Using Data Augmentation in a Hybrid Model for Aspect-Based Sentiment Analysis

by

Tomas Liesting

Bachelor Thesis
Student ID: 453177
Thesis supervisor: dr. Flavius Frasincar
Co-reader: prof. dr. Ilker Birbil

Bachelor of Science in Econometrics and Operations Research
Erasmus University Rotterdam
Business Analytics and Quantitative Marketing
June 2020

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 this thesis, we investigate the effect of data augmentation on a state-of-the-art hybrid model for aspect-based sentiment analysis. We apply modified versions of easy data augmentation (EDA), backtranslation and word mixup. We evaluate the proposed techniques on the SemEval 2015 SemEval 2016 datasets. The best result is 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 model.

Contents

Chapter 1

Introduction

Currently, many online platforms compare and review products, locations, and services. Social media sites like IMDB, Facebook, Yelp, and many others contain, compare, and analyze millions of reviews. These texts contain valuable information for many companies on how and what to improve, and influence consumers which product to buy or which location to go to. Research has demonstrated that 91% of the people regularly read online reviews, and around 84% trust these online reviews (Bloem, 2017), implying that these opinionated texts influence the decisions of the majority of the consumers. The relevance of these reviews therefore, also extends to the providers of services or products. Imagine, for example, reviews on restaurants. Knowing what the guests' general opinions are and what the opinions are about specific aspects of the restaurant like the food, atmosphere, or price level could suggest selective and effective improvements (Yan et al., 2015). With this information, a restaurant can finetune its recipes, decoration, and prices to maximize profits.

Therefore, it is not surprising that the analysis of these opinionated texts awoke much interest in the past years. Many companies want to know what consumers think about their business and what they should improve. However, the amount of information available is enormous. Tripadvisor by itself already has over 435 million reviews (Tripadvisor, 2017). Manually reading and interpreting these is neither possible, nor efficient, and a computer that can process and analyze this information is desirable for many parties. Many companies consequently try to apply sophisticated techniques to extract sentiments from written reviews (Grimes, 2017). These companies analyze and classify these reviews as, for example, positive or negative.

1.1 Aspect-Based Sentiment Analysis

The field that focuses on computational techniques for the automatic analysis and representation of the human language is Natural Language Processing (NLP) (Cambria and White, 2014). A subset of NLP is sentiment analysis or opinion mining. This field focuses on the extraction of sentiments about products or services (Liu, 2015). Extracting more specific aspects as described above, is done in a subfield of sentiment analysis called aspect-based sentiment analysis (ABSA) (Schouten and Frasincar, 2016). Aspects can take many shapes and forms and are dependent on the domain/product. Aspects of phones can be the camera, battery, or size, while for restaurants, examples of aspects can be food, price, or atmosphere. ABSA has three crucial components (Schouten and Frasincar, 2016). First, the target words that the sentiment is about have to be extracted (called target extraction). Second, the aspects have to be detected depending on the 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) results in ABSA (partly) use neural networks (NNs) (Do et al., 2019). NNs are models based on the working of a human brain (Kubat, 1999). As the name suggests, most NNs consist of a network of neurons, divided into layers. A layer of a NN are neurons that can operate in parallel. The second layer is dependent on the outcome of the first layer, but the nodes within the first layer are not dependent on each other—every neuron within a NN consists of a linear transformation of the data and an activation function. An activation function transforms the outcome of the linear transformations in the desired output for either the output or the next stage.

In NNs, many variations of architecture are possible. Also, different methods to transform the data before doing computations are used. An example is the use of word embeddings (Mikolov et al., 2013a; Pennington et al., 2014), which are distributed representations or feature vectors of words in a vector space of a fixed dimension and generally improve NLP solutions (Mikolov et al., 2013b). Another way to improve NNs is by using data augmentation, mainly used in analyzing images (Perez and Wang, 2017). 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 (Perez and Wang, 2017).

1.2 Hybrid Approach for Aspect-Based Sentiment Analysis

One model achieving high accuracy in sentiment classification is the Hybrid Approach for Aspect-Based Sentiment Analysis (HAABSA) (Wallaart and Frasincar, 2019). This two-step hybrid model (TSHM) combines the use of an ontology-based reasoner with a neural network and attempts to determine the polarity of the opinion on a given aspect. The neural networks are based on an LCR-Rot-Hop model, meaning that the context is split into left, target, and right context, going through three respective bidirectional LSTM cells, after which it applies rotatory attention. It furthermore uses word embeddings from GloVe (Pennington et al., 2014) and obtains SOTA results for several ABSA sentiment evaluation tasks, and will therefore be used in this thesis.

