



UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN

Tecnologie dell'Informazione

CICLO XXXIV

Text Analysis with Deep Learning and Data Augmentation

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutore:

Chiar.mo Prof. Andrea Prati

Dottorando: Akbar Karimi

Anni 2018 - 2022

To my life companion, Parinaz

Contents

I	Introduction	1
1	Context	3
1.1	Thesis Structure	4
2	Problems and Solutions	6
2.1	Adversarial Training for Aspect-Based Sentiment Analysis with BERT	7
2.2	Improving BERT Performance for Aspect-Based Sentiment Analysis	7
2.3	Toxic Spans Detection with CharacterBERT and Bag-of-Words Model	8
2.4	AEDA: An Easier Data Augmentation Technique for Text Classification	8
2.5	Aspect-Based Emotion Analysis and Multimodal Coreference: A Case Study of Customer Comments on Adidas Instagram Posts	9
II	Background and Related Work	10
3	Aspect-Based Sentiment Analysis and Deep Learning Methods	12
3.1	What Is ABSA?	12
3.1.1	Aspect Extraction (AE)	13
3.1.2	Aspect Sentiment Classification (ASC)	14
3.1.3	Aspect-Based Emotion Analysis (ABEA)	14
3.2	The Transformer	14
3.2.1	Self-attention	15
3.3	BERT Model and Its Variants	16

3.3.1	BERT Training	16
3.3.2	RoBERTa	17
3.3.3	Post-trained BERT (BERT-PT)	17
3.4	Convolutional Neural Networks	17
3.4.1	Double Embedding CNN	18
3.5	Recurrent Neural Networks	19
3.5.1	Coarse-to-Fine Task Transfer for ABSA	20
4	Data Augmentation Methods	22
4.1	EDA: Easy Data Augmentation Techniques	22
4.2	Data Augmentation Using Back-translation	23
4.3	Mixup Data Augmentation and Its Variants	23
4.3.1	Mixup Transformer	24
4.3.2	HypMix: Hyperbolic Interpolative Data Augmentation	24
4.4	Adversarial Examples	25
4.5	AEDA: An Easier Data Augmentation	26
III	Research Findings	27
5	Adversarial Training for Aspect-Based Sentiment Analysis with BERT	29
5.1	Introduction	29
5.2	Related Work	31
5.3	Aspect-Based Sentiment Analysis Tasks	32
5.4	Model	33
5.5	Experimental Setup	36
5.6	Experiments	37
5.6.1	Hyperparameters and adversarial training	37
5.6.2	Perturbation size in adversarial training	42
5.7	Conclusion	43

6	Improving BERT Performance for Aspect-Based Sentiment Analysis	44
6.1	Introduction	44
6.2	Related Work	45
6.3	Aspect-Based Sentiment Analysis Tasks	46
6.3.1	Aspect Extraction	46
6.3.2	Aspect Sentiment Classification	46
6.4	Models	47
6.4.1	Conditional Random Fields	47
6.4.2	Parallel Aggregation	48
6.4.3	Hierarchical Aggregation	49
6.5	Experiments	49
6.5.1	Datasets	50
6.5.2	Performance of BERT Layers	51
6.5.3	Increasing Training Epochs	51
6.6	Results	53
6.7	Conclusion	55
7	Toxic Spans Detection with CharacterBERT and Bag-of-Words Model	56
7.1	Introduction	56
7.2	System Description	57
7.2.1	Pre-processing	57
7.2.2	CharacterBERT	58
7.2.3	Bag-of-Words Model	59
7.2.4	Combining the Two Models	60
7.3	Experiments and Results	61
7.3.1	Performance of CharacterBERT	61
7.3.2	Analysis of the Bag-of-Words model	61
7.4	Conclusion	63
8	AEDA: An Easier Data Augmentation Technique for Text Classification	65
8.1	Introduction	65
8.2	Related Work	67

Contents	iv
8.3 AEDA Augmentation	68
8.4 Experimental Setup	68
8.4.1 Datasets	68
8.4.2 Models	69
8.5 Results	69
8.5.1 AEDA Outperforms EDA	70
8.5.2 Trend on Training Set Sizes	72
8.6 Ablation Study	72
8.6.1 Number of Augmentations	72
8.6.2 Effect of Random Initialization	72
8.6.3 Using AEDA with Deep Models	74
8.7 Discussion	74
8.8 Conclusion and Future Work	75
9 Aspect-Based Emotion Analysis and Multimodal Coreference: A Case Study of Customer Comments on Adidas Instagram Posts	76
9.1 Introduction	76
9.2 Related Work	78
9.3 Dataset	79
9.3.1 Dataset Collection	79
9.3.2 Dataset Annotation	80
9.4 Experiments & Results	84
9.5 Discussion	86
9.6 Conclusion	88
9.7 Acknowledgments	88
IV Conclusions and Future Work	89
10 Conclusions	91
11 Future Work	94

Contents	v
Publications	96
Bibliography	113
Acknowledgements	114

List of Figures

3.1	A sample Adidas product	13
3.2	Self-attention in Transformer [1]	15
3.3	The Transformer encoder and the BERT model. Figure taken from humboldt-wi.github.io	16
3.4	CNN proposed by [2] for sentence classification	18
3.5	Double embedding CNN [3]	19
3.6	RNN for text classification	19
3.7	LSTM unit. Figure taken from colah.github.io	20
3.8	MGAN model [4]	21
4.1	Mixup Transformer [5]	24
4.2	Punctuation marks used for AEDA [6]	26
5.1	BERT word embedding layer [7]	33
5.2	The proposed architecture: BERT Adversarial Training (BAT) . . .	34
5.3	Results on the impact of training epochs and dropout value in post- trained BERT for AE task	38
5.4	Comparing best results of BERT-PT and BAT with different sizes of perturbations (ϵ) for AE task	38
5.5	Results on the impact of training epochs and dropout value in post- trained BERT for ASC task	40









5.6	Comparing best results of BERT-PT and BAT with different sizes of perturbations (ϵ) for ASC task	40
6.1	An example of representing a sentence with its word labels using CRFs.	48
6.2	Parallel aggregation (P-SUM)	49
6.3	Hierarchical aggregation (H-SUM)	50
6.4	Performance of BERT layers initialized by BERT-PT weights for ASC on RST14 validation data. Each model is the BERT model using the specified number of layers. 1L means using the first layers, 2L means using the first 2 layers, etc. Accuracy values are percentages.	52
6.5	Training and validation losses of the 12-layer BERT model initialized with BERT-PT weights for AE (laptop (a) and restaurant (b)) and ASC (laptop (c) and restaurant (d)). In each figure, the upper lines are validation losses and the bottom lines are training losses, each line corresponding to a seed number.	53
7.1	The difference between the BERT and CharacterBERT models is the way they compute the initial embeddings. The former uses word-piece embeddings while the latter uses character embeddings. Figure taken from [8].	59
7.2	Performance of the Bag-of-Words model on validation set. The frequencies and ratios are the minimum thresholds that specify whether or not to consider a word toxic.	61
7.3	Heatmap of the results (F1 scores) with different values of term frequency and toxicity ratio before combining with CharacterBERT . .	64
7.4	Heatmap of the results (F1 scores) with different values of term frequency and toxicity ratio after combining with CharacterBERT . . .	64

8.1	Average performance of the generated data using our proposed augmentation method (AEDA) compared with that of the original and EDA-generated data on five text classification tasks. Using both EDA and AEDA, we added 9 augmented sentences to the original training set to train the models. For each task, we ran the models with 5 different seed numbers and took the average score.	66
8.2	Performance of the RNN model trained on various proportions of the original, EDA-generated, and AEDA-generated training data for five text classification tasks. All the scores are the average of 5 runs. . .	71
8.3	Impact of number of augmentations on the performance of the RNN model trained on various training sizes. Scores are the average of 5 runs over the five datasets. The y axis shows the percentage of improvement.	73
8.4	Average performance of EDA and AEDA over 21 different seed numbers. The results are in line with the experiments run over 5 seeds. .	73

List of Tables

4.1	EDA [9] examples	23
4.2	Examples of white-box and black-box attacks. The adversarial attacks replace the green letter (word) in the original sentence with the red letter (word). Respectively, the predictions of the original and the adversarial examples with the network’s confidence are in green and red.	25
5.1	Laptop and restaurant datasets for AE. S: Sentences; A: Aspects; Rest16: Restaurant dataset from SemEval 2016	36
5.2	Laptop and restaurant datasets for ASC. Pos, Neg, Neu: Number of positive, negative, and neutral sentiments, respectively; Rest14: Restaurant dataset from SemEval 2014	36
5.3	Comparison of best results for Aspect Extraction	41
5.4	Comparison of best results for Aspect sentiment classification. Acc: Accuracy; MF1: Macro-F1	41
5.5	Comparison of BAT results using BERT-BASE initialization	42
5.6	Comparison of BAT results using BERT-PT initialization	42
6.1	Laptop (LPT14) and restaurant (RST16) datasets from SemEval 2014 and 2016, respectively, for AE. S: Number of sentences; A: Number of aspects.	50

6.2	Laptop (LPT14) and restaurant (RST14) datasets from SemEval 2014 for ASC. S: Number of all sentences; Pos, Neg, Neu: Number of positive, negative, and neutral sentiments, respectively.	51
6.3	Comparison of the results for Aspect Extraction (AE) and Aspect Sentiment Classification (ASC). BERT-PT* is the original BERT-PT model using our model selection. The boldfaced numbers show the outperforming models using the same settings. Each score in the table is the average of 9 runs. Results for the cited papers are reported from the corresponding paper. The other models are run for 4 epochs. LPT: Laptop, RST: Restaurant, Acc: Accuracy , MF1: Macro-F1. Values are percentages.	54
7.1	Three examples from the training set	58
7.2	Top 10 toxic words in the training set	60
7.3	Comparing results of the proposed models. The boldfaced one is the submitted version. BoW: Bag-of-Words model.	62
7.4	Words selected as the toxic words with minimum frequency of 40 and minimum toxicity ratio of 0.7 (BoW (v1))	63
8.1	Examples of the augmented data using the AEDA technique	68
8.2	Statistics of the utilized datasets. N_{class} : Number of classes, L_{avg} : Sentence average length, N_{train} : Number of training samples, N_{test} : Number of test samples, $ V $: Number of unique words	69
8.3	Comparing average performance of EDA and AEDA across all datasets on different training set sizes. For each training sample, 16 augmented sentences were added. Scores are the average of 5 runs. Numbers are in percentages.	70
8.4	Comparing the impact of EDA and AEDA on the BERT model. The model was trained on the combination of the original data and one augmentation for each training sample. The scores are the average of 5 runs.	74

9.1	Aspect taxonomy and examples from the dataset for each main category	81
9.2	Examples of annotated comments.	82
9.3	Inter-annotator agreement for aspect and emotion categories (Cohen's Kappa) and emotional dimensions (Krippendorff's alpha). . .	83
9.4	Number of comments per emotion category / emotional rating.  means that the image was needed for understanding the comment,  means that the image was not needed and the text alone was sufficient.	84
9.5	Number of comments per aspect category.  means that the image was needed for understanding the comment,  means that the image was not needed and the text alone was sufficient.	85
9.6	Accuracy of aspect and emotion classification for the complete dataset (All), and for subsets of the data containing either only instances where visual information was needed for full understanding () or instances where visual information was not needed ().	86
9.7	Examples of comments that include an anaphor.  means that the image was needed for understanding the comment,  means that the image was not needed and text alone was sufficient. When the classifier made a correct prediction, it is indicated with a C; when a false prediction was made, it is indicated with an F.	88

Part I

Introduction

This part includes two chapters. In the first chapter, we provide the context for thesis topic and give reasons why it is important and valuable to analyze textual content. We also present the thesis structure at the end of Chapter 1. In Chapter 2, we give a summary of each one of the works that make up this thesis.

Chapter 1

Context

Adidas is a large company with numerous customers. More than 34 million people follow its latest products only on Instagram. As the satisfaction of these clients has a direct impact on the company's sales and consequently on its profits, it is crucial for the company managers to gain knowledge of how the consumers feel about their products. However, with millions of people talking and writing about various products, it is extremely difficult, if not impossible, to manually analyze the user opinions. Similar to Adidas, there are thousands of businesses with many customers publishing their opinions on the products and services that the companies deliver.

Apart from businesses, understanding how people perceive an event, a person, a campaign, or any other collective matter can be of great interest for those who are involved. With the numerous number of sources publishing data at a fast pace, there is a need for automatic analysis of these data. In some situations, the automatic analysis can also be used to reduce harm that might be caused by online communities. For instance, automatic models can be built to identify the toxic content and slow down the spread of misinformation.

