	14	8	22	10
RESTAURANT#PRICES	10	16	26	5
DRINKS#PRICES	15	5	20	4
FOOD#GENERAL	1	0	0	0

The *SentimentMention* class determines whether the sentiment of a sentiment expression is positive or negative for a specific aspect. There are three types of *SentimentMentions*. Type 1 concerns words that have the same sentiment for every aspect. An example is the word *bad*, which is always negative, no matter the context. Type 2 are words that always have the same polarity but only apply to some aspects. An example is the word *delicious*, which can be applied to food and drinks, but not, for example, to a couch. Type 3 is the last one and represents words of which their polarity is dependent on the context. Take, for example, the word *cold*. If ice-cream is cold, it has a positive or neutral polarity, while cold fries generally have a negative polarity.

4.2 Multi-Hop Left-Center-Right Neural Network with Rotatory Attention

The second step of the hybrid model is the machine learning method. As the ontology reasoner can only predict sentiment in 60% of the cases, an ML technique is used to classify the remaining data [14]. The best model presented by Wallaart and Frasincar [20], the multi-hop LCR-Rot, splits up a sentence into left context, target, and right context vectors. These vectors are the input for a left, center, and right bidirectional LSTM cell, respectively. The outputs of these cells are the inputs for a rotatory attention mechanism, consisting of two steps. The first step determines the most indicative words in the left and right contexts, and the second step captures the most important words for the target. The rotatory attention is applied for several iterations, where three iterations yielded the best results.

For the ML technique, the word embeddings are created using the GloVe (or global vectors) embeddings [8]. The advantage of GloVe is that, unlike the skip-gram or CBOW methods, GloVe uses both local information and a global word co-occurrence matrix to create word embeddings. This method resulted in the best performance of the hybrid model compared to the CBOW and skip-gram methods. The word embeddings of GloVe are freely available for download[1].

4.3 Data Augmentation

Data augmentation in ABSA seems to be an untouched subject. For NLP, in general, the literature suggests three different kinds of data augmentation, namely easy data augmentation (EDA), back-translation, and mixup. The following subsections discuss the adaptations of the methods appropriate for ABSA and suggest several extensions.

4.3.1 Easy Data Augmentation for ABSA. The idea of easy data augmentation (EDA) proposed by Wei and Zou [21] is to provide an easy and effective way to augment data in NLP. This is done by randomly replacing synonyms, randomly inserting or deleting words, and randomly swapping words. For the case of ABSA, there are some specifics to be considered. First of all, the used model splits the sentence up in left context, target, and right context vectors to determine the polarity, instead of using the sentence as a whole as input. Secondly, the target expression (a word or a group of words) should remain present and unchanged. Removing the target words or splitting the expression up would make a classification of the polarity impossible, as the machine learning algorithm cannot handle this. Keeping these considerations in mind, we propose several adaptations to make EDA compatible with ABSA. Then, we introduce variations and extensions to these methods.

Random Insertion. Random insertion is the process of selecting a random word in the sentence, finding a synonym in WordNet [3] using the natural language toolkit (NLTK) [1], and inserting this synonym at some random place in the sentence. This is possible without many modifications, as this augmentation can be done before splitting the sentence into left, target, and right context. The only thing to keep in mind is that the insertion should not be within the target expression. This is ensured through replacing the target expression with a fixed expression (namely $\$t\$$) before the insertion. Afterward, we replace the fixed expression with the target expression again and append the sentence to the training data.

Random Deletion. This method selects a random word from a sentence and removes it. For this method, we make sure that the target words cannot be deleted. Therefore, the random deletion procedure is done on the left context vector and the right context vector, not on the target expression. However, we suspect that this process would not work very well in ABSA, as the sentiment of a sentence can easily change when removing a word. If the target, for example, is the food, and the word *delicious* is removed, determining the polarity would be much more difficult for a model.

Random Swap. This method takes two words in the sentence and swaps these words. Again we must make sure that the target words remain unchanged. Replacing the target with a single expression (again $\$t\$$) resolves this issue, as this ensures that the full target expression is swapped instead of only part of the target.