1.3 Research Objective

In this paper, we propose an extension to the HAABSA model to improve the quality of the sentiment predictions. Data augmentation will be used before the training of the NN in order to increase the training data. In the field of ABSA, to the best of our knowledge, data augmentation is not yet used. Using this is an exciting extension of current approaches. This leads to the following research question:

How can data augmentation improve a Two-Step Hybrid model used for Aspect-Based Sentiment Analysis?

Three subquestions arise out of the main research question:

- What data augmentation techniques for NLP are currently available?
- Which techniques are appropriate for the enhancement of the TSHM?
- How can the current techniques be extended to use them in a TSHM?

1.4 Thesis Structure

The paper is organized as follows. In Chapter 2, relevant data augmentation techniques for NLP used by other researchers are presented. In Chapter 3 the datasets used for training and evaluation are introduced. Afterward, in Chapter 4, we explain how the techniques discussed in Chapter 2 can be used for ABSA, and we propose several extensions on this work. In Chapter 5, the effect of these

techniques on the accuracy of the model is discussed. In Chapter 6, we present the implications of our research and propose ideas for future research.

Chapter 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, like in Perez and Wang (2017). In this paper, the authors evaluate several techniques where a given image is rotated, tilted, cropped, or shifted, improving the predictions. In NLP, some attention has been given to data augmentation as well, which will be discussed in this Chapter.

First of all, Wei and Zou (2019) attempted to 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 stop-words, are selected and replaced with synonyms. Secondly, they use Random Insertion (RI), where they insert a random synonym of a random word at a random position in a sentence. Their third method is Random Swap (RS), where two words in a sentence are randomly swapped, and lastly, they propose Random Deletion (RD), 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 this data augmentation task has been performed using the full dataset. They do note that their effect is more substantial on smaller datasets in comparison to larger datasets.

Another way to use data augmentation is introduced by Sennrich et al. (2016), used 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. (2018) 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 (Rajpurkar et al., 2016). 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. They 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 (2019) used a combination of the previous methods on a pretrained multilayered AWD-LSTM model (Merity et al., 2017) on an IMDB movie dataset. In this analysis, the authors found that using EDA on the full dataset does not lead to substantial improvements and that backtranslation has a small gain. Using a subset of the dataset does lead to improved performances with backtranslation. Wei and Zou (2019) 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 use of mixup, originally introduced by Zhang et al. (2018) for image recognition. Their idea was to (linearly) interpolate between feature vectors of an image, which should be identical to the interpolation of the associated target vectors. They 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 drawn from a Beta(α, β) distribution, where the authors put $\alpha = \beta$ 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 in Equation 2.1

$$\begin{aligned} \tilde{x}_{ij} &= \lambda x_i + (1 - \lambda) x_j \\ \tilde{y}_{ij} &= \lambda y_i + (1 - \lambda) y_j \end{aligned} \tag{2.1}$$

As word embeddings essentially are feature vectors, similar to images, this technique can also be applied to NLP tasks (Guo et al., 2019). Their 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. Secondly, 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 accuracy in both CNN and LSTM methods significantly.

To summarize, the usage of easy data augmentation (or EDA) as introduced by Wei and Zou (2019) is an interesting development, though the effect of this method appears to be small when using (contextual) word embeddings. Back-translation has proven to provide small to substantial improvements on large and small datasets, respectively. Both sentence mixup and word mixup appear to be interesting developments in large datasets and smaller datasets. In the field of ABSA, data augmentation has not yet been used to the best of our knowledge.

Chapter 3

Data

This chapter gives an overview of the used data. In Section 3.1 we give a general overview of the data, Section 3.2 discusses the preprocessing of the data, and Section 3.3 gives some insights into the employed datasets.

3.1 SemEval 2015 and 2016

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 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 (Pontiki et al., 2015) and task 5 subtask 2 of SemEval 2016 (Pontiki et al., 2016). The goal of these tasks is to predict the polarity of a sentence about a given aspect. This way, the performance of the data augmentation can be compared to the performance of other models.