Text analysis consists of a broad range of methods and algorithms that can be used to tackle the automatic analysis of user-generated data in the textual format. Sentiment Analysis (SA) in general, and Aspect-Based Sentiment Analysis (ABSA) in particular, is a subarea of text analysis that deals with sentiment extraction from

such raw data. In its simplest form, the user-generated sentences or comments can be classified into two classes, namely positive and negative. While this classification is fruitful to some extent, it ignores the neutral opinions and forces them to be one of the two classes. In addition, it does not provide any information on what entity the sentiments refer to. These problems are addressed by ABSA, which takes into account the neutral class. In addition, it contains another task called Aspect Extraction (AE) which deals with the extraction of the entity towards which the extracted sentiment refers to. In some cases, there can be several categories to which the a sentence may belong. For instance, we might want to identify whether a question is about a location, a human being, a value, an entity etc. The number of categories can go up to hundreds, which can be addressed by text classification.

In addition to ABSA and text classification, there are numerous other applications of text analysis that we can adopt based on our needs. In this thesis, however, we mostly deal with the two above-mentioned areas and propose some solutions that can be advantageous.

1.1 Thesis Structure

The thesis is organized in four parts:

- Part I, the current one, provides the context for this thesis. In addition, it introduces the problems we have addressed as well as our proposed solutions.
- Part II deals with the background knowledge and the main models that have been used in this work. It explains convolutional and recurrent neural networks as well as the Transformer and the BERT models. In addition, several data augmentation methods are discussed. We also review the related works along the way.
- Part III comprises the research findings as well as the proposed models for the addressed problems.

- Part IV gives a general conclusion of the thesis and discusses the future directions.

Chapter 2

Problems and Solutions

In this thesis, we tackle several different problems in text analysis. They range from dealing with sequence labeling tasks, to text classification and data augmentation. Additionally, we introduce an annotated dataset for the task of emotion analysis.

The contributions of this thesis is as follows:

- Integration of adversarial training technique with the BERT language model for Aspect-Based Sentiment Analysis (ABSA)
- Two simple modules that are added on top of the BERT model and help improve the performance of the system for ABSA
- A simple bag-of-words model for toxic language detection
- An easy to implement yet effective data augmentation method for text classification tasks
- An annotated multimodal dataset extracted from social media for Aspect-Based Emotion Analysis (ABEA)

2.1 Adversarial Training for Aspect-Based Sentiment Analysis with BERT

Aspect-Based Sentiment Analysis (ABSA) studies the extraction of sentiments and their targets. Collecting labeled data for this task in order to help neural networks generalize better can be laborious and time-consuming. As an alternative, similar data to the real-world examples can be created artificially through an adversarial process which is carried out in the embedding space. Although these examples are not real sentences, they have been shown to act as a regularization method which can make neural networks more robust. In this work, we fine-tune the general purpose BERT and domain specific post-trained BERT (BERT-PT) using adversarial training. After improving the results of post-trained BERT with different hyperparameters, we propose a novel architecture called BERT Adversarial Training (BAT) [10] to utilize adversarial training for the two major tasks of Aspect Extraction and Aspect Sentiment Classification in sentiment analysis. The proposed model outperforms the general BERT as well as the in-domain post-trained BERT in both tasks. To the best of our knowledge, this is the first study on the application of adversarial training in ABSA. The code is publicly available on a GitHub repository at <https://github.com/IMPLabUniPr/Adversarial-Training-for-ABSA>

2.2 Improving BERT Performance for Aspect-Based Sentiment Analysis

In this work, we propose two simple modules called Parallel Aggregation and Hierarchical Aggregation [11] to be utilized on top of BERT for two main ABSA tasks, namely Aspect Extraction (AE) and Aspect Sentiment Classification (ASC). With the proposed modules, we show that the intermediate layers of the BERT architecture can be utilized for the enhancement of the model performance¹.

¹<https://github.com/IMPLabUniPr/BERT-for-ABSA>

2.3 Toxic Spans Detection with CharacterBERT and Bag-of-Words Model

With the ever-increasing availability of digital information, toxic content is also on the rise. Therefore, the detection of this type of language is of paramount importance. We tackle this problem utilizing a combination of a state-of-the-art pre-trained language model (CharacterBERT) and a traditional bag-of-words technique. Since the content is full of toxic words that have not been written according to their dictionary spelling, attendance to individual characters is crucial. Therefore, we use CharacterBERT to extract features based on the word characters. It consists of a CharacterCNN module that learns character embeddings from the context. These are, then, fed into the well-known BERT architecture. The bag-of-words method [12], on the other hand, further improves upon that by making sure that some frequently used toxic words get labeled accordingly. The code is available for further research and reproduction of the results².

2.4 AEDA: An Easier Data Augmentation Technique for Text Classification

In this work, we propose **AEDA** (An Easier Data Augmentation) technique [6] to help improve the performance on text classification tasks. AEDA includes only random insertion of punctuation marks into the original text. This is an easier technique to implement for data augmentation than EDA method [9] with which we compare our results. In addition, it keeps the order of the words while changing their positions in the sentence leading to a better generalized performance. Furthermore, the deletion operation in EDA can cause loss of information which, in turn, misleads the network, whereas AEDA preserves all the input information. Following the baseline, we perform experiments on five different datasets for text classification. We show that using the AEDA-augmented data for training, the models show superior performance

²<https://github.com/IMPLabUniPr/UniParma-at-semeval-2021-task-5>

compared to using the EDA-augmented data in all five datasets. The source code is available for further study and reproduction of the results³.

2.5 Aspect-Based Emotion Analysis and Multimodal Coreference: A Case Study of Customer Comments on Adidas Instagram Posts

While aspect-based sentiment analysis of user-generated content has received a lot of attention in the past years, emotion detection at the aspect level has been relatively unexplored. Moreover, given the rise of more visual content on social media platforms, we want to meet the ever-growing share of multimodal content. In this paper, we present a multimodal dataset for Aspect-Based Emotion Analysis (ABEA). Additionally, we take the first steps in investigating the utility of multimodal coreference resolution in an ABEA framework. The presented dataset consists of 4,900 comments on 175 images and is annotated with *aspect* and *emotion* categories and the emotional dimensions of *valence* and *arousal*. Our preliminary experiments suggest that ABEA does not benefit from multimodal coreference resolution, and that *aspect* and *emotion* classification only requires textual information. However, when more specific information about the aspects is desired, image recognition could be essential.

³https://github.com/akkarimi/aeda_nlp

Part II

Background and Related Work

This part consists of two chapters which provide an explanation of the baselines and review some of the related works. In Chapter 3, we define aspect-based sentiment analysis as well as aspect-based emotion analysis and describe a deep pre-trained language model called BERT, along with its variants, that we have utilized in our experiments. In Chapter 4, we go over some of the data augmentation techniques that can be used for increasing the amount of training data.

Chapter 3

Aspect-Based Sentiment Analysis and Deep Learning Methods

In this chapter, we go over the main background topics of the thesis, namely Aspect-Based Sentiment Analysis (ABSA), Aspect-Based Emotion Analysis (ABEA), and the pre-trained language models used to carry out the experiments. To tackle ABSA and other tasks, we have made use of a pre-trained language called BERT and its variants. A key component of these models is the Transformer architecture that has paved the way for faster pre-training of the model. Therefore, in this chapter, after explaining the ABSA task and its subtasks, we give a brief explanation of the Transformer module as well as the different flavors of the BERT model.

3.1 What Is ABSA?

In general, the sentiment analysis task deals with specifying whether a sentence is positive, negative, or neutral. However, this task can extract little information from the content. Questions such as *What is the target of the extracted sentiment?* or *Which category does an entity belong to?* can not be answered by this general task. In order to address this issue, a fine-grained task called Aspect-Based Sentiment Analysis (ABSA) has been proposed [13, 14, 15]. ABSA was originally divided into four



Figure 3.1: A sample Adidas product

different subtasks, namely Aspect Extraction (AE), Aspect Sentiment Classification (ASC), Aspect Category Detection (ACD), and Aspect Category Polarity Classification (ACPC). Given that in this thesis, we have addressed the first two subtasks, we provide a description of them in the following subsections.

3.1.1 Aspect Extraction (AE)

Users might have opinions on a product in general or on some of its specific parts. For instance, here is a comment on one of the Adidas products that can be seen in Figure 3.1: *It's cool but that boost just looks kinda odd with that paint*. In this comment, we are interested in the part which is referenced by the consumer (paint) in addition to the sentiment towards the product. The AE subtask in ABSA deals with the extraction of this entity. It is worth noting that, in some cases, the aspect term can include several words all of which need to be extracted.

One approach that can be taken to address the AE task is to formulate it as a sequence labeling task. Therefore, given a sequence input with n words, $w_1, w_2, w_3, \dots, w_n$, we assign a label to each word in the training data and train models to predict the input labels in the test data. The labeling scheme follows the *BIO* convention where the words that are not aspect terms are labeled “*O*”. The aspect terms, on the other hand, can have two labels. Those in the beginning of the aspect phrases are

labeled “*B*” and the rest is assigned the label “*T*”.

3.1.2 Aspect Sentiment Classification (ASC)

Given one or multiple aspect terms, this task deals with the extraction of sentiments towards those aspect terms. This is a three-class text classification with *positive*, *negative*, and *neutral* classes. The ASC task can become more challenging if there are several aspects in a given sequence since each of those aspects might have different polarities. One simple solution which we utilize is to replicate the sequence in the training data as many times as the number of its aspects and give each replica one of the sentiments.

3.1.3 Aspect-Based Emotion Analysis (ABEA)

While being similar to ABSA, Aspect-Based Emotion Analysis (ABEA) aims at deepening its scope by analyzing emotional dimension such as *joy* and *anger* in addition to general multiclass classification of sequences. As a result, the purpose of ABEA is to extract more information regarding different feelings of the writer. Each one of the emotional dimensions can be seen as a separate classification task or all the specified emotions can be classified at once. We can notice that this deeper analysis of the content can be more informative for the interested parties. However, a problem that comes up is the need for additional annotated datasets specifically designed for this task. As one of our contributions, we have collected such a dataset and made it available for further research.

3.2 The Transformer

The Transformer model [1] was originally proposed for machine translation. As a result, it has an encoder-decoder architecture. However, for language modeling, the encoder part of the Transformer would suffice. Therefore, the BERT architecture consists of several Transformer encoders put together. The main motivation for the Transformer model was to do away with sequential models since due to their sequential

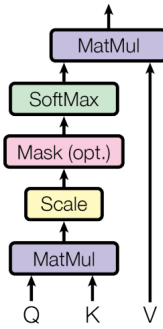


Figure 3.2: Self-attention in Transformer [1]

characteristic, they can be time-consuming to train. On the contrary, the Transformer model can be parallelized which, in turn, makes it faster to train.

3.2.1 Self-attention

Figure 3.3 (left), shows the components of the Transformer model. As we can see, it consists of a multi-head attention and a feed-forward layer as well as addition and normalization layers. The multi-head attention itself consists of multiple identical modules called self-attention which is at the core of the Transformer model. Figure 3.2 depicts this module. In mathematical terms, this module calculates Formula 3.1:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3.1)$$

where Q , K , V are called *query*, *key*, and *value* matrices, respectively, and consist of vectors with the same names coming from the input sentence. In the denominator, d_k is a normalization factor and is equal to the dimension of the query and key values. The self-attention module can be described as a mapping function that maps *queries* and *key – value* pairs to the output. From Formula 3.1, we can see that it results in a weighted sum over the values, meaning that parts of the sentence that carry more related information to the task at hand are given a higher weight. As a result, the Transformer learns more representative embeddings of the words in the vocabulary.

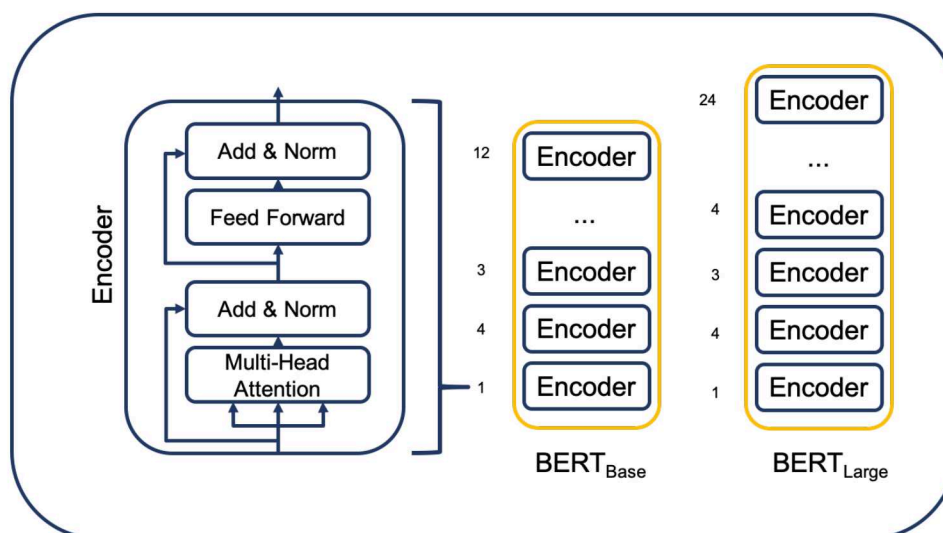


Figure 3.3: The Transformer encoder and the BERT model. Figure taken from humboldt-wi.github.io

3.3 BERT Model and Its Variants

Bidirectional Encoder Representations from Transformers (BERT) [7] is a general purpose pre-trained language model that has been widely used for various tasks in NLP. Due to its strength in representing textual content, it has shown superior performance in many downstream tasks as well. Figure 3.3 (right) shows the diagram of two versions of the BERT model. The base model, $BERT_{base}$ stacks 12 Transformer encoders on top of each other while the large one, $BERT_{large}$, contains 24 encoders.