Synonym Replacement. This method takes a random word in a sentence, looks up a synonym in WordNet, and replaces the word with its synonym. We apply a similar procedure as before and replace the target with a fixed expression ($\$t\$$) to make sure that this expression cannot be selected for the synonym replacement.

These four methods combined create the original EDA for ABSA. All four procedures are similar to a certain extent, so only the synonym replacement is given in Algorithm 1.

Algorithm 1: Synonym replacement in original EDA

```
Data: $sentences$, sentences of the SemEval dataset; $tar$, target
      expression in the sentece; $\alpha$, percentage of words replaced
Result: $augmentedSentences$
begin
    Set $augmentedSentences = \phi$
    foreach $originalSentence \in sentences$ do
        $sentence \longleftarrow replace(tar, \$t\$, originalSentence)$
        $sentence \longleftarrow split(sentence)$
        $numberAugmentations \longleftarrow \alpha * length(sentence)$
        while $i < numberAugmentations$ do
            $word \longleftarrow$
              $takeRandomWordNotTarget(sentence)$
            $syn \longleftarrow$
              $takeRandomWordNetSynonym(word)$
            $sentence \longleftarrow$replace $word$ by $syn$
        end
        $augmentedSentences \longleftarrow$
          $augmentedSentences \cup sentence$
    end
    return $augmentedSentences$
end
```

4.3.2 Extensions on EDA. EDA aims to be a quick and straightforward method that can be used on all sorts of NLP tasks. In our case, we look for data augmentation techniques specifically useful for ABSA, and as our training datasets consists of about 1000-2000 sentences, speed is not a big issue for this problem. Therefore, we propose several extensions for EDA.

Word Sense Disambiguation. First of all, synonym replacement and random insertion in the case of EDA pick a random synonym from the WordNet sentiment lexicon. This raises some problems. First of all, when looking up synonyms sometimes words with a different function in a sentence are returned. Take the example in Figure 1, repeated below in Example 1:

Example 1. *Judging from previous posts this used to be a good place, but not any longer.*

When looking up synonyms for the word *judging*, WordNet returns both nouns (like *judgment*) and verbs (like *evaluate*), meaning that the function of a word is context-dependent. Additionally,

[1]http://nlp.stanford.edu/data/glove.42B.300d.zip

words with similar functions, or parts-of-speech (POS), can also have different meanings. Take the word *post* in the same sentence. Without context, it can be a military post, mail, or a social media post. This means that many synonyms that are replaced in the original sentences are not actually synonyms, polluting the sentences rather than enhancing them. The process of selecting the correct meaning of the word in a sentence is called word sense disambiguation (WSD) [19].

In order to properly use WSD, we propose a twofold method. First, the POS of every word is determined. This is a complex problem in itself that already got much attention. We use the standard NLTK POS tagger, which is a greedy averaged perceptron tagger [5].

Afterward, the true sense of the word is determined. An easy algorithm for WSD is proposed by Lesk [6]. In the Lesk algorithm, all possible definitions of the words in a sentence are looked up and, based on the overlap between two sentences, the proper definition of a word is determined. This method has been refined by the simplified version of Lesk [19]. Here, a word is looked up in a dictionary, and the definition in the dictionary is used, which gives the highest overlap with the context words. Alternatively, let w be a word for which we want to know its meaning, with its corresponding context words $C = [c_1, c_2, ..., w, ..., c_m]$, which contains $m + 1$ words. Let $S = [s_1, s_2, ..., s_n]$ be the possible meanings of the word, with corresponding definitions $[D_1, D_2, ..., D_n]$, where D_i is the set of words in the definitions. We take the meaning with the maximum overlap, where the overlap is calculated as in Equation 2.

$$Overlap(w, s_i) = |D_i \cap C| \quad (2)$$

The previously discussed method, called simplified Lesk, has been implemented in a WordNet [3] library. WordNet is an extensive lexical database of English words, where nouns, verbs, adjectives, and adverbs are grouped in sets of synonyms called synsets. It attempts to capture all possible meanings of a word with their corresponding definitions, making it a useful tool for WSD. For the simplified Lesk algorithm, the Pywsd library is used [17].