The datasets both contain restaurant reviews with one to several sentences in 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 3.1. In this figure the aspect is called the *target*, a general category is given in the *category* field, and the polarity of the sentiment is given in the *polarity* field. The dataset of SemEval 2015 has a similar structure.

```
<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 3.1: Sample sentence of the SemEval 2016 data

3.2 Preprocessing

In order to use the data, some preprocessing has to be done, similarly to the work of Wallaart and Frasincar (2019). First of all, the implicit targets have to be removed. These are the targets that do not have a specific target word in the sentence, and can therefore not be processed by the used algorithms.

Additionally, the used word vectors are from the GloVe framework (Pennington et al., 2014). These word embeddings are of dimension 300, created by using local information and a global co-occurrence matrix. This way, GloVe significantly outperforms skip-gram or CBOW methods on many tasks (Pennington et al., 2014). Furthermore, data augmentation seems to have more effect on non-contextual word embeddings than on contextual word embeddings (Wei and Zou, 2019).

3.3 Description of Data

Table 3.1 shows the frequency of the training data and test data of both datasets. In all training sets, neutral sentences are the least frequent, and positive are the most frequent. Table 3.2 gives an overview of the categories in the SemEval datasets. Food quality, general service, and general ambiance are the most numerous 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.

Table 3.1: 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

Table 3.2: 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
DRINKSSTYLE_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

Chapter 4

Framework

This chapter gives an overview of the framework used in this thesis. Section 4.1 gives a short overview of the ontology reasoner used in HAABSA. Afterward, Section 4.2 gives an overview of the used machine learning method and the used word embeddings. Finally, Section 4.3 explains existing data augmentation techniques and proposed extensions taking in account ABSA specifics.

4.1 Ontology Reasoner

HAABSA is a hybrid model for ABSA that consists of two stages. First of all, an ontology reasoner is used, which is similar to the reasoner used by Schouten and Frasincar (2018). 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 that the word *fish* is linked to the category *FOOD#QUALITY*.

The *SentimentMention* class determines whether the sentiment of a sentiment expression is positive or negative for the specific aspect. There are three types of *SentimentMention*. Type 1 concerns concepts that have the same sentiment for every aspect. An example is the word *bad*, which is always negative, no matter the context. Type 2 words 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. The last type, type 3, is 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 (Schouten and Frasincar, 2018). The best model in Wallaart and Frasincar (2019), 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 context, 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 (Pennington et al., 2014). The advantage of GloVe is that, unlike 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 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 will discuss the adaptations of the methods appropriate for ABSA and suggested extensions.

4.3.1 Easy Data Augmentation for ABSA

The idea of easy data augmentation (EDA) proposed by (Wei and Zou, 2019), 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

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

input. Secondly, the target expression should remain present and together. Removing the target words or splitting the expression up would make a classification of the polarity difficult, as the machine learning algorithm cannot handle this.

Keeping the above considerations in mind, we propose several adaptations to make EDA compatible with ABSA. Afterward, we introduce variations and extensions to these methods.

Random Insertion. Random insertion is the process of selecting a random word in the sentence, find a synonym on WordNet (Fellbaum, 8 ed) using the natural language toolkit (NLTK) (Bird et al., 2009), and insert 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 the left, center, 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 vector. 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 together, as the target vector needs to stay in place. 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 on WordNet, and replaces the word with its synonym. We apply a similar procedure to ensure that the target is not replaced by replacing the target with a fixed expression and making sure that this expression cannot be selected for replacement.

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

Algorithm 1: synonym replacement in original EDA

Data: *sentences*, sentences of the SemEval dataset; *tar*, target word in the sentece; α, percentage of words replaced

Result: Augmented sentences

begin

Set $AugmentedSentences = \phi$