3.3.1 BERT Training

The BERT model is trained in a self-supervised manner using the Masked Language Model (MLM) paradigm. Using this method, the input sequence is fed to the network with 15 percent of it masked. Then, in the output layer, the full sequence is given as the ground-truth. The objective is to predict the masked tokens of the sequence. Going through this process, the network learns about the structure of the language

and is then able to perform downstream tasks by fine-tuning with a small amount of labeled data.

3.3.2 RoBERTa

The RoBERTa model [16] is a variant of the BERT model with an almost identical architecture. However, the hyperparameters of BERT have been carefully tuned in order to obtain a better performance. Some of these hyperparameters include training the model for a longer period of time, using longer input sequences for training, utilizing larger batch sizes, and making use of more data to train on.

3.3.3 Post-trained BERT (BERT-PT)

The pre-trained BERT model has been trained on Wikipedia articles and the Book-Corpus dataset. As a result, it has learned some general knowledge about the language. However, in order to have a superior performance when dealing with specific domains such as restaurant and laptops reviews, it needs to be trained further on some related data. Authors of [17] has post-trained this model on Amazon laptop reviews [18] and the Yelp dataset challenge¹. The result is the post-trained BERT (BERT-PT) that outperforms the general-purpose BERT model on the laptop and restaurant datasets from SemEval 2014 [13] and 2016 [15] competitions. In our models for ABSA, we utilize the post-trained parameters to initialize the network and build on that.

3.4 Convolutional Neural Networks

The utilized CNN is a simple network consisting of one layer of convolution which is one-dimensional and one pooling layer. Assuming that there are m words in a sentence and that each sentence is represented by a k -dimensional vector, the input sentence will have an $m \times k$ representation. The word vectors used for initialization have been trained by [19].

¹<https://www.yelp.com/dataset/challenge>

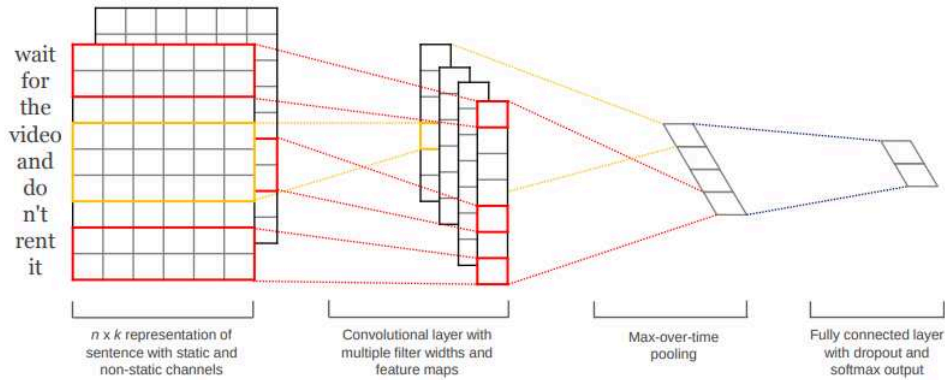


Figure 3.4: CNN proposed by [2] for sentence classification

Figure 3.4 depicts the architecture of the CNN model used for sentence classification. The convolution layer consists of several kernels with various sizes. The kernel size indicates the number of words it takes into account when processing the input sentence. In other words, a convolution of size n can capture the n -gram features. This can be particularly helpful as the information in a sentence is contained in chunks and considering words separately could cause information loss. In the utilized CNN, three kernels of size 3, 4, 5 are used. After applying the various-sized convolutions, the resulting outputs (feature maps) are concatenated. Then, a max-over-time pooling operation [20] is applied to the resulted feature maps. In the end, to acquire a probability distribution over the labels, a fully connected softmax layer is used.

3.4.1 Double Embedding CNN

An example of the CNN model used for ABSA is the Double Embedding CNN model [3]. Similar to BERT-PT, its authors use general-purpose and domain-specific embeddings in order to gain a better performance of aspect extraction task. However, instead of using the BERT model, they utilize the word vectors introduced by [21] to obtain the general word embeddings. Then, using the fastText library [22], they obtain domain-specific word embeddings from in-domain laptop and restaurant datasets.

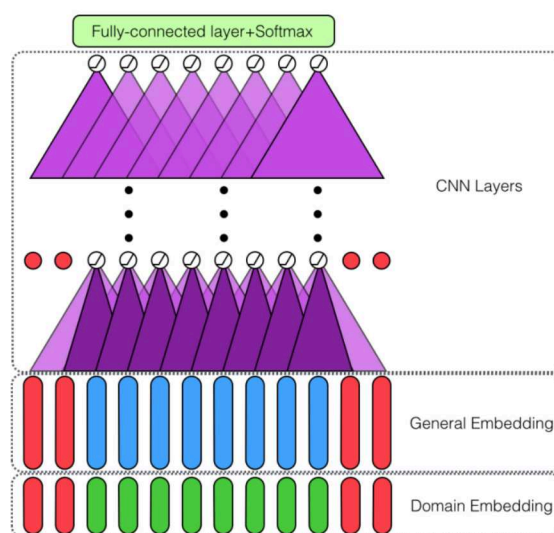


Figure 3.5: Double embedding CNN [3]

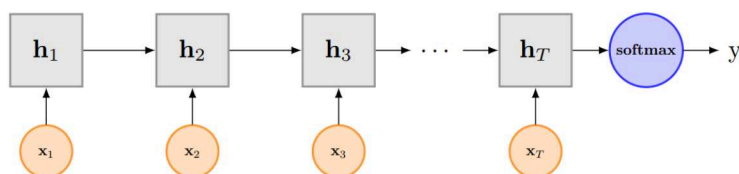


Figure 3.6: RNN for text classification

These two types of embeddings are then concatenated and fed into 4 CNN layers (Figure 3.5).

3.5 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) process the input data sequentially. Figure 3.6 shows the diagram of a simple RNN model. We have utilized a bidirectional version of this model [23]. A bidirectional RNN processes the data from right to left in addition to processing it from left to right. The results of the two directions can then be

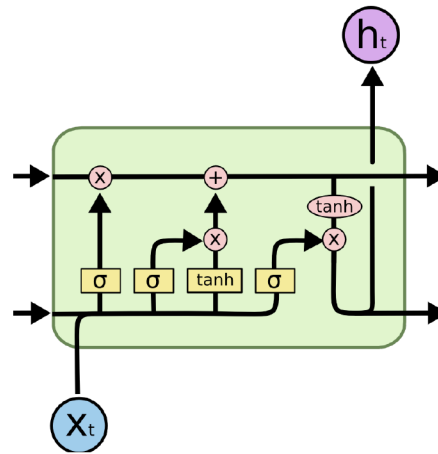


Figure 3.7: LSTM unit. Figure taken from colah.github.io

summed up or concatenated. There are various versions of RNNs that we can use to compute the hidden states of this model. Among them, Long Short-Term Memories (LSTMs) [24] are one of the widely used ones. Figure 3.7 shows the inner parts of an LSTM module, where σ is the Sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3.2)$$

The horizontal line that goes through the LSTM unit and is affected by minimal changes is called the *cell state* which helps LSTMs not to forget the previously processed information. As a result, it is able to capture the long-term dependencies in a given sequence. The other parts either control how much of the input information should be preserved and passed on to the next part or how the cell state should be updated.

3.5.1 Coarse-to-Fine Task Transfer for ABSA

The coarse-to-fine task transfer model (called MGAN [4]) makes use of RNNs for ABSA. It utilizes one task, Aspect Category (AC) classification, with an abundant amount of training data to improve another task which is aspect extraction and has

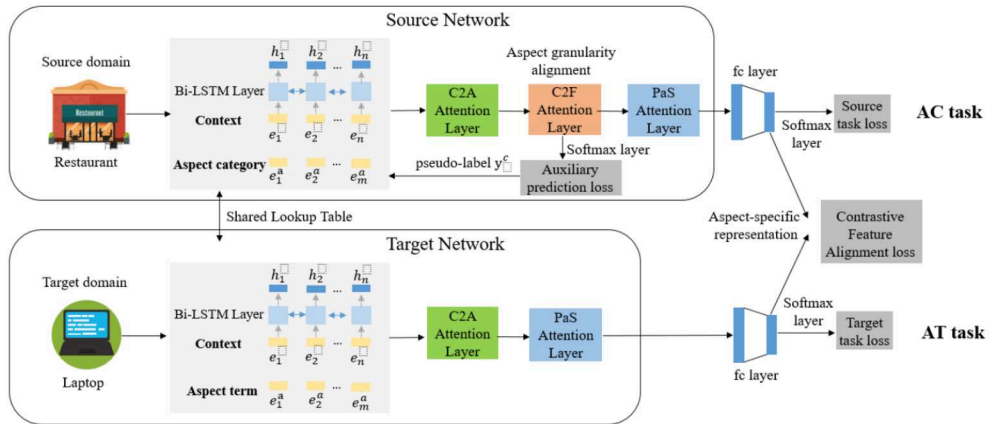


Figure 3.8: MGAN model [4]

only a small amount of data for training. As we can see in Figure 3.8, the architecture consists of two parallel parts, one for AC classification and the other for aspect extraction. The authors use an attention layer and an auxiliary task to leverage the aspect category representation. This is carried out by using the attention layer to predict pseudo-labels for the same aspect categories it receives as an input, making it act similarly to an autoencoder. This can help the attention layer attend to the parts of the input sequence where the aspect terms could be found.

Chapter 4

Data Augmentation Methods

In this chapter, we review some of the data augmentation techniques, including our proposed method called AEDA.

Data augmentation has long been practiced to increase the amount of training data in many areas of machine learning such as computer vision and NLP [25, 26, 27]. It refers to the manipulation of the training data in order to create new samples that are similar to the real ones with only a small difference. The goal is to introduce distorted samples so that the network can become more robust to a variety of changes in the input sequence. As a result, data augmentation is expected to improve generalization and the performance on the unseen data. While data augmentation can help the model, the cost of its process needs to be taken into account. Some methods require large models whereas others need extra resources.

4.1 EDA: Easy Data Augmentation Techniques

The EDA method [9] is one of the easiest data augmentation techniques that we can use. Given the input sequence, it augments it using four simple operations, namely Synonym Replacement (SR), Random Insertion (RI), Random Deletion (RD), and Random Swap (RS). Some of the examples created by this method can be seen in Table 4.1.

Operation	Sentence
None	A sad, superior human comedy played out on the back roads of life.
SR	A lamentable , superior human comedy played out on the backward road of life.
RI	A sad, superior human comedy played out on funniness the back roads of life.
RS	A sad, superior human comedy played out on roads back the of life.
RD	A sad, superior human out on the roads of life.

Table 4.1: EDA [9] examples

4.2 Data Augmentation Using Back-translation

Back-translation [28] is a data augmentation method that leverages machine translation systems. In this method, we can translate the training data into one or several other languages and then back-translate them to the source language. As a consequence, this method not only might substitute words with their synonyms but also it can change the grammatical structure of the input sequence. Another advantage of this technique is that we have many options to choose from as the middle language. This can mean that for a given training sample, we can have several augmented samples. However, the disadvantage of back-translation is that it requires a machine translation system. In addition, the quality of the translated sentences depends on the quality of the translation system which can be poor for low-resource languages.

4.3 Mixup Data Augmentation and Its Variants

The mixup data augmentation method [29] takes two random examples from the training set in order to create a linear and convex combination of them. With this method, both input and its label go through the same transformation. The input is a raw vector and the label is a one-hot vector. If we consider x_i, x_j to be raw input

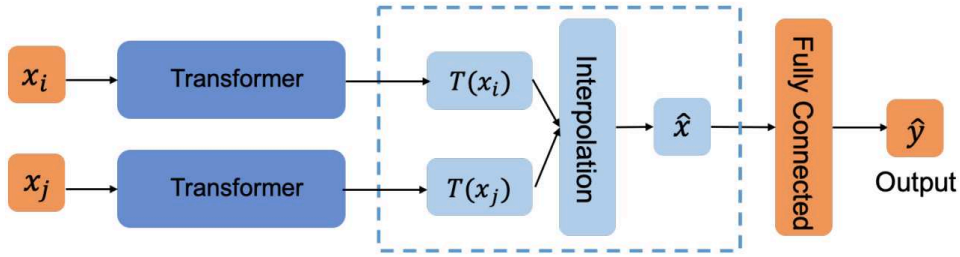


Figure 4.1: Mixup Transformer [5]

vectors and y_i, y_j their respective labels, then the new data point is created by:

$$\begin{aligned}\tilde{x} &= \lambda x_i + (1 - \lambda)x_j \\ \tilde{y} &= \lambda y_i + (1 - \lambda)y_j\end{aligned}\tag{4.1}$$

where $\lambda \in [0, 1]$. It was shown that this method can improve the performance as well as the robustness of the network. Based on this technique, several other variants have been proposed, among which mixup Transformer and HypMix can be mentioned.