Simplified Lesk is used for both synonym replacement and random insertion, adjusting the original implementation of EDA. The pseudocode of synonym replacement in the adjusted manner is given in Algorithm 2.

Target swap across sentences. In the described tasks, more information is available than only the sentence. In the training data (as can be seen in Figure 1), a category is given together with its target expression and its position. We can therefore replace random swaps by swapping target words of the same categories. This way, similar targets have multiple contexts in which they can appear. The amount of possible swaps is variable, meaning that it is possible to create many augmented records. We will experiment with one swap per sentence, such that this method can be compared to other data augmentation methods, as this results in the same amount of augmentations. The pseudocode of this method is given in Algorithm 3.

Algorithm 2: Synonym replacement in adjusted EDA

```
Data: $sentences$, sentences of the SemEval dataset; $tar$, target
    expression in the sentece; $\alpha$, percentage of words replaced
Result: $augmentedSentences$
begin
    Set $augmentedSentences = \phi$
    foreach $originalSentence$ in $sentences$ do
        $partsOfSpeech \longleftarrow$
          $POSTagger(originalSentence)$
        $sentence \longleftarrow replace(tar, \$t\$, originalSentence)$
        $sentence \longleftarrow split(sentence)$
        $numberAugmentations \longleftarrow \alpha * length(sentence)$
        while $i < numberAugmentations$ do
            $word \longleftarrow$
              $takeRandomWordNotTarget(sentence)$
            $maxOverlap \longleftarrow 0$
            $bestSense \longleftarrow$most frequent sense of $word$
            foreach $sense$ in senses of $word$ with same
              part-of-speech in $partsOfSpeech$ do
                $signature \longleftarrow$set of words in the gloss and
                  examples
                $overlap \longleftarrow computeWordnetOverlap($
                $signature, originalSentence)$
                if $overlap > maxOverlap$ then
                    $maxOverlap \longleftarrow overlap$
                    $bestSense \longleftarrow sense$
                end
            end
            $sentence \longleftarrow$replace $word$ with random synonym
              from the synset corresponding to $bestSense$ in
              $originalSentence$
        end
        $augmentedSentences \longleftarrow$
          $augmentedSentences \cup sentence$
    end
    return $augmentedSentences$
end
```

4.3.3 Backtranslation. For backtranslation we face a similar problem as with EDA. When translating a sentence to another language and back, the target expression could have changed, resulting in not knowing at which position in the sentence the target is. To tackle this problem, the left context and the right context are independently translated into different languages, while the target remains the same. We pick Dutch and Spanish with Latin alphabets, and we pick Japanese to investigate the effect of translation to a non-Latin alphabet. In order to translate the data and back the Google Translate API is used[2]. Algorithm 4 describes the way this backtranslation is done.

4.3.4 Mixup. Mixup is a linear interpolation between feature vectors, identical to the interpolation of the associated class vectors [23]. The technique of mixup is used initially for image classification tasks, but recently had been applied for NLP. For this paper, we introduce a variant of mixup appropriate for the considered two-step hybrid model for ABSA.

In order to properly apply mixup at a sentence level, Guo et al. [4] introduced word-mixup. This way the input sentences are zero-padded on the right in order to make them of the same length, after which the linear interpolation is done. As our model makes use of a left context, target, and right context, we propose the following

[2]https://cloud.google.com/translate/docs/basic/translating-text

Algorithm 3: Target swap in adjusted EDA

```
Data: $sentences$, sentences of the SemEval dataset; $asp$, aspect of
      a given sentence; $categories$, the categories in the SemEval
      datasets
Result: $augmentedSentences$
begin
    Set $augmentedSentences = \phi$
    foreach $category \in categories$ do
        $sentences \longleftarrow$ sentences in $category$
        for $sentence_i, sentence_j \in sentences$ do
            $sentence_i \longleftarrow replace(asp_i, asp_j, sentence_i)$
            $sentence_j \longleftarrow replace(asp_j, asp_i, sentence_j)$
            $augmentedSentences \longleftarrow$
              $augmentedSentences \cup sentence_i \cup sentence_j$
            $sentences \longleftarrow$
              $sentences - \{sentence_i, sentence_j\}$
        end
    end
    return $augmentedSentences$
end
```