foreach $originalSentence \in sentences$ **do**

$sentence \longleftarrow$replace tar with $\$t\$$ in $originalSentence$

$sentence \longleftarrow$split $sentence$ by space

$numberAugmentations \longleftarrow \alpha * length(sentence)$

while $i < numAugmentations$ **do**

$word \longleftarrow takeRandomWordNotTarget(sentence)$

$syn \longleftarrow takeRandomWordNetSynonym(word)$

$sentence \longleftarrow$replace $word, syn$

$augmentedSentences \longleftarrow augmentedSentences \cup sentence$

return *augmentedSentences*

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 dataset consists of about 2000 sentences, speed is not a big issue for this problem either. 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 is returned. Take the example in Figure 3.1, 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, 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, which pollute the sentences rather than enhancing it. The process of selecting the correct meaning of the word is called word sense disambiguation (WSD) (Vasilescu et al., 2004).

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 (Honnibal, 2013).

Afterward, the true sense of the word is determined. An easy algorithm for WSD is proposed by Lesk (1986). 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 significantly outperformed by a simplified version of Lesk (Vasilescu et al., 2004). 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 possible meaning with the maximum overlap, where the overlap is calculated as in Equation 4.1.

$$Overlap(w, s_i) = |D_i \cap C| \tag{4.1}$$

This method is called simplified Lesk, and has been adapted to the WordNet library (Fellbaum, 8 ed) rather than a regular dictionary. This led to significantly better results than the original Lesk, and is therefore the method that we will use. 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 its corresponding definitions, making it a useful tool for WSD. For the simplified Lesk algorithm, the Pywsd library is used (Tan, 2014).

This process 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 3.1), a category is given together with its target word and its position. We can therefore replace random swaps by swapping target words of similar 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.

4.3.3 Backtranslation

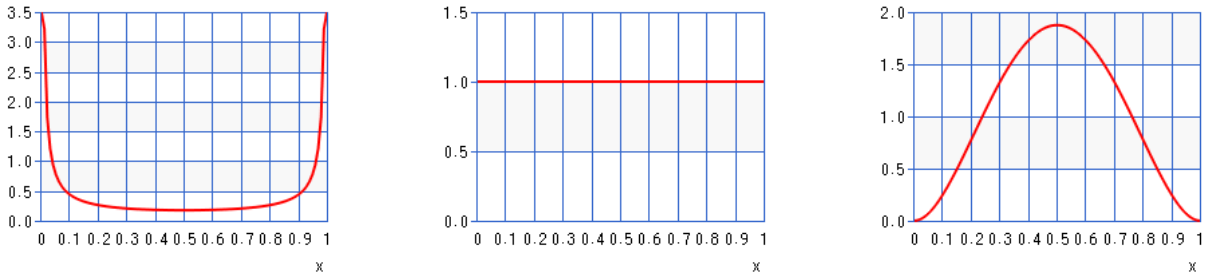
For backtranslation we face a similar problem as with EDA. When translating a sentence to another language and back, the target expression could have changed, resulting in not knowing at which position in the sentence the target is. To tackle this problem, the left context and the right context are independently 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 target vectors (Zhang et al., 2018). The technique of mixup is used initially for image classification tasks, but recently had been done using mixup for NLP. For this thesis, we introduce a variant of mixup appropriate for the two-step hybrid model for ABSA.

In order to properly apply mixup at a sentence level, Guo et al. (2019) introduced word-mixup. This way the input sentences are zero-padded on the right in order to make them the same length, after which the linear interpolation is done. As our model makes use of a left context, right context, and target vector, we propose the following 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 right zero padding the contexts and target on the right. This creates $[l_1, l_2, ..., l_{L_i}, o_1, o_2, ..., o_{Q_l - L}]$ where Q_l is the maximum length of the left context vector and $o_i \in \mathcal{R}^d$ is an embedding with zeros. We obtain $S_{i,l} \in \mathcal{R}^{d \times Q_l}$. Afterwards, the following linear interpolation is done:

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



(a) $\alpha = 0.1$ (b) $\alpha = 1$ (c) $\alpha = 3$

Figure 4.1: Plots of the probability density function of the $Beta(\alpha, \alpha)$ distribution for different α

$$\tilde{S}_{ijl} = \lambda S_{i,l} + (1 - \lambda) S_{j,l} \tag{4.2}$$

$$\tilde{y}_{ij} = \lambda y_i + (1 - \lambda) y_j \tag{4.3}$$

where $\tilde{S}_{ijl}$ is the left context vector of the augmented file, $\tilde{y}_{ij}$ is the polarity of the augmented record λ is a random value drawn from $Beta(\alpha, \alpha)$. Mixup regularizes the NN such that linear behavior is preferred above more complex functions. This reduces overfitting when training, and makes the training more robust when having corrupt labels. Zhang et al. (2018) 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), which can be seen in Figure 4.1a. This way, the augmented data is slightly interpolated, which should prevent overfitting, but not so much that it underfits. In this thesis we will therefore experiment with $\alpha \in [0.1, 0.4]$.

Algorithm 2: synonym replacement in adjusted EDA

Data: *sentences*, sentences of the SemEval dataset; tar, target word in the sentece; α, percentage of words replaced

Result: Augmented sentences

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 $i < numAugmentations$ **do**

$word \longleftarrow takeRandomWordNotTarget(words)$

$maxOverlap \longleftarrow 0$

bestSense ⟵most frequent sense of word

foreach *sense* ***in*** senses of word with same part-of-speech in *partsOfSpeech* **do**