4.3.1 Mixup Transformer

While the original mixup method uses raw input vectors to create new samples, Mixup Transformer uses the Transformer encoder to obtain vector representations of a pair of sentences. These representations are then linearly interpolated to create a mixed representation. Correspondingly, their labels are also interpolated. Figure 4.1 shows a diagram of this method. While this method shows improvement over the $BERT_{base}$ model, it is still expensive to implement since it requires the Transformer encoder.

4.3.2 HypMix: Hyperbolic Interpolative Data Augmentation

The HypMix method [30] introduces a similar interpolation technique as the mixup method with the difference that it uses hyperbolic space instead of the Euclidean

Chancellor Gordon Brown has sought to quell speculation over who should run the Labour Party and turned the attack on the op(B) position Conservatives. 75% World 94% Business [31]
seoul allies calm on nuclear (atomic) shock. south korea's key allies play down a shock admission its scientists experimented to enrich uranium. 74.25% Sci/Tech 86.66% World [32]

Table 4.2: Examples of white-box and black-box attacks. The adversarial attacks replace the green letter (word) in the original sentence with the red letter (word). Respectively, the predictions of the original and the adversarial examples with the network's confidence are in green and red.

space to transform the training samples. Due to its nature, HypMix can address the geometric complexities of hidden space better than its counterparts in the Euclidean space. Similarly to the mixup Transformer, its disadvantage is that a language model is required to carry out the augmentation.

4.4 Adversarial Examples

While neural networks make mistakes in their predictions, we can also make them do this deliberately. In other words, we can target the shortcomings of neural works with various methods that modify the input sequences to fool them. In computer vision, for instance, this is done by modifying the pixels of an image in a way that the changes are not visible to the human eye but the network misclassifies with a high certainty. The newly created artificial examples that fool the neural network model are called *adversarial examples* (aka *attacks*).

In text processing, there are two main categories for creating adversarial examples. One is called black-box attacks which consists of modifying the raw input sequence directly. For instance, we can randomly select some letters and change their case [31] or we can substitute words with similar words [32]. Table 4.2 shows two examples of black-box attacks. Another set of methods is called white-box attacks



Figure 4.2: Punctuation marks used for AEDA [6]

where we can access the network from inside and modify the word embeddings. For instance, FGSM [33] perturbs the original sequence by adding the gradient of the loss function to it.

While the adversarial examples are designed to mislead the network, we can consider them as augmented data and add them to the training data. This will possibly have the same effect as the other data augmentation methods, which is to make the model perform more robustly and show better generalization.

4.5 AEDA: An Easier Data Augmentation

The above-mentioned augmentation methods require an extra resource in order to be implemented. For instance, while EDA is simple and pretty straightforward to implement, it still needs a dictionary of synonyms. Similarly, the mixup methods need a language model to produce the embeddings as the inputs to the augmentation method. To further simplify data augmentation, we propose AEDA which does not require any resources, be it another language model or additional resources such as dictionaries while being effective in the generalization of the network. It only includes random insertion of punctuation marks which can be seen in Figure 4.2. The insertion of these marks changes the position of the words in a sequence forcing the network not to memorize the training data. Therefore, it improves the model performance especially where there is a small amount of data.

Part III

Research Findings

This part consists of five chapters. In Chapter 5, we explain our proposed model called BAT that trains the BERT model in adversarial manner to address the Aspect-Based Sentiment Analysis (ABSA) task. In Chapter 6, we give a description of the two modules that are added on top of BERT to tackle the ABSA task. In Chapter 7, we describe our bag-of-words model combined with CharacterBERT for toxic language detection. In Chapter 8, the explanation of our novel data augmentation method called AEDA is given. Finally, in Chapter 9, a description of our annotated dataset for aspect-based emotion analysis is provided.

Chapter 5

Adversarial Training for Aspect-Based Sentiment Analysis with BERT

5.1 Introduction

Understanding what people are talking about and how they feel about it is valuable especially for industries which need to know the customers' opinions on their products. Aspect-Based Sentiment Analysis (ABSA) is a branch of sentiment analysis which deals with extracting the opinion targets (aspects) as well as the sentiment expressed towards them. For instance, in the sentence *the spaghetti was out of this world*, a positive sentiment is mentioned towards the target which is *spaghetti*. Performing these tasks requires a deep understanding of the language. Traditional machine learning methods such as Support Vector Machines (SVM) [34], Naive Bayes [35], Decision Trees [36], Maximum Entropy [37] have long been practiced to acquire such knowledge. However, in recent years due to the abundance of available data and computational power, deep learning methods such as Convolutional Neural Nets (CNNs) [38, 2, 39], Recurrent Neural Networks (RNNs) [40, 41, 42], and the Transformer [1] have outperformed the traditional machine learning techniques

in various tasks of sentiment analysis. Bidirectional Encoder Representations from Transformers (BERT) [7] is a deep and powerful language model which uses the encoder of the Transformer in a self-supervised manner to learn the language model. It has been shown to result in state-of-the-art performances on the GLUE benchmark [43] including text classification. In [17], it is shown that adding domain-specific information to this model can enhance its performance in ABSA. Using their post-trained BERT (BERT-PT), we add adversarial examples to further improve BERT’s performance on Aspect Extraction (AE) and Aspect Sentiment Classification (ASC) which are two major tasks in ABSA. A brief overview of these two sub-tasks is given in Section 5.3.

Adversarial examples are a way of fooling a neural network to behave incorrectly [44]. They are created by applying small perturbations to the original inputs. In the case of images, the perturbations can be invisible to human eye, but can cause neural networks to output a completely different response from the true one. Since neural nets make mistakes on these examples, introducing them to the network during the training can improve their performance. This is called *adversarial training* which acts as a regularizer to help the network generalize better [33]. Due to the discrete nature of text, it is not feasible to produce perturbed examples from the original inputs. As a workaround, authors of [45] apply this technique to the word embedding space for text classification. Inspired by them and building on the work of [17], we experiment with adversarial training for ABSA.

Our main contribution in this work is the proposal of a novel architecture to apply adversarial training in the fine-tuning process of BERT language model for aspect extraction and aspect sentiment classification tasks in sentiment analysis. Our experiments show that the proposed model outperforms the performances of general BERT as well as domain specific BERT-PT. As a minor contribution we demonstrate that the number of training epochs and dropout values can have significant impacts on the model’s performance.

5.2 Related Work

Since the early works on ABSA [46, 47, 48], several methods have been put forward to address the problem. In this section, we review some of the works which have utilized deep learning techniques.

In [49], the authors design a seven-layer CNN architecture and make use of both part of speech tagging and word embeddings as features. In [3], convolutional neural networks and domain-specific data are utilized for AE and ASC. They show that adding the word embeddings produced from the domain-specific data to the general purpose embeddings semantically enriches them regarding the task at hand. A recent work shows that using in-domain data can enhance the performance of the state-of-the-art language model (BERT) [17]. Similarly, BERT is fine-tuned by [50] on domain-specific data for ASC. They perform a two-stage process, first of which is self-supervised in-domain fine-tuning, followed by supervised task-specific fine-tuning. Working on the same task, authors of [51] apply graph convolutional networks taking into consideration the assumption that in sentences with multiple aspects, the sentiment about one aspect can help determine the sentiment of another aspect.

Since its introduction by [52], attention mechanism has become widely popular in many natural language processing tasks including sentiment analysis. In [4], the authors design a network to transfer aspect knowledge learned from a coarse-grained network which performs aspect category sentiment classification to a fine-grained one performing aspect-level sentiment classification. This is carried out using an attention mechanism (Coarse2Fine) which contains an autoencoder that emphasizes the aspect term by learning its representation from the category embedding. Similar to the Transformer, which does away with RNNs and CNNs and use only attention for translation, in [53], an attention model is designed for ASC with the difference that they use lighter (weight-wise) multi-head attentions for context and target word modeling. Using bidirectional LSTMs [24], authors of [54] propose a model that takes into account the history of aspects with an attention block called Truncated History Attention (THA). To capture the opinion summary, they also introduce Selective Transformation Network (STN) which highlights more important information

with respect to a given aspect. In [55], aspect extraction task is approached in an unsupervised way. Functioning the same way as an autoencoder, their model has been designed to reconstruct sentence embeddings in which aspect-related words are given higher weights through attention mechanism.

While adversarial training has been utilized for sentence classification [45, 56], its effects have not been studied in ABSA. Therefore, in this work, we study the impact of applying adversarial training to the powerful BERT language model.

5.3 Aspect-Based Sentiment Analysis Tasks

In this section, we give a brief description of two major tasks in ABSA which are called Aspect Extraction (AE) and Aspect Sentiment Classification (ASC). These tasks were sub-tasks of task 4 in SemEval 2014 contest [13], and since then they have been the focus of attention in many studies.

Aspect Extraction. Given a collection of review sentences, the goal is to extract all the terms, such as *waiter*, *food*, and *price* in the case of restaurants, which point to aspects of a larger entity [13]. In order to perform this task, it is usually modeled as a sequence labeling task, where each word of the input is labeled as one of the three letters in $\{B, I, O\}$. Label *B* stands for *Beginning* of the aspect terms, *I* for *Inside* (aspect terms' continuation), and *O* for *Outside* or non-aspect terms. The reason for *Inside* label is that sometimes aspects can contain two or more words and the system has to return all of them as the aspect. In order for a sequence (s) of n words to be fed into the BERT architecture, they are represented as

$$[CLS], w_1, w_2, \dots, w_n, [SEP]$$

where the $[CLS]$ token is an indicator of the beginning of the sequence as well as its sentiment when performing sentiment classification. The $[SEP]$ token is a token to separate a sequence from the subsequent one. Finally, w_i are the words of the sequence. After they go through the BERT model, for each item of the sequence, a vector representation of the size 768, size of BERT's hidden layers, is computed.

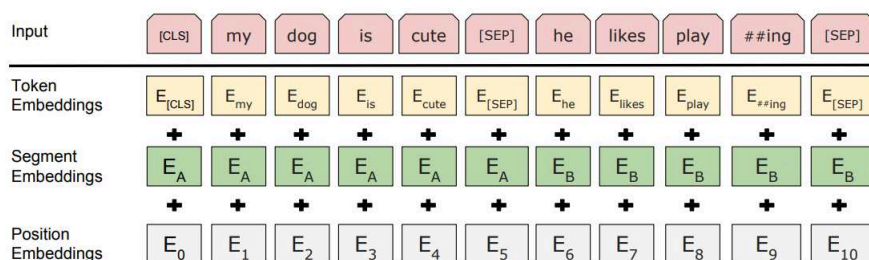


Figure 5.1: BERT word embedding layer [7]

Then, we apply a fully connected layer to classify each word vector as one of the three labels.

Aspect Sentiment Classification. Given the aspects with the review sentence, the aim in ASC is to classify the sentiment towards each aspect as *positive*, *negative*, *neutral*. In this task, the input format for the BERT model is the same as in AE. The [CLS] token in the input representation (Figure 5.1) of the BERT is where the sentiment is encoded. After the input goes through the network, in the last layer the sentiment is extracted from this token by applying a fully connected layer to its encoding.

Input sequences can have multiple aspects, meaning that a sequence can contain multiple targets with a specific sentiment polarity while BERT architecture has one element which is responsible for sentiment representation of each input. This problem is addressed in the preprocessing step when preparing the input data for the model. Sequences with multiple aspects are repeated as many times as the number of aspects they contain, each time with one of their specific aspect sentiments.

5.4 Model

Our model is depicted in Figure 5.2. As can be seen, we create adversarial examples from BERT embeddings using the gradient of the loss. Then, we feed the perturbed examples to the BERT encoder to calculate the adversarial loss. In the end, the back-

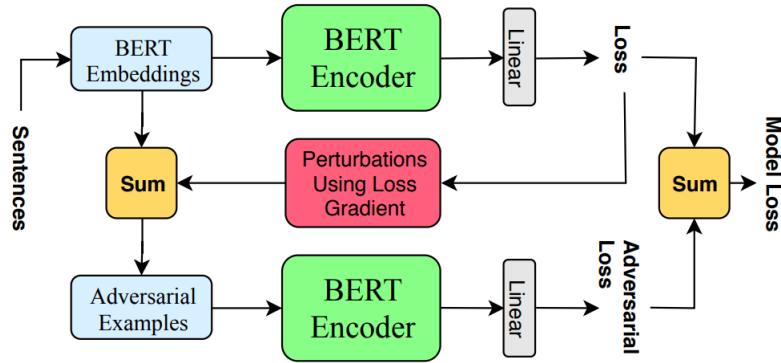


Figure 5.2: The proposed architecture: BERT Adversarial Training (BAT)

propagation algorithm is applied to the sum of both losses.