Algorithm 4: Backtranslation

```
Data: $sentences$, sentences of the SemEval dataset; $tar$, target
      expression in a sentence; $lang$, target language to translate
      to
Result: $augmentedSentences$
begin
    Set $augmentedSentences = \phi$
    foreach $originalSentence \in sentences$ do
        $left, target, right \longleftarrow$
          $split(originalSentence, tar)$
        $translatedLeft \longleftarrow translate(left, lang)$
        $translatedRight \longleftarrow translate(right, lang)$
        $backtranslatedLeft \longleftarrow$
          $translate(translatedLeft, Eglish)$
        $backtranslatedRight \longleftarrow$
          $translate(translatedRight, English)$
        $backtranslated \longleftarrow backtranslatedLeft +$
          $target + backtranslatedRight$
        $augmentedSentences \longleftarrow$
          $augmentedSentences \cup backtranslated$
    end
    return $augmentedSentences$
end
```

method. We define $S_i = [w_1, w_2, ..., w_{N_i}]$ as a sentence i which contains N_i words, and $S_j = [w_1, w_2, ..., w_{N_j}]$ as sentence j with N_j words. We first split the sentence up in $S_{i,l} = [l_1, l_2, ..., l_{L_i}]$, $S_{i,c} = [c_1, c_2, ..., c_{C_i}]$, and $S_{i,r} = [r_1, r_2, ..., s_{R_i}]$, which are the left context, target, and right context, respectively ($L_i + C_i + R_i = N_i$). The polarity vector is a one hot encoded vector of dimension three, meaning that $y_i \in \mathcal{R}^3$. We perform similar operations on all three vectors, so we take the left vector as an example. The first step is to apply the right zero padding for the left context. This creates $[l_1, l_2, ..., l_{L_i}, o_1, o_2, ..., o_{Q_l - L_i}]$ where Q_l is the maximum length of the left context vector and $o_i \in \mathcal{R}^d$ is an embedding with zeros (d is the embedding dimension). We obtain $S_{i,l} \in \mathcal{R}^{d \times Q_l}$. Afterwards, the following linear interpolation is done:

$$\tilde{S}_{ijl} = \lambda S_{i,l} + (1 - \lambda) S_{j,l} \quad (3)$$

$$\tilde{y}_{ij} = \lambda y_i + (1 - \lambda) y_j \quad (4)$$

where $\tilde{S}_{ijl}$ is the left context vector of the augmented record, $\tilde{y}_{ij}$ is the polarity of the augmented record, and λ is a random value drawn from the distribution $Beta(\alpha, \alpha)$. Mixup regularizes the NN such that the linear behavior is preferred to more complex functions. This reduces overfitting when training, and makes the training more robust when having corrupt labels. Zhang et al. [23] found that an $\alpha \in [0.1, 0.4]$ worked well, as values higher than this resulted in underfitting. This is because linearly interpolating with a low α results in a high probability that λ takes a value in the tails (so either close to zero or close to one). This way, the augmented data is slightly interpolated, which should prevent overfitting, but not so much that it underfits. In this paper we will therefore experiment with $\alpha \in [0.1, 0.4]$.

5 EVALUATION

This section presents the best results of the performed experiments. Section 5.1 presents the baseline method. Section 5.2 gives the results of the augmentation techniques. Last, Section 5.3 discusses some insights in the obtained results.

5.1 Baseline and Comparison

While reproducing the results presented in Wallaart and Frasincar [20], we found that the ontology reasoner has an accuracy of 0.87 for the data of 2016, similar to the reported accuracy, and 0.83 for the dataset of 2015, which is slightly higher than the reported accuracy. However, as our paper aims to investigate the effect of data augmentation, this is not troublesome. The reproduced model is used as a baseline, and the accuracy of the predictions on the test set of SemEval 2015 and SemEval 2016 is the employed performance measure.