signature ⟵set of words in the gloss and examples

$overlap \longleftarrow computeWordnetOverlap(signature, originalSentence)$

if $overlap > maxOverlap$ **then**

$maxOverlap \longleftarrow overlap$

$bestSense \longleftarrow sense$

sentence ⟵replace *word* with *bestSense* in *originalSentence*

$augmentedSentences \longleftarrow augmentedSentences \cup sentence$

return *augmentedSentences*

Algorithm 3: Target swap in adjusted EDA

Data: *sentences*, sentences of the SemEval dataset; asp_i, aspect of sentence i; *categories* the categories in the SemEval datasets

Result: Augmented sentences

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$_j \longleftarrow replace(asp_j, asp_i, sentence_j)$

$augmentedSentences \longleftarrow augmentedSentences \cup sentence_i \cup sentence_j$

return *augmentedSentences*

Algorithm 4: Backtranslation

Data: *sentences*, sentences of the SemEval dataset; *tar*, target word in a sentence; *lang*, target language to translate to

Result: Augmented sentences

begin

Set $AugmentedSentences = \phi$

foreach $originalSentence \in sentences$ **do**

$left$, $target$, $right \longleftarrow split(originalSentence, tar)$

$translatedLeft \longleftarrow translate(left, lang)$

$translatedRight \longleftarrow translate(right, lang)$

$backtranslatedLeft \longleftarrow translate(translatedLeft, Eglish)$

$backtranslatedRight \longleftarrow translate(translatedRight, English)$

$backtranslated \longleftarrow backtranslatedLeft + target + backtranslatedRight$

$augmentedSentences \longleftarrow augmentedSentences \cup backtranslated$

return *augmentedSentences*

Chapter 5

Evaluation

This chapter presents the results of the experiments in Chapter 4. Section 5.1 presents the reproduction of the HAABSA paper. The following sections will present the performance of the individual data augmentation techniques, after which they will be analyzed and varied.

5.1 Baseline and Comparison

While reproducing the results presented in Wallaart and Frasincar (2019), we find 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 this thesis 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 validation set of SemEval 2015 and SemEval 2016 will be the measure.

Furthermore, for a fair comparison of our methods, a constant ratio between original and augmented data is maintained. This means that if we have n original training records and we perform one augmentation technique we will also have n augmented sentences. However, if we perform four augmentations on each sentence, we will have $4n$ augmentations, so we will use four times the original data as well, such that a 1:1 ratio of original data and augmented data is maintained. In Section 5.5, the ratio will be varied based using the most successful method.

Table 5.1: Results of individual EDA augmentations. Best results are given in bold.

Model	2015			2016		
	in-sample	out-of-sample	#aug	in-sample	out-of-sample	#aug
HAABSA	90.9 %	77.9%	0	**85.7%**	83.9%	0
HAABSA + EDA_ri	86.5%	78.4%	1278	80.7%	83.8%	1880
HAABSA + EDA_rd	83.2%	77.7%	1278	81.5%	83.6%	1880
HAABSA + EDA_rs	85.8%	**78.7** %	1278	79.2%	83.7%	1880
HAABSA + EDA_sr	85.1%	77.9%	1278	80.6%	83.6%	1880
HAABSA + EDA_sr_wsd	**91.4**%	78.1%	1278	83.0%	83.6%	1880
HAABSA + EDA_ri_wsd	86.6%	77.3%	1278	82.1%	83.3%	1880
HAABSA + EDA_sw_tar	84.4%	78.6%	1278	81.6%	**84.1%**	1880

5.2 Easy Data Augmentation

First, each augmentation technique is analyzed individually. These results are given in Table 5.1. The target swap improves the out-of-sample predictions in 2016 with 0.2 percentage points and 2015 with 0.7 percentage points. Furthermore, random deletion yields a decrease in out-of-sample accuracy in both 2015 as well as 2016, which is in line with the expectations. Afterward, the effects of combined EDA-methods are presented. The effects are presented in Table 5.2. We distinguish between original EDA, which includes *EDA_ri*, *EDA_rd*, *EDA_rs* and *EDA_sr*, and the adjusted EDA, which includes *EDA_ri_wsd*, *EDA_sw_tar* and *EDA_sr_wsd*. We see that adjusted EDA increases the accuracy of the 2015 dataset by 1 percentage point and 0.5 percentage point in 2016, which is more than the original EDA techniques. Furthermore, as the original EDA has four augmentations and the adjusted method has three, the original method trains on four times the original data and the augmentations, and the adjusted method trains on three times the original data and the augmentations.

Table 5.2 shows that the adjusted version of EDA outperforms both the original HAABSA as well as HAABSA with original EDA in the 2015 and 2016 SemEval datasets with $0.5-1$ percentage points. Also, the original EDA outperforms HAABSA with $0.2-0.3$ percentage points. This follows the literature, as Wei and Zou (2019) concluded that the effect of EDA is small but positive. One thing to keep in mind is that an increase of 1 percentage point in the out of sample predictions

Table 5.2: Results of combined EDA augmentations. Best results are given in bold.

Model	2015			2016		
	in-sample	out-of-sample	#aug	in-sample	out-of-sample	#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