BERT Word Embedding Layer. The calculation of input embeddings in BERT is carried out using three different embeddings. As shown in Figure 5.1, it is computed by summing over token, segment, and position embeddings. Token embedding is the vector representation of each token in the vocabulary which is achieved using WordPiece embeddings [57]. Position embeddings are used to preserve the information about the position of the words in the sentence. Segment embeddings are used in order to distinguish between sentences if there is more than one (e.g. for question answering task there are two). Words belonging to one sentence are labeled the same.

BERT Encoder. BERT encoder is constructed by making use of Transformer blocks from the Transformer model. For BERT-BASE, these blocks are used in 12 layers, each of which consists of 12 multi-head attention blocks. In order to make the model aware of both previous and future contexts, BERT uses the Masked Language Model (MLM) where 15% of the input sentence is masked for prediction.

Fully Connected Layer and Loss Function. The job of the fully connected layer in the architecture is to classify the output embeddings of BERT encoder into sentiment classes. Therefore, its size is 768×3 where the first element is the hidden layers' size of BERT encoder and the second element is the number of classes. For the loss function, we use cross entropy loss implemented in Pytorch.

Adversarial Examples. Adversarial examples are created to attack a neural network to make erroneous predictions. There are two main types of adversarial attacks which are called white-box and black-box. White-box attacks [31] have access to the model parameters, while black-box attacks [58] work only on the input and output. In this work, we utilize a white-box method working on the embedding level. In order to create adversarial examples, we utilize the formula used by [45], where the perturbations are created using gradient of the loss function. Assuming $p(y|x; \theta)$ is the probability of label y given the input x and the model parameters θ , in order to find the adversarial examples the following minimization problem should be solved:

$$r_{adv} = \arg \min_{r, \|r\| \leq \epsilon} \log p(y|x+r; \hat{\theta}) \quad (5.1)$$

where r denotes the perturbations on the input and $\hat{\theta}$ is a constant copy of θ in order not to allow the gradients to propagate in the process of constructing the artificial examples. Solving the above minimization problem means that we are searching for the worst perturbations while trying to minimize the loss of the model. An approximate solution for Equation 5.1 is found by linearizing $\log p(y|x; \theta)$ around x [33]. Therefore, the following perturbations are added to the input embeddings to create new adversarial sentences in the embedding space.

$$r_{adv} = -\epsilon \frac{g}{\|g\|_2} \quad (5.2)$$

where

$$g = \nabla_x \log p(y|x; \hat{\theta}) \quad (5.3)$$

and ϵ is the size of the perturbations. In order to find values which outperform the original results, we carried out experiments with five values for epsilon whose results are presented in Figure 5.4 and discussed in Section 5.6. After the adversarial examples go through the network, their loss is calculated as follows:

$$-\log p(y|x+r_{adv}; \theta)$$

Then, this loss is added to the loss of the real examples in order to compute the model's overall loss.

Dataset	Train		Test	
	S	A	S	A
Laptop	3045	2358	800	654
Rest16	2000	1743	676	622

Table 5.1: Laptop and restaurant datasets for AE. S: Sentences; A: Aspects; Rest16: Restaurant dataset from SemEval 2016

Dataset	Train			Test		
	Pos	Neg	Neu	Pos	Neg	Neu
Laptop	987	866	460	341	128	169
Rest14	2164	805	633	728	196	196

Table 5.2: Laptop and restaurant datasets for ASC. Pos, Neg, Neu: Number of positive, negative, and neutral sentiments, respectively; Rest14: Restaurant dataset from SemEval 2014

5.5 Experimental Setup

Datasets. In order for the results to be consistent with previous works, we experimented with the benchmark datasets from SemEval 2014 task 4 [13] and SemEval 2016 task 5 [15] competitions. The laptop dataset is taken from SemEval 2014 and is used for both AE and ASC tasks. However, the restaurant dataset for AE is a SemEval 2014 dataset while for ASC is a SemEval 2016 dataset. The reason for the difference is to be consistent with BERT-PT. The summary of these datasets can be seen in Tables 5.1 and 5.2.

Implementation details. We performed all our experiments on a GPU (GeForce RTX 2070) with 8 GB of memory. Except for the code specific to our model, we adapted the codebase utilized by BERT-PT. For the Subsection A of Section 5.6, to carry out the experiments of BERT-PT model, batches of 32 were specified. However,

for our proposed model, we reduced the batch size to 16 in order for the GPU to be able to store the network. For Subsection B, we used batches of 16 for all them. For optimization, the Adam optimizer with a learning rate of $3e - 5$ was used. From SemEval’s training data, 150 examples were chosen for the validation and the remaining was used for training the model.

Implementing the creation of adversarial examples for ASC task was slightly different from doing it for AE task. During our experiments, we realized that modifying all the elements of input vectors does not improve the results. Therefore, we decided not to modify the vector for the *[CLS]* token. Since the *[CLS]* token is responsible for the class label in the output, it seems reasonable not to change it in the first place and only perform the modification on the word vectors of the input sentence. In other words, regarding the fact that the *[CLS]* token is the class label, to create an adversarial example, we should only change the words of the sentence, not the label.

Evaluation. To evaluate the performance of the model, we utilized the official script of the SemEval contest for AE. These results are reported as F1 scores. For ASC, to be consistent with BERT-PT, we utilized their script whose results are reported in Accuracy and Macro-F1 (MF1) measures. Macro-F1 is the average of F1 score for each class and it is used to deal with the issue of unbalanced classes.

5.6 Experiments

5.6.1 Hyperparameters and adversarial training

In order to carry out the experiments, first we initialize our model with post-trained BERT which has been trained using uncased version of BERT-BASE on laptop and restaurant data. We attempt to discover what number of training epochs and which dropout probability yield the best performance for BERT-PT. Since one and two training epochs result in very low scores, results of 3 to 10 training epochs have been depicted. For AE, we experiment with 10 different dropout values in the fully connected (linear) layer. The results can be seen in Figure 5.3 for laptop and restaurant datasets. To be consistent with the previous work and because of the results having high variance, each point in the figure (F1 score) is the average of 9 runs. In the end,

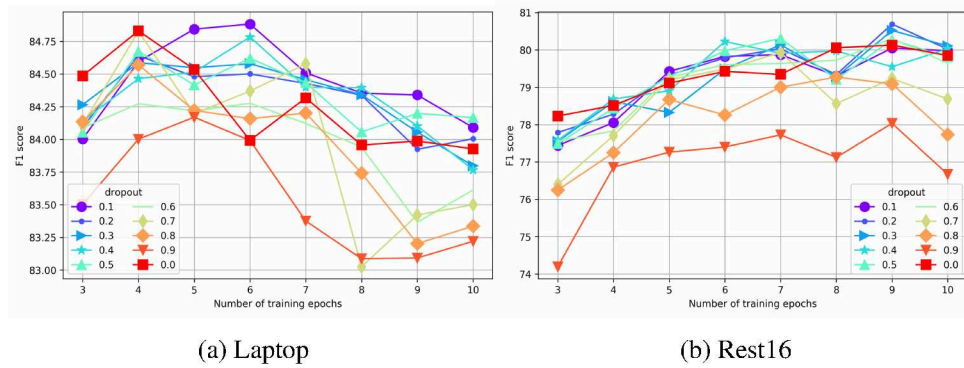


Figure 5.3: Results on the impact of training epochs and dropout value in post-trained BERT for AE task

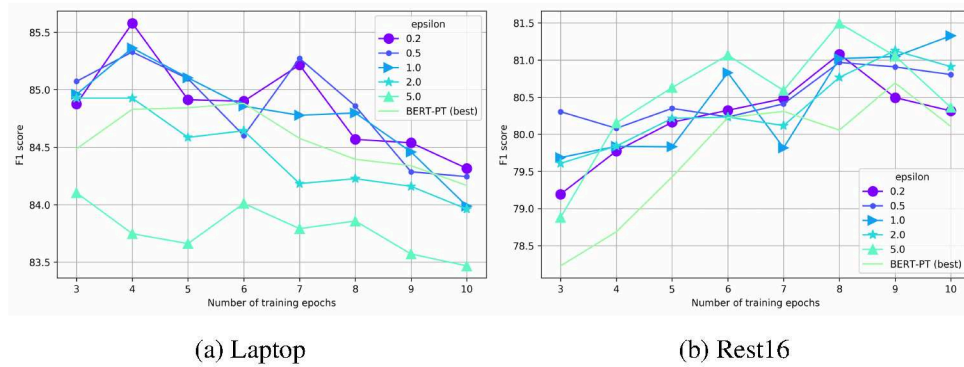


Figure 5.4: Comparing best results of BERT-PT and BAT with different sizes of perturbations (ϵ) for AE task

for each number of training epochs, a dropout value, which outperforms the other values, is found.

In our experiments, we noticed that the validation loss increases after 2 epochs as has been mentioned in the original paper. However, the test results do not follow the same pattern. Looking at the figures, it can be seen that as the number of training epochs increases, better results are produced in the restaurant domain while in the laptop domain the scores go down. This can be attributed to the selection of validation

sets as for both domains the last 150 examples of the SemEval training set were selected. Therefore, it can be said that the examples in the validation and test sets for laptop have more similar patterns than those of restaurant dataset. To be consistent with BERT-PT, we performed the same selection.

In order to compare the effect of adversarial examples on the performance of the model, we choose the best dropout for each number of epochs and experiment with five different values for epsilon (perturbation size). The results for laptop and restaurant can be seen in Figure 5.4. As is noticeable, in terms of scores, they follow the same pattern as the original ones. Although most of the epsilon values improve the results, it can be seen in Figure 5.4 that not all of them will enhance the model’s performance. In the case of $\epsilon = 5.0$ for AE, while it boosts the performance in the restaurant domain for most of the training epochs, it negatively affects the performance in the laptop domain. The reason for this could be the creation of adversarial examples which are not similar to the original ones but are labeled the same. In other words, the new examples greatly differ from the original ones but are fed to the net as being similar, leading to the network’s poorer performance.

Observing, from AE task, that higher dropouts perform poorly, we experiment with the 5 lower values for ASC task in BERT-PT experiments. In addition, for BAT experiments, two different values $\{0.01, 0.1\}$ for epsilon are tested to make them more diverse. The results are depicted in Figures 5.5 and 5.6 for BERT-PT and BAT, respectively. While in AE, towards higher number of training epochs, there is an upward trend for restaurant and a downward trend for laptop, in ASC a clear pattern is not observed. Regarding the dropout, lower values (0.1 for laptop, 0.2 for restaurant) yield the best results for BERT-PT in AE task, but in ASC a dropout probability of 0.4 results in top performance in both domains. The top performing epsilon value for both domains in ASC, as can be seen in Figure 5.6, is 5.0 which is the same as the best value for restaurant domain in AE task. This is different from the top performing $\epsilon = 0.2$ for laptop in AE task which was mentioned above.