5.2 Augmentation techniques

In Table 3, the baseline is compared to the individual results of the augmentation techniques. While the EDA stands for the combined solution with the four methods presented in Section 4.3.1 (random insertion, random deletion, random swap and synonym replacement), adjusted EDA represents the techniques introduced in Section 4.3.2 (random insertion and synonym replacement with word sense disambiguation, and target swap). In terms of both SemEval datasets, it can be seen that the adjusted EDA outperforms the original EDA. Both EDA approaches lead to a performance boost with respect to the baseline HAABSA between 0.3-1 and 0.2-0.5 (EDA-adjusted EDA) percentage points of accuracy for SemEval 2015 and SemEval 2016 test datasets, respectively.

Furthermore, the backtranslation technique is assessed using Dutch, Spanish, and Japanese. This method has varying results, sometimes increasing or decreasing accuracy for different languages. We suspect that backtranslation generates varying results because the target expression had to remain fixed. This way, the augmentation techniques were performed on a part of a sentence

Table 3: Results for different data augmentation methods. Best results per technique are given in bold.

Model	2015			2016		
	training acc.	testing acc.	#aug.	training acc.	testing acc.	#aug.
HAABSA	90.9 %	77.9%	0	85.7%	83.9%	0
HAABSA + EDA	85.8%	78.2%	5116	96.2%	84.1%	7520
HAABSA + EDA adj.	**95.2%**	**78.9%**	3837	**96.5%**	**84.4**%	5640
HAABSA + BT NL	89.9%	78.0%	1278	78.9%	83.8%	1880
HAABSA + BT ES	85.1%	77.6%	1278	82.0%	84.3%	1880
HAABSA + BT JA	86.9%	77.9%	1278	81.3%	**84.4%**	1880
HAABSA mixup $\alpha = 0.1$	88.0%	77.6%	1278	80.8%	83.8%	1880
HAABSA mixup $\alpha = 0.2$	85.0 %	77.9%	1278	80.4%	84.3%	1880
HAABSA mixup $\alpha = 0.3$	89.2%	77.9%	1278	83.8%	84.1%	1880
HAABSA mixup $\alpha = 0.4$	82.8%	77.9%	1278	79.6%	84.1%	1880

Table 4: Results of EDA adjusted ratio variations Best results per technique are given in bold.

Model	2015			2016		
	training acc.	testing acc.	#aug.	training acc.	testing acc.	#aug.
HAABSA + EDA adj. 1:1	**95.2%**	**78.9%**	3837	**96.5%**	**84.4**%	5640
HAABSA + EDA adj. 1:3	86.1%	78.1%	3834	80.7%	84.1%	5640
HAABSA + EDA adj. 3:1	84.4%	77.9%	3834	91.1%	83.5%	5640

instead of on a sentence as a whole. For backtranslation, we translate the left and the right contexts to the target language and back, keeping the target fixed. This way, the translation engine has to translate different parts of a sentence. While the backtranslation from Dutch slightly outperforms the other languages and the baseline with 0.1 percentage points for the case of SemEval 2015 test dataset, the backtranslation from Japanese improve the accuracy of SemEval 2016 test dataset till the level of adjusted EDA. As a result, we consider that the Japanese language yields the best results for the backtranslation.

The last data augmentation technique we evaluate is the mixup with $\alpha \in [0.1, 0.4]$. We notice that the setting of the α parameter to 0.2 produces the best accuracy, improving the accuracy of HAABSA applied on the SemEval 2016 test data with 0.4 percentage points. On the contrary, the accuracy on the SemEval 2015 test data stays constant or slightly decreases ($\alpha = 0.1$). For mixup, we had to linearly interpolate between two different left contexts, two different targets, and two different right contexts. As sentences can have very different structures, linear interpolation between parts of sentences appears to have a small effect on the accuracy.

We empirically found that the 1:1 ratio of augmentations and original records yield the highest results, after we compared a 3:1 ratio and a 1:3 ratio. Overall, we can conclude that the best technique is the adjusted EDA. The largest boost in performance is given by the target swap, followed by adjusted synonym replacement, and then adjusted random insertion. However, given that the accuracy of HAABSA also takes into account the ontology, the effect of the adjusted EDA technique, as it is shown in Table 3, is diminished. Knowing that the accuracy of the baseline multi-hop LCR-Rot is 72.9% and 78.9% on the SemEval 2015 and SemEval 2016 test data, respectively, the adjusted EDA leads to an increase of 1.3 percentage points on the SemEval 2016 dataset, and an increase of 2 percentage points on the SemEval 2015 dataset.