Table 5.3: Results of individual backtranslation augmentations. Best results are given in bold.

Model	2015			2016		
	in-sample	out-of-sample	#aug	in-sample	out-of-sample	#aug
HAABSA	**90.9 %**	77.9%	0	**85.7%**	83.9%	0
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

means that the accuracy of LCR-Rot hop increased relatively more. This is because the out-of-sample predictions are a combination of the ontology reasoner and the machine learning algorithm, and the accuracy of the ontology reasoner remains constant. Therefore, the adjusted version of EDA yields an increase of 1.2 percentage points in the LCR-Rot hop on the SemEval 2016 dataset and an increase of 2.3 percentage points on the SemEval 2015 dataset.

5.3 Backtranslation

In the experiments, three different languages are used for backtranslation, namely Dutch, Spanish, and Japanese. The results of the individual backtranslations are given in Table 5.3. The effect of backtranslation differs across languages and across datasets, where Japanese results in the highest increase in SemEval 2016 of 0.5 percentage point, and Dutch in SemEval 2015, with an increase of 0.1 percentage point. From these results, the effectiveness of backtranslation seems low, as none of the backtranslations result in a strictly better performance on both datasets.

Table 5.4: Results of individual mixup augmentations. Best results are given in bold.

Model	2015			2016		
	in-sample	out-of-sample	#aug	in-sample	out-of-sample	#aug
HAABSA	**90.9%**	**77.9%**	0	**85.7%**	83.9%	0
HAABSA mixup $\alpha = .1$	88.0%	77.6%	1278	80.8%	83.8%	1880
HAABSA mixup $\alpha = .2$	85.0 %	**77.9%**	1278	80.4%	**84.3%**	1880
HAABSA mixup $\alpha = .3$	89.2%	**77.9%**	1278	83.8%	84.1%	1880
HAABSA mixup $\alpha = .4$	82.8%	**77.9%**	1278	79.6%	84.1%	1880

5.4 Mixup

Table 5.4 presents the results for the experiments of mixup. Mixup seems to have a small effect on the accuracy of the models, yielding a 0.4 percentage point increase in the SemEval 2016 dataset and no increase in accuracy in the SemEval 2015 dataset for $\alpha = 0.2$.

5.5 Variations and Analysis

In this Section some analyses and insights in the results are given. First, in Section 5.5.1, some examples of augmented sentences are given and discussed. Secondly, the effect of the ratio of original data and augmented data on the results is discussed in section 5.5.2. As EDA adjusted appeared to have the highest increase in accuracy, this method is investigated with varying ratios.

5.5.1 Individual examples of the results

In this section, some examples are given of the augmented sentences. Table 5.5 presents these examples. 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 this table, the data augmentation techniques sometimes have the desired effect, creating a new sentence with extra information. This is the case in the target swap and in some cases of the synonym replacement. However, in many cases, the augmentation techniques seem to pollute the data in a way that does not add extra information or adds wrong

Table 5.5: 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 RS	being the is rude to **hostess** point of the offensive	The **waitress** food very with patient us and the was is phenomenal!
EDA original SR	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!
EDA adjusted SR	the **hostess** is uncivil to the point of being unsavory	The **waitress** was very patient with us and the intellectual nourishment is phenomenal!
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!
EDA backtranslation JA	of **hostess** it's rude enough to cause discomfort.	of **waitress** very patient and the food is amazing!

information. For example, when the algorithm replaces the word us with atomic number 92. This is because us can be interpreted as the plural of u, where u is the atomic number for Uranium. This is a large disadvantage of the original EDA method.

Another interesting observation is the way backtranslation works. In Table 5.5, 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 sentence itself. This might negatively influence the accuracy in this case.

5.5.2 Original : Augmentations ratio

From the individual results of backtranslation and mixup, we conclude that these methods do not improve the hybrid approach a lot and that the adjusted version of EDA yields the best improvements. For variations, we attempt to vary the ratio of original data and adjusted data.

Table 5.6: Results of EDA adjusted ratio variations

Model	2015			2016		
	in-sample	out-of-sample	#aug	in-sample	out-of-sample	#aug
HAABSA	90.9%	77.9%	0	85.7%	83.9%	0
HAABSA adj 1:1	**95.2%**	**78.9%**	3834	**96.5%**	**84.4%**	5640
HAABSA adj 1:3	86.1%	78.1%	3834	80.7%	84.1%	5640
HAABSA adj 3:1	84.4%	77.9%	3834	91.1%	83.5%	5640

The effects of these ratios are shown in Table 5.6. The results presented in Section 5.2 are of ratio 1:1. From this analysis, it becomes clear that the 1:1 ratio performs better than the other ratios. We suspect this is because we oversample the original data to adjust the original:augmentations ratio. This means that a 3:1 ratio contains nine times the original data, resulting in twelve times as much data as in the original. This way, we suspect that within each iteration overfitting occurs, resulting in a lower accuracy than the 1:1 ratio. Furthermore, we suspect that a 1:3 ratio performs less good than the 1:1 ratio because the augmentations, in this case, weigh three times as heavy as the original data, resulting in too much pollution of the training set.