From the experiments on BERT-PT hyperparameters, we extract the best results of BERT-PT and compare them with those of BAT. These are summarized in Tables 5.3 and 5.4 for aspect extraction and aspect sentiment classification, respectively. The

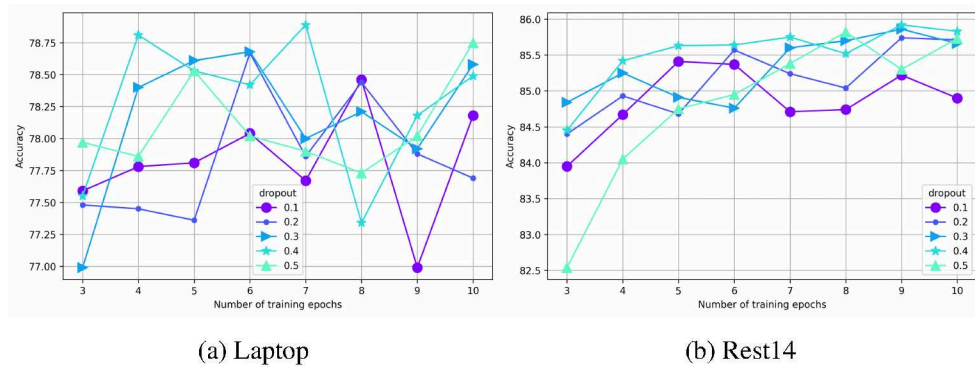


Figure 5.5: Results on the impact of training epochs and dropout value in post-trained BERT for ASC task

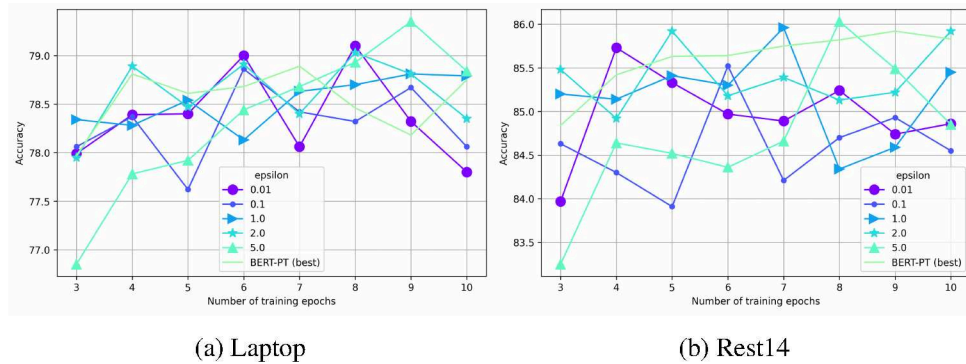


Figure 5.6: Comparing best results of BERT-PT and BAT with different sizes of perturbations (ϵ) for ASC task

base results reported are from the corresponding papers. As can be seen in Table 5.3, the best parameters for BERT-PT have greatly improved its original performance on restaurant dataset (**+2.72**) compared to laptop (**+0.62**). Similar improvements can be seen in ASC results with an increase of **+2.16** in MF1 score for restaurant compared to **+0.81** for laptop which is due to the increase in the number of training epochs for restaurant domain since it exhibits better results with more training while the model reaches its peak performance for laptop domain in earlier training epochs. In addition, applying adversarial training improves the network's performance in both tasks,

Domain	Laptop	Rest16
Methods	F1	F1
THA/STN [54]	79.52	73.61
DE-CNN [3]	81.59	74.37
BERT [17]	79.28	74.10
BERT-PT [17]	84.26	77.97
BERT-PT (best)	84.88	80.69
BAT (Ours)	85.57	81.50

Table 5.3: Comparison of best results for Aspect Extraction

Domain	Laptop		Rest14	
Methods	Acc	MF1	Acc	MF1
MGAN [54]	76.21	71.42	81.49	71.48
BERT [17]	75.29	71.91	81.54	71.94
BERT-PT [17]	78.07	75.08	84.95	76.96
BERT-PT (best)	78.89	75.89	85.92	79.12
BAT (Ours)	79.35	76.5	86.03	79.24

Table 5.4: Comparison of best results for Aspect sentiment classification. Acc: Accuracy; MF1: Macro-F1

though at different rates. While for laptop there are similar improvements in both tasks (**+0.69** in AE, **+0.61** in ASC), for restaurant we observe different enhancements (**+0.81** in AE, **+0.12** in ASC). This could be attributed to the fact that these are two different datasets whereas the laptop dataset is the same for both tasks. Furthermore, the perturbation size plays an important role in performance of the system. By choosing the appropriate ones, as was shown, better results are achieved.

Domain	Aspect Extraction (AE)		Aspect Sentiment Classification (ASC)			
	Laptop	Rest16	Laptop		Rest14	
Methods	F1	F1	Acc	MF1	Acc	MF1
BERT-BASE	81.19	75.54	75.46	71.88	80.73	70.82
BAT ($\epsilon = 0.01$)	81.93 (+0.74)	76.16 (+0.62)	75.34 (-0.12)	71.86 (-0.02)	81.50 (+0.77)	72.25 (+1.43)
BAT ($\epsilon = 0.1$)	81.33 (+0.14)	75.87 (+0.33)	75.88 (+0.54)	72.21 (+0.33)	80.77 (+0.04)	70.94 (+0.12)
BAT ($\epsilon = 1.0$)	81.55 (+0.36)	76.37 (+0.83)	74.89 (-0.45)	71.30 (-0.58)	81.20 (+0.47)	71.84 (+1.02)
BAT ($\epsilon = 2.0$)	81.58 (+0.39)	76.38 (+0.84)	75.71 (+0.25)	71.99 (+0.11)	82.27 (+1.54)	73.70 (+2.88)
BAT ($\epsilon = 5.0$)	80.34 (-0.85)	76.26 (+0.72)	74.16 (-1.3)	70.89 (-0.99)	80.34 (-0.39)	71.04 (+0.22)

Table 5.5: Comparison of BAT results using BERT-BASE initialization

Domain	Aspect Extraction (AE)		Aspect Sentiment Classification (ASC)			
	Laptop	Rest16	Laptop		Rest14	
Methods	F1	F1	Acc	MF1	Acc	MF1
BERT-PT	84.8	79.22	77.53	74.72	84.54	76.57
BAT ($\epsilon = 0.01$)	85.25 (+0.45)	80.15 (+0.93)	77.99 (+0.46)	74.96 (+0.24)	84.80 (+0.26)	77.12 (+0.55)
BAT ($\epsilon = 0.1$)	85.17 (+0.37)	79.42 (+0.2)	77.99 (+0.46)	74.98 (+0.26)	84.65 (+0.11)	76.94 (+0.37)
BAT ($\epsilon = 1.0$)	85.06 (+0.26)	79.80 (+0.58)	78.18 (+0.65)	75.31 (+0.59)	85.29 (+0.75)	77.86 (+1.29)
BAT ($\epsilon = 2.0$)	84.69 (-0.11)	79.89 (+0.67)	78.42 (+0.89)	75.32 (+0.6)	85.21 (+0.67)	77.75 (+1.18)
BAT ($\epsilon = 5.0$)	83.76 (-1.04)	80.2 (+0.98)	76.80 (-0.73)	73.72 (-1.0)	85.60 (+1.06)	78.40 (+1.83)

Table 5.6: Comparison of BAT results using BERT-PT initialization

5.6.2 Perturbation size in adversarial training

In order to discover the effect of adversarial training on the performance of the model, first we fix the number of training epochs at 4 and dropout value at 0.1 as our base configuration. Then we apply the BAT model with five perturbation sizes of $\{0.01, 0.1, 1.0, 2.0, 5.0\}$ to both BERT-BASE and BERT-PT with the same configurations. The results can be found in Tables 5.5 and 5.6, respectively. Results are reported from our implementation. As can be seen from the results, in the majority of cases for all the epsilon values, the performance over the base models improves. However, in a few cases and for the same reason which was mentioned above, it falls below the baseline. Although the BAT model in the search for the worst case adversarial examples opts for larger values leading to top performances in most cases, it

runs the risk of breaking out of the safe zone and produces examples which are not similar to (or in the vicinity of) original ones. This causes the model to sometimes perform poorly with large epsilon values. On the other hand, the smallest values in general enhance the performance of both baseline models more consistently, though with less significant amounts.

5.7 Conclusion

In this work, we introduced the application of adversarial training in Aspect-Based Sentiment Analysis. The experiments with our proposed architecture show that the performance of the general purpose BERT and in-domain post-trained BERT on aspect extraction and aspect sentiment classification tasks are improved by utilizing adversarial examples during the network training. As future work, other white-box adversarial attacks as well as black-box ones can be utilized for a comparison of adversarial training methods for various sentiment analysis tasks. Furthermore, the impact of adversarial training in the other tasks in ABSA namely Aspect Category Detection and Aspect Category Polarity could be investigated.

Chapter 6

Improving BERT Performance for Aspect-Based Sentiment Analysis

6.1 Introduction

In an industry setting, it is extremely important to have a valid conception of how consumers perceive the products. Nowadays, they communicate their perception through their comments on the products, using mostly social networks. They might have positive opinions which can lead to the success of a business or negative ones possibly leading to its demise. Due to the abundance of these views in many areas, their analysis is a time-consuming and labor-intensive task which is why a variety of machine learning techniques such as Support Vector Machines (SVM) [59, 34, 60], Maximum Entropy [61, 37], Naive Bayes [62, 35, 63], and Decision Trees [64, 36] have been proposed to perform opinion mining.

In recent years, Deep Learning (DL) techniques have been widely utilized due to the increase in computational power and the huge amount of freely available data on the Web [39, 40, 41]. One of the areas on which these techniques have had a great impact is Natural Language Processing (NLP) where modeling (i.e. understanding) the language plays a crucial role. BERT [7] is a state-of-the-art model of this kind which has become widely utilized in many NLP tasks [65, 66] as well as in other

fields [67, 68]. It has been trained on a large corpus of Wikipedia documents and books in order to *learn* the language syntax and semantics from the context. The main component of its architecture is called the Transformer [1] block consisting of attention heads. These heads have been designed to pay particular attention to parts of the input sentences that correspond to a particular given task [69]. In this work, we utilize BERT for Aspect-Based Sentiment Analysis (ABSA) tasks.

Our main contribution is the proposal of two simple modules that can help improve the performance of the BERT model. In our models we opt for Conditional Random Fields (CRFs) for the sequence labeling task which yield better results. In addition, our experiments show that training BERT for more number of epochs does not cause the model to overfit. However, after a certain number of training epochs, the learning seems to stop.

6.2 Related Work

Recently, there has been a large body of work which utilizes the BERT model for various tasks in NLP in general such as text classification [70], question answering [71], summarization [72] and, in particular, ABSA tasks [73].

Using Graph Convolutional Networks (GCNs), authors of [51] take into account sentiment dependencies in a sequence. In other words, they show that when there are multiple aspects in a sequence, the sentiment of one of them can affect that of the other one. Making use of this information can increase the performance of the model. Some studies convert the Aspect Extraction (AE) task into a sentence-pair classification task. For instance, in [74], auxiliary sentences are constructed using the aspect terms of a sequence. Then, utilizing both sequences, they fine-tune BERT on this specific task.

Word and sentence level representations of a model can also be enriched using domain-specific data. This is shown in [17] by post-training the BERT model, which they call BERT-PT, on additional restaurant and laptop data. In our experiments, we use their pre-trained model for the initialization of our models. Due to the particular architecture of the BERT model, extra modules can be attached on top of it. The

authors of [4] add different layers such as an RNN and a CRF layer to perform ABSA in an end-to-end fashion. In our work, we use the same layer modules from the BERT architecture and employ the hidden layers for prediction as well.

6.3 Aspect-Based Sentiment Analysis Tasks

Two of the main tasks in ABSA are Aspect Extraction (AE) and Aspect Sentiment Classification (ASC). While the latter deals with the semantics of a sentence as a whole, the former is concerned with finding which word that sentiment refers to. We briefly describe them in this section.

6.3.1 Aspect Extraction

In AE, the goal is to extract a specific aspect of a product towards which some type of sentiment is expressed in a review. For instance, in the sentence, “*The laptop has a good battery.*”, the word *battery* is the aspect which is extracted. Sometimes, the aspect words can be multiple in which case all of them need to be labeled accordingly. This task can be seen as a sequence labeling task, where the words are assigned a label from the set of three letters namely $\{B, I, O\}$. Each word in the sequence can be the beginning word of aspect terms (B), among the aspect terms (I), or not an aspect term (O). The classification of each word into one of these three classes, is accomplished using a fully connected layer on top of the BERT architecture and applying the Softmax function.

6.3.2 Aspect Sentiment Classification

In this task, the goal is to extract the sentiment expressed in a review by the consumer. Given a sequence, one of the three classes of *Positive*, *Negative*, and *Neutral* is extracted as the class of that sequence. The representation for this element is embodied in the architecture of the BERT model. For each sequence as input, there are two extra tokens that are used by the BERT model:

$$[CLS], w_1, w_2, \dots, w_n, [SEP]$$

where w_i are the sequence words and $[CLS]$ and $[SEP]$ tokens are concatenated to the sentence in the input stage. While the $[CLS]$ token is there to store the sentiment representation of the sentence, the $[SEP]$ token is used to separate input sequences in case there are more than one (e.g. in a question answering task). In the final layer of the architecture, a Softmax function is applied to the $[CLS]$ embedding and the class probability is computed.

6.4 Models

Deep models can capture deeper knowledge of the language as they grow. As shown by [75], the initial to middle layers of BERT can extract syntactic information, whereas the language semantics are represented in higher layers. Since extracting the sentence sentiment is semantically demanding, we expect to see this in higher layers of the network. This is the intuition behind our models where we exploit the final layers of the BERT model.

The two models that we introduce here are similar in principle, but slightly differ in implementation. Also, for the two tasks, the losses are computed differently. While for the ASC task we utilize cross-entropy loss, for the AE task, we make use of CRFs. The reason for this choice is that the AE task can be treated as sequence labeling. Therefore, taking into account the previous labels in the sequence is of high importance, which is exactly what the CRF layer does.

6.4.1 Conditional Random Fields

CRFs [76] are a type of graphical models and have been used both in computer vision (e.g. for pixel-level labeling [77]) and in NLP for sequence labeling.