5.3 Insights

To understand why some of data augmentations techniques are more effective than others, we need to assess their capacity to generate meaningful sentences. In doing this, we randomly select two sentences from the employed datasets. Random deletion and random insertion are left out, as these follow logically from the given examples, similarly to the backtranslation of other sentences. Additionally, the categories for both sentences are the same, namely *SERVICE#GENERAL*, such that the target swap can be performed. As can be seen from Table 5, the adjusted EDA methods (synonym replacement and target swap) mainly have the desired effect of creating new sentences with extra information. This inference straightens the idea already suggested in Table 3, according to which adjusted EDA is the most effective data augmentation technique. On the other hand, the simple EDA methods usually seem to pollute the data in a way that does not add extra information or adds wrong information. For example, when the algorithm replaces the word “us" with the atomic number “92". This is because “us" can be interpreted as the plural of “u", where “u" is the atomic symbol for Uranium.

Another interesting observation is the way backtranslation works. In Table 5, the word “the", when translated to Japanese and back, becomes “of". Additionally, when translating a part of a sentence to Japanese and back, the translation engine attempts to create sentences from only part of a sentence. Therefore, the context on the right of both sentences can be read as a standalone sentence. Even if the meaning of the initial sentence is usually kept, sometimes, the backtranslation process might negatively influence the overall accuracy.

Lastly, the ratio of original sentences and augmented sentences has been modified to 3:1, 1:1 and 1:3. This means that the original data is used thrice for the 3:1, and that the augmentations are done thrice in the 1:3. However, this experiment showed that a 1:1 ratio yielded the best results, as it is shown in Table 4.

Table 5: Individual examples of augmentations, aspects are in bold

Type augmentation	Sentence 1	Sentence 2
Original	the **hostess** is rude to the point of being offensive	The **waitress** was very patient with us and the food is phenomenal!
EDA original random swap	*being* **hostess** is rude to point of *the* offensive	The **waitress** *food* very with patient us and the *was* is phenomenal!
EDA original synonym replacement	the **hostess** is rude to the *breaker point* of being nauseous	The **waitress** was very patient with *atomic number 92* and the food is phenomenal!
Adjusted EDA synonym replacement	the **hostess** is *uncivil* to the point of being *unsavory*	The **waitress** was very patient with us and the *nourishment* is phenomenal!
Adjusted EDA Target Swap	the **waitress** is rude to the point of being offensive	The **hostess** was very patient with us and the food is phenomenal!
Backtranslation Japanese	of **hostess** it's rude enough to cause discomfort.	of **waitress** very patient and the food is amazing!

6 CONCLUSION

This paper focuses on data augmentation methods in the field of ABSA on a sentence level by extending the state-of-the-art model proposed by Wallaart and Frasincar [20]. The first proposed approach is called EDA, and it is inspired by the work of Wei and Zou [21], who created a general set of four methods applicable for all NLP classification tasks. As this approach proves to be too naive to properly address the ABSA task, some adjustments are introduced based on word sense disambiguation and swaps between targets that share the same category. Besides the EDA methods and their variations, we also investigated the effect of the backtranslation and the mixup methods as a linear interpolation between feature vectors. Both methods only improve the HAABSA applied on SemEval 2016 test data, while generating similar results with the baseline on the SemEval 2015 data. Overall, the adjusted EDA is the most effective data augmentation method, increasing the accuracy of the SemEval 2015 and SemEval 2016 test datasets with 1.0 and 0.5 percentage points, respectively.

Furthermore, in this paper, the data augmentation techniques were used on all sentences. It would be interesting to analyze which kind of sentences give the highest yield when being augmented and which sentences harm the accuracy when being augmented. One direction to consider is the effect of the length of a sentence on the data augmentation. This analysis can also be extended for mixup, in the sense that instead of selecting random sentences used for mixup, it might be useful to select only the sentences with similar lengths for their contexts. This implies that the augmentation techniques do not necessarily have to be used upon all sentences, but can be used only on the ones with similar characteristics.