Chapter 6

Concluding remarks

This thesis focused on data augmentation in the field of aspect-based sentiment analysis (ABSA) on a sentence level. In Section 6.1, a summary of the main findings and implications of these findings are presented. In Section 6.2 some recommendations for future research are given.

6.1 Conclusion

This thesis extended the work of Wallaart and Frasincar (2019), investigating the effect of data augmentation on the accuracy of machine learning models in aspect-based sentiment analysis. Several data augmentation techniques have been adapted and extended to improve predictions of ABSA. The first method is based on the work of Wei and Zou (2019), who attempted to create a general data augmentation method applicable for all NLP classification tasks, consisting of four methods. This approach resulted in an increase of 0.3 percentage points on the SemEval dataset of 2015 and an increase of 0.2 percentage points on the SemEval 2016 dataset compared to the original two-step hybrid approach. Afterward, some adjustments are proposed on this original EDA method. This method was called adjusted EDA, and resulted in an increase of 1.0 percentage point on the SemEval 2015 dataset and an increase of 0.5 percentage points on the 2016 SemEval dataset. Afterward, the effect of backtranslation was investigated. This method translates a sentence from English to a target language and back, which frequently generates a sentence different from the original sentence. This method had varying results, sometimes increasing accuracy and sometimes decreasing the accuracy. Translating to Japanse and back had the best results, though it did not increase accuracy in the SemEval 2015 dataset. It did result in an improvement of 0.5 percentage points on the SemEval 2016 dataset. A similar result was found for the use of mixup, which

yielded positive results for some values of α and negative results in other cases. The best result was obtained for $\alpha = 0.2$, which resulted in no increase in the 2015 SemEval dataset and a 0.4 percentage point increase in the 2016 SemEval dataset.

We suspect that backtranslation and mixup resulted in varying results because the target had to remain fixed. This way, the augmentation techniques were performed on a part of a sentence instead of on a sentence as a whole. For backtranslation, we translated the left context to the target language and back, and we translated the right context to the target language and back, keeping the target fixed. This way, the translation engine had to translate part of a sentence. 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.

Overall, we conclude that the effects of data augmentation are relatively small on the hybrid model's accuracy for aspect-based sentiment analysis. The improvement that yielded the best results (EDA adjusted) required six times as much data to train on, which results in a proportionate increase in training time. On larger scales, these kinds of augmentation techniques are not realistic.

6.2 Future Work

This research leads to various questions for future work. First of all, there are still improvements to be made on HAABSA in general, as it cannot classify sentences with implicit targets. This improvement would also be engaging concerning data augmentation as it raises new challenges and opportunities. These implicit targets do not require to keep the aspect fixed, yielding more flexibility in the backtranslation and the mixup augmentation techniques.

Also, in the future we plan to experiment with other WSD algorithms, as there are many different algorithms focusing on WSD (Wang et al., 2019).

Furthermore, in this thesis, the data augmentation techniques were used on all sentences. It would be exciting research to analyze which kind of sentences give the highest yield when being augmented and which sentences harm the accuracy when being augmented. A pattern like augmentation in short sentences works better than augmentations in longer sentences would be a fascinating finding. This analysis can also be extended for mixup, in the sense that instead of selecting random sentences used for mixup, it is applied on sentences with similar lengths of the context vectors. This implies that the augmentation techniques do not necessarily have to be used

upon all sentences, but can be used only on some.

Bibliography

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc.

Bloem, C. (2017). *84 Percent of People Trust Online Reviews as Much as Friends. Here's How to Manage What They See.* retrieved from 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.

Cambria, E. and White, B. (2014). Jumping NLP curves: A review of natural language processing research. volume 9, pages 48–57. IEEE.

Do, H. H., Prasad, P., Maag, A., and Alsadoon, A. (2019). Deep learning for aspect-based sentiment analysis: A comparative review. *Expert Systems with Applications*, 118:272 – 299.