Since AE can be considered a sequence labeling task, we opt for using a CRF layer in the last part of our models. The justification for the use of a CRF module for AE is that doing so helps the network to take into account the joint distribution of the labels. This can be significant since the labels of sequence words are dependent on the words that appear before them. For instance, as is seen in Figure 6.1, the occurrence of the adjective *good* can give the model a clue that the next word is

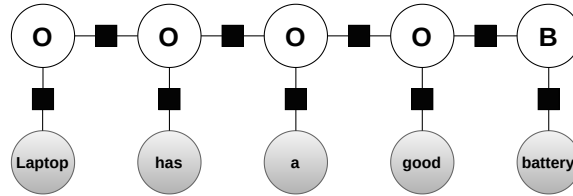


Figure 6.1: An example of representing a sentence with its word labels using CRFs.

probably not another adjective. The equation with which the joint probability of the labels is computed is as follows:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^T \exp \left\{ \sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right\} \quad (6.1)$$

In Formula 6.1, \mathbf{x} is the observed sequence, \mathbf{y} is the sequence of labels, and k and t are the indices for feature functions and time steps in the sequence, respectively. The relations between sequence words are represented by using feature functions $\{f_k\}$. These relations can be strong or weak, or non-existent at all. They are controlled by their weights $\{\theta_k\}$ which are computed during the training phase. Finally, $Z(\mathbf{x})$ is a normalization factor.

6.4.2 Parallel Aggregation

In [78], the authors show that the hidden layers of deep models can be exploited more to extract region specific information. Inspired by their work, we propose a model called P-SUM applying BERT layer modules on each one of the best performing BERT layers. Figure 6.2 shows the details of this model. We exploit the last four layers of the BERT model by adding one more BERT layer plus a fully connected layer and calculating the loss of that branch on the input data, using a Softmax function and a conditional random fields layer. The reason is that all deeper layers contain most of the related information regarding the task. Therefore, extracting this information from each one of them and combining them can produce richer representations of the semantics. In order to calculate the total loss, the loss values of all branches are summed up which is indicated with Σ notation in the diagram. This is done so, in

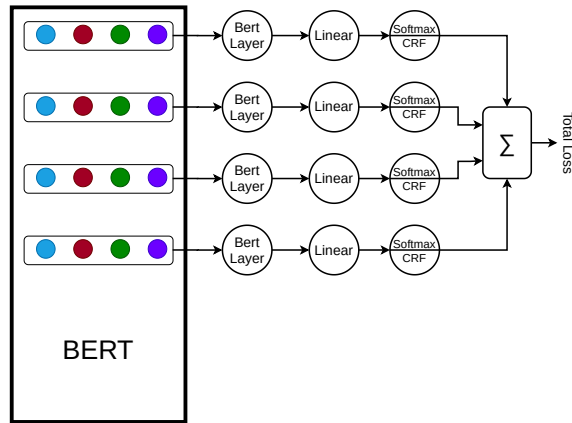


Figure 6.2: Parallel aggregation (P-SUM)

order to take all the losses into account when optimizing the parameters. However, to compute the network’s output *logits*, we average over the output *logits* of the four branches.

6.4.3 Hierarchical Aggregation

Our hierarchical aggregation (H-SUM) model is inspired by the use of Feature Pyramid Networks (FPNs) [79]. The goal is to extract more semantics from the hidden layers of the BERT model. The architecture of the H-SUM model can be seen in Figure 6.3. Here, after applying a BERT layer on each one of the hidden layers, the output is aggregated (element-wise) with the previous layer. At the same time, similar to the P-SUM, each branch produces a loss value which contributes to the total loss equally since the total loss is the summation of all of them.

6.5 Experiments

In order to carry out our experiments, we use the same codebase as [17]. We ran the experiments on a GPU (GeForce RTX 2070) with 8 GB of memory using batches of

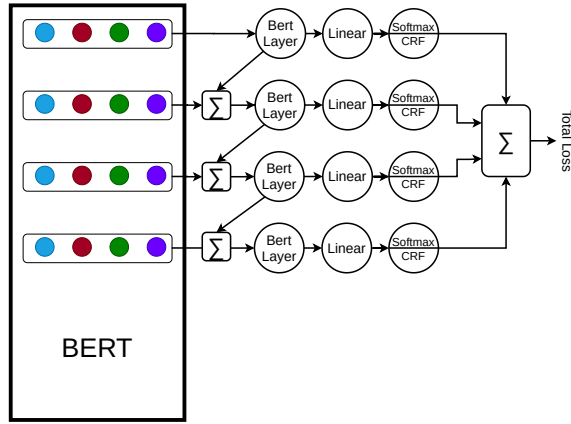


Figure 6.3: Hierarchical aggregation (H-SUM)

Dataset	Train		Test	
	S	A	S	A
LPT14	3045	2358	800	654
RST16	2000	1743	676	622

Table 6.1: Laptop (LPT14) and restaurant (RST16) datasets from SemEval 2014 and 2016, respectively, for AE. S: Number of sentences; A: Number of aspects.

16 for both our models and the BERT-PT model as the baseline. For training, Adam optimizer was used and the learning rate was set to $3e - 5$. From the distributed training data, we used 150 examples as the validation. To evaluate the models, the official scripts were used for the AE tasks and the script from the same codebase was used for the ASC task. Results are reported in F1 for AE and in Accuracy and MF1 for ASC. While F1 score is the harmonic mean of precision and recall, MF1 score is the average of F1 score for each class.

6.5.1 Datasets

In our experiments, we utilized laptop and restaurant datasets from SemEval 2014 [80] Subtask 2 and 2016 [15] Subtask 1. The collections consist of user reviews

Dataset	Train				Test			
	S	Pos	Neg	Neu	S	Pos	Neg	Neu
LPT14	2313	987	866	460	638	341	128	169
RST14	3102	2164	805	633	1120	728	196	196

Table 6.2: Laptop (LPT14) and restaurant (RST14) datasets from SemEval 2014 for ASC. S: Number of all sentences; Pos, Neg, Neu: Number of positive, negative, and neutral sentiments, respectively.

which have been annotated manually. Tables 6.1 and 6.2 show the statistics of these datasets. In choosing the datasets, we opted for the ones utilized in previous works [10, 17] so that we can draw a reliable comparison between the performance of our models and those ones.

6.5.2 Performance of BERT Layers

Depending on the depth of the network, it can perform differently. Therefore, we carried out experiments to find out how each layer of the BERT model performs. The results are shown in Figure 6.4. As can be seen, better performance is achieved in the deeper layers, especially the last four. Therefore, our modules operate on these four layers to achieve an improved model.

6.5.3 Increasing Training Epochs

More training can lead to a better performance of the network. However, one risks the peril of overfitting especially when the number of training examples are not considered to be large compared to the number of parameters contained in the model. However, in the case of BERT, as was also observed by [4], it seems that with more training the model does not overfit although the number of the training data points is relatively small. The reason behind this could be the fact that we are using an already pre-trained model which has seen an enormous amount of data (Wikipedia and Books Corpus). Therefore, we can expect that by performing more training, the model will

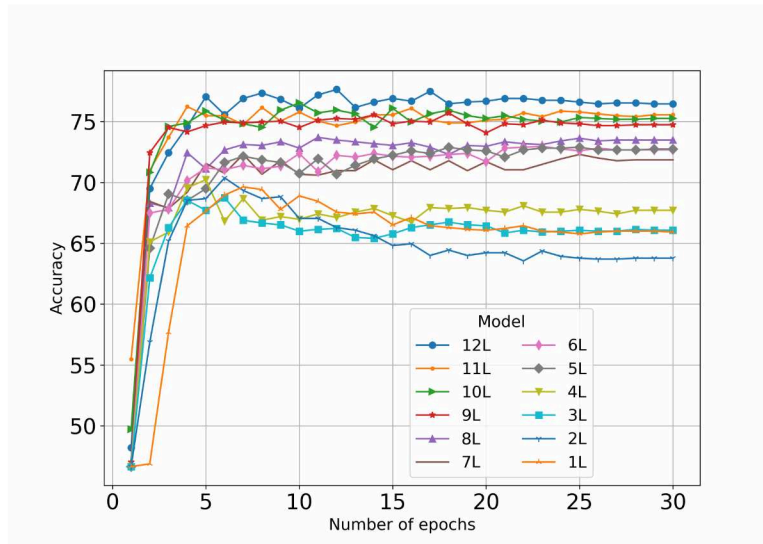


Figure 6.4: Performance of BERT layers initialized by BERT-PT weights for ASC on RST14 validation data. Each model is the BERT model using the specified number of layers. 1L means using the first layers, 2L means using the first 2 layers, etc. Accuracy values are percentages.

still be able to generalize.

The same observation can be made by looking at the validation losses in Figure 6.5. In case of an overfit, we would expect the losses to go up and the performance to go down. However, we see that with the increase in loss after the second epoch, the performance still improves for a couple of epochs and then fluctuates in the subsequent ones (Figure 6.4). This suggests that with more training, the network weights continue to change until they remain almost stable in later epochs, indicating that there is no more learning. From Figure 6.4, we see that with 4 or 5 training epochs we get near the maximum performance. Although some later epochs such as 12 yield better results for the 12-layer version, it can be considered negligible.

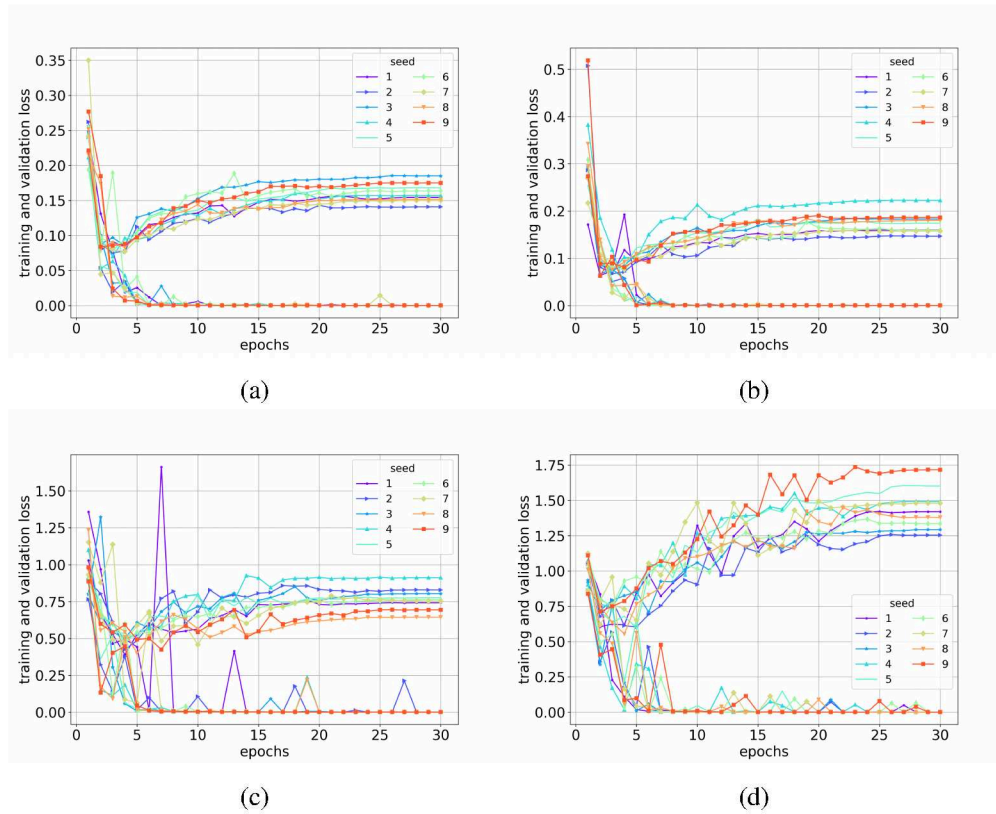


Figure 6.5: Training and validation losses of the 12-layer BERT model initialized with BERT-PT weights for AE (laptop (a) and restaurant (b)) and ASC (laptop (c) and restaurant (d)). In each figure, the upper lines are validation losses and the bottom lines are training losses, each line corresponding to a seed number.

6.6 Results

Our experimental results show that with the increase of the training epochs the BERT model also improves. These results can be seen in Table 6.3. To compare our proposed models with [17], we perform the same model selection for both of them. Unlike [17] and [10] who select their best models based on the lowest validation loss, we choose the models trained with four epochs after observing that accuracy goes up on the

Models	AE		ASC			
	LPT14	RST16	LPT14		RST14	
	F1	F1	Acc	MF1	Acc	MF1
BERT	79.28	74.10	75.29	71.91	81.54	71.94
DE-CNN [3]	81.59	74.37	-	-	-	-
BERT-PT [17]	84.26	77.97	78.07	75.08	84.95	76.96
BAT [10]	85.57	81.50	79.35	76.50	86.03	79.24
BERT-PT*	85.57	81.57	78.21	75.03	85.43	77.68
P-SUM	85.94	81.99	79.55	76.81	86.30	79.68
H-SUM	86.09	82.34	79.40	76.52	86.37	79.67

Table 6.3: Comparison of the results for Aspect Extraction (AE) and Aspect Sentiment Classification (ASC). BERT-PT* is the original BERT-PT model using our model selection. The boldfaced numbers show the outperforming models using the same settings. Each score in the table is the average of 9 runs. Results for the cited papers are reported from the corresponding paper. The other models are run for 4 epochs. LPT: Laptop, RST: Restaurant, Acc: Accuracy, MF1: Macro-F1. Values are percentages.

validation sets (Figure 6.4). Therefore, in Table 6.3, we report the original BERT-PT scores as well as the ones for our model selection. From Table 6.3, it can also be seen that the proposed models outperform the newly selected BERT-PT model in both datasets and tasks with improvements in MF1 score as high as **+1.78** and **+2** for ASC on laptop and restaurant, respectively.