REFERENCES

[1] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* O'Reilly Media, Inc.

[2] Craig Bloem. 2017. *84 Percent of People Trust Online Reviews as Much as Friends. Here's How to Manage What They See.* (2017). https://www.inc.com/craig-bloem/84-percent-of-people-trust-online-reviews-as-much-.html#:~:text=Research%20shows%20that%2091%20percent,one%20and%20six%20online%20reviews

[3] Christiane Fellbaum. 1998, ed.. *WordNet: An Electronic Lexical Database.* MIT Press.

[4] Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. Augmenting Data with Mixup for Sentence Classification: An Empirical Study. *arXiv preprint arXiv:1905.08941* (2019).

[5] Matthew Honnibal. 2013. A Good Part-of-Speech Tagger in about 200 Lines of Python. (2013). https://explosion.ai/blog/part-of-speech-pos-tagger-in-python

[6] Michael Lesk. 1986. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In *5th Annual International Conference on Systems Documentation, (SIGDOC 1986).* ACM, 24–26.

[7] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing LSTM language models. (2017). *arXiv preprint arXiv:1708.02182.*

[8] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *2014 conference on empirical methods in natural language processing (EMNLP).* ACL, 1532–1543.

[9] Luis Perez and Jason Wang. 2017. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *arXiv preprint arXiv:1902.09314* (2017).

[10] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. SemEval-2015 Task 12: Aspect Based Sentiment Analysis. In *9th International Workshop on Semantic Evaluation (SemEval 2015).* ACL, 486–495.

[11] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2016. SemEval-2016 Task 5: Aspect Based Sentiment Analysis. In *10th International Workshop on Semantic Evaluation (SemEval 2016).* ACL, 19–30.

[12] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *2016 Conference on Empirical Methods in Natural Language Processing, (EMNLP 2016).* ACL, 2383–2392.

[13] Kim Schouten and Flavius Frasincar. 2016. Survey on Aspect-Level Sentiment Analysis. *IEEE Transactions on Knowledge and Data Engineering* 28, 3 (2016), 813–830.

[14] Kim Schouten and Flavius Frasincar. 2018. Ontology-Driven Sentiment Analysis of Product and Service Aspects. In *15th Extended Semantic Web Conference (ESWC 2018) (LNCS),* Vol. 1084. Springer, 608–623.

[15] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *54th Annual Meeting of the Association for Computational Linguistics, (ACL 2016).* ACL.

[16] Sam Shleifer. 2019. Low Resource Text Classification with ULMFiT and Backtranslation. *arXiv preprint arXiv:1905.08941* (2019).

[17] Liling Tan. 2014. Pywsd: Python Implementations of Word Sense Disambiguation (WSD) Technologies [software]. (2014). https://github.com/alvations/pywsd

[18] Tripadvisor. 2017. *TripAdvisor Network Effect and the Benefits of Total Engagement.* (2017). https://www.tripadvisor.com/TripAdvisorInsights/w828#:~:text=TripAdvisor%20and%20Total%20Engagement&text=For%20accommodations%2C%20engaging%20with%20TripAdvisor,%3Dmore%20customers%3Dmore%20revenues

[19] Florentina Vasilescu, Philippe Langlais, and Guy Lapalme. 2004. Evaluating Variants of the Lesk Approach for Disambiguating Words. In *Fourth International Conference on Language Resources and Evaluation (LREC, 2004).* ELRA.

[20] Olaf Wallaart and Flavius Frasincar. 2019. 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. Springer, 363–378.

[21] Jason W. Wei and Kai Zou. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, (EMNLP-IJCNLP, 2019).* ACL, 6381–6387.

[22] Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and Accurate Reading Comprehension by Combining Self-Attention and Convolution. In *6th International Conference on Learning Representations (ICLR 2018).*

[23] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond Empirical Risk Minimization. In *6th International Conference on Learning Representations, (ICLR 2018).*

[24] Shiliang Zheng and Rui Xia. 2018. Left-Center-Right Separated Neural Network for Aspect-based Sentiment Analysis with Rotatory Attention. (2018). *arXiv preprint arXiv:1802.00892.*