Fellbaum, C. (1998, ed.). *WordNet: An Electronic Lexical Database*. MIT Press.

Grimes, S. (2017). *4 AI Startups that Analyze Customer Reviews.* retrieved 5 April 2020 from https://venturebeat.com/2017/01/27/4-ai-startups-that-analyze-customer-reviews/.

Guo, H., Mao, Y., and Zhang, R. (2019). Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*.

Honnibal, M. (2013). A good part-of-speech tagger in about 200 lines of python.

Kubat, M. (1999). Neural networks: a comprehensive foundation. *Knowledge Engineering Review*, 13(4):409–412.

Lesk, M. (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*, pages 24–26. ACM.

Liu, B. (2015). *Sentiment Analysis - Mining Opinions, Sentiments, and Emotions*. Cambridge University Press.

Merity, S., Keskar, N. S., and Socher, R. (2017). Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, (ICLR 2013)*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *27th Annual Conference of Neural Information Processing Systems (NIPS, 2013)*, pages 3111–3119. Curran Associates, Inc.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Perez, L. and Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1902.09314*.

Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., and Androutsopoulos, I. (2015). Semeval-2015 task 12: Aspect based sentiment analysis. In *9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495. Association for Computational Linguistics.

Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., and Androutsopoulos, I. (2016). Semeval-2016 task 5: Aspect based sentiment analysis. In *10th International Workshop on Semantic Evaluation (SemEval 2016)*, pages 19–30. Association for Computational Linguistics.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, page 2383–2392.

Schouten, K. and Frasincar, F. (2016). Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 28(3):813–830.

Schouten, K. and Frasincar, F. (2018). Ontology-driven sentiment analysis of product and service aspects. In *15th Extended Semantic Web Conference (ESWC 2018)*. Springer.

Sennrich, R., Haddow, B., and Birch, A. (2016). Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, (ACL, 2016)*. The Association for Computer Linguistics.

Shleifer, S. (2019). Low resource text classification with ULMFiT and backtranslation. *arXiv preprint arXiv:1905.08941*.

Tan, L. (2014). Pywsd: Python implementations of word sense disambiguation (wsd) technologies [software]. https://github.com/alvations/pywsd.

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

Vasilescu, F., Langlais, P., and Lapalme, G. (2004). Evaluating variants of the Lesk approach for disambiguating words. In *Fourth International Conference on Language Resources and Evaluation (LREC, 2004)*.

Wallaart, O. and Frasincar, F. (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)*, volume 11503 of *LNCS*, pages 363–378. Springer.

Wang, Y., Wang, M., and Fujita, H. (2019). Word sense disambiguation: A comprehensive knowledge exploitation framework. *Knowledge-Based Systems*, 190.

Wei, J. W. and Zou, K. (2019). EDA: easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, (EMNLP-IJCNLP, 2019)*, pages 6381–6387. Association for Computational Linguistics.

Yan, X., Wang, J., and Chau, M. (2015). Customer revisit intention to restaurants: Evidence from online reviews. *Information System Frontiers*, 17(3):645–657.

Yu, A. W., Dohan, D., Le, Q., Luong, T., Zhao, R., and Chen, K. (2018). Fast and accurate reading comprehension by combining self-attention and convolution. In *6th International Conference on Learning Representations (ICLR, 2018)*.

Zhang, H., Cissé, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, (ICLR, 2018)*.

Appendix A: Explanation of Code

The code for data augmentation is created in one class called 'data_augmentation.py'. This class has several methods, each method doing one of the data augmentation techniques that is described in the thesis. For EDA, the code is modified from the original EDA.[1] In this method several adjustments have been made to be compatible for ABSA. All other data augmentation techniques are programmed in this class, and all data augmentation techniques require one or two sentences as an input.

The additional code is used in two different places. First of all, the file data_reader_2016 is modified, where two additional for loops are created that augment the sentence and write these augmented sentences to a separate 'augmented records' file, such that we can distinguish between the original records and the augmented records. For mixup, the data augmentation is done in LCRRotAlt, as the words have to be looked up in the word embeddings file. In the LCRRotAlt, a method called load_inputs_twitter is called from the utils file, where the data is read and prepared. In this method the augmentations are loaded, and afterwards in the main of the LCRRotAlt the mixup is applied.

Some other modifications had to be done, like changing the way the embeddings are created in loadData, an extra boolean called augment_data is added to the main code and there are several extra configurations that can be adjusted.

The code can be found on Github at https://github.com/tomasLiesting/HAABSADA

[1] https://github.com/jasonwei20/eda$_n lp$