It is also worth noting that, in terms of accuracy, the H-SUM module performs better than the P-SUM module in most cases. This could be attributed to the hierarchical structure of the module and the fact that each branch of this module benefits from the information processed in the preceding branch.

6.7 Conclusion

We proposed two simple modules utilizing the hidden layers of the BERT language model to produce deeper semantic representations of input sequences. The layers are once aggregated in a parallel fashion and once hierarchically. For each branch of the architecture built on top of the selected hidden layers, we compute the loss separately. These losses are then aggregated to produce the final loss of the model. We address aspect extraction using conditional random fields which helps to take into account the joint distribution of the sequence labels to achieve more accurate predictions. Our experiments show that the proposed approaches outperform the post-trained vanilla BERT model.

Chapter 7

Toxic Spans Detection with CharacterBERT and Bag-of-Words Model

7.1 Introduction

The user generated digital content is increasing rapidly every second of the day. This can include some toxic language whose detection can be difficult due to the complexities of human languages. We address this problem by participating in SemEval Workshop 2021 Task 5 [81].

In many cases, the data, which are considered to be toxic, contain words that have not been written in their standard forms. There might also be a lot of misspelling or letter replacements. In addition, usually the words that are considered to be the most offensive are bleeped which makes them difficult to be recognized if we use a model which learns the content representation based on the words. Apart from word related issues, the context also plays a crucial role in the meaning that a word conveys since words in different contexts can have various meanings.

Therefore, in order to cope with these issues, we opt for a recently pre-trained language model which has been trained on character level. CharacterBERT [8] is a

deep neural network model that has been pre-trained on Wikipedia and OpenWebText [82] corpora using the BERT architecture [7] with an addition of a character-aware Convolutional Neural Network (CNN) [83, 84]. BERT-based models have now become pervasive in many different natural language processing tasks such as reading comprehension [17], named entity recognition [85], sentiment analysis [11], and language understanding [86] as well as similar ones to toxic language detection such as propaganda detection [87]. While the BERT model is beneficial in extracting the contextual information from the text on the word level, the character-aware CNN attends to the individual letters which helps in dealing with out-of-vocabulary, unknown and rare words.

In addition to using a deep language model for detecting toxic language, we employ a very simple Bag-of-Words model that can achieve a close performance to that of the deep model. By building a dictionary of toxic words from the training data and by taking into account their frequency and ratio of toxicity, we come up with a simple model that performs as closely as about 2 percent difference in performance to the deep model’s result. Moreover, we improve the results of CharacterBERT by combining it with the output of a version of the Bag-of-Words model.

7.2 System Description

Our system consists of four main stages namely, pre-processing, applying CharacterBERT, applying Bag-of-Words model, and finally combining the results of the two models. We describe each of these stages in the following subsections.

7.2.1 Pre-processing

The training dataset consists of rows of various lengths and an array of character spans indicating their toxic parts. Each row can contain several sentences. Table 7.1 shows three examples of the training data.

We approach the task of toxic spans detection as a sequence labelling task where each word of the input row is classified into one of the predefined classes. We define three classes of {B, I, O}, meaning that each word can be the first word (B) of a set of

spans	text
[8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39]	Another violent and aggressive immigrant killing a innocent and intelligent US Citizen.... Sarcasm
[0, 1, 2, 3]	Damn, a whole family. Sad indeed.
[7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]	What a knucklehead. How can anyone not know this would be offensive??

Table 7.1: Three examples from the training set

continuous toxic words, in between (I), or not toxic (O). Therefore, in order for both of our models to be able to process these inputs, we first need to break the rows into words and label them as one of the above-mentioned classes. This was carried out by simply splitting each input row at the *space* characters. Then, after creating a dataset that has been labeled on the word level, we can use it as the input of our models. The same is done for the Bag-of-Words model with a difference in treating the bleeped words which is described in Subsection 7.2.3.

7.2.2 CharacterBERT

CharacterBERT model is almost identical to the well-known BERT model with a difference in initial embedding. In the general BERT model, words are broken into pieces and the embeddings for these word pieces are computed. In CharacterBERT, however, words are divided into letters or characters. Then, using CNN modules the embeddings are computed on the character level (Figure 7.1). This makes the network extract features on the lowest level, making it suitable for contexts which contain many unseen vocabulary terms such as misspelled words or technical jargon. After the initial character-aware CNN layer, there is the BERT_{base} architecture which contains 12 layers (blocks) of Transformer [1] with the hidden size of 768 and 12 attention heads. The final layer representations are converted into logits using a fully connected layer after which a Softmax layer is applied to extract the token's (word's) class.

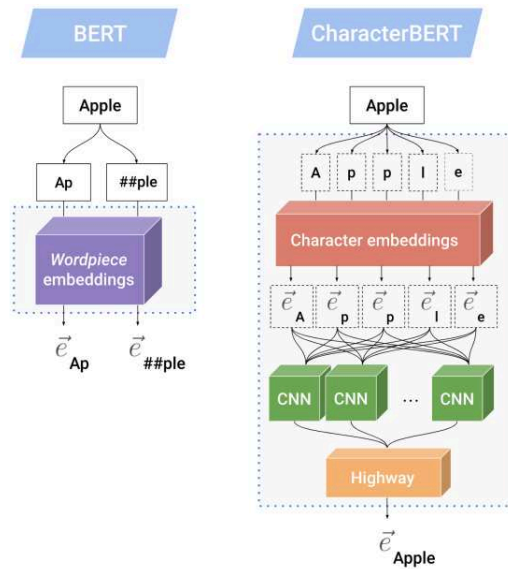


Figure 7.1: The difference between the BERT and CharacterBERT models is the way they compute the initial embeddings. The former uses word-piece embeddings while the latter uses character embeddings. Figure taken from [8].

7.2.3 Bag-of-Words Model

This model is a simple script of fewer than 80 lines of code. However, its performance on the Toxic Spans Detection task can get very close to the CharacterBERT model which has millions of parameters. In this model, by examining the training set, we first build a dictionary of toxic words with their frequency. Table 7.2 presents the top ten words of this dictionary in terms of frequency.

Then, we locate the words from the toxic dictionary in each sentence of the test set. If the word is found and its frequency as well as its toxicity ratio in the training set are higher than certain values, it is labeled as toxic. This ratio which we call *toxicity ratio* (Formula 7.1) along with the *term frequency* are the only parameters of the Bag-of-Words model.

Word	Frequency
stupid	973
idiot	557
idiots	353
stupidity	223
ignorant	190
dumb	157
moron	147
fool	141
pathetic	138
crap	121

Table 7.2: Top 10 toxic words in the training set

$$\text{toxicity ratio} = \frac{\text{labeled as toxic frequency}}{\text{total frequency}} \quad (7.1)$$

The test dataset also contains words that are bleeped. Since these words can be considered toxic with a high certainty (otherwise they would not be bleeped), we extract them separately from the test set and label them directly as toxic.

7.2.4 Combining the Two Models

In order to get the improved version of the toxic language labeling, the union of the spans detected by the bag-of-words model and that of CharacterBERT is taken. The results will improve if there are words labeled correctly with the Bag-of-Words model that are not in the output for CharacterBERT. This can be achieved by specifying a high toxicity ratio for a word to be labeled as toxic. Also, the wrongly labeled tokens should not be too many since it can have a negative effect on the F1 score. Therefore, the frequency with which a toxic word appears should be somewhat high. Striking a balance between these two parameters can help improve the output of CharacterBERT.

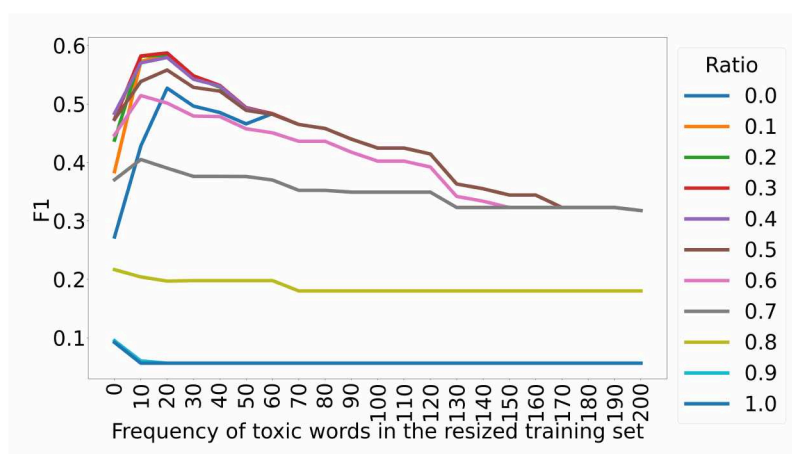


Figure 7.2: Performance of the Bag-of-Words model on validation set. The frequencies and ratios are the minimum thresholds that specify whether or not to consider a word toxic.

7.3 Experiments and Results

7.3.1 Performance of CharacterBERT

We ran the experiments for the general domain CharacterBERT with its default setting on a GPU (GeForce RTX 2070) which had 8GB of memory. We specified batch sizes of 4 for both training and testing and fine-tuned it on the toxic data only for one epoch which produces an F1 score of 65.13. It is worth noting that more training did not improve the performance.

7.3.2 Analysis of the Bag-of-Words model

In order to experiment with the Bag-of-Words model, we divide the original training set into a resized training set with 7000 sentences and a validation set with 939 sentences which were taken from the end of the original training set. Then, we find the best parameters on the validation set and using those parameters on the test data, we get a performance of almost 63 percent which is only 2 percent smaller than our

Model	F1
CharacterBERT	65.13
BoW (v1)	51.75
BoW with best parameters (v2)	62.79
CharacterBERT + BoW (v2)	65.87
CharacterBERT + BoW (v1)	66.72

Table 7.3: Comparing results of the proposed models. The boldfaced one is the submitted version. BoW: Bag-of-Words model.

deep model. Figure 7.2 shows this model’s performance for its two parameters on the validation set. One parameter represents the *minimum* frequency with which a toxic word appears in the resized training set and the other one is its *minimum* toxicity ratio in the resized training data.

We can see from Figure 7.2 that the best results are achieved when the minimum frequency is 20 and the minimum ratio is 0.3 or 0.4. Since a larger ratio can be a sign of more toxicity, we choose 0.4 as the ratio and a frequency of 20 as the thresholds with which we apply the model on the test set. This gives an F1 score of 62.79 percent (Table 7.3) which is not that much below the result of the deep model.

We can also see from Table 7.3 that although combining the output of the Bag-of-Words model with that of CharacterBERT improves the results a little bit, it is still not as significant as the first version. In the first version of the Bag-of-Words model, which was found during our primary experiments, the minimum word frequency is 40 and the minimum toxicity ratio is 0.7. With these parameters, only 10 words are selected from the training set. The frequency and toxicity ratio of these words can be seen in Table 7.4.

Although the performance of this version is a lot lower than the second version (v2) of the BoW model, it helps to improve the performance of CharacterBERT. The reason for this behavior can be attributed to the fact that models with higher thresholds both in terms of frequency and toxicity ratio tend to output more certain results, albeit fewer words than the ones that should be labeled as toxic. Therefore,

Word	Frequency	Toxicity Ratio
stupid	973	0.78
idiot	557	0.84
idiots	353	0.81
stupidity	223	0.77
moron	147	0.71
idiotic	98	0.74
hypocrite	75	0.88
shit	56	0.72
scum	52	0.70
hypocrites	44	0.76

Table 7.4: Words selected as the toxic words with minimum frequency of 40 and minimum toxicity ratio of 0.7 (BoW (v1))

many toxic words that are less probable are not extracted and F1 score drops.

Looking at Figure 7.3, we can see that, indeed, the best parameters from the experiments on the validation set (ratios 0.3 and 0.4 with frequencies 10 and 20) yield some of the best results on the test set. However, when these results are combined with the output of the CharacterBERT, we see that the higher the toxicity ratio the better the results (Figure 7.4) until 0.7 which gives the maximum improvement. The 0.8 ratio makes the predictions still a little better but 0.9 does not affect them since the words that are labeled as toxic with this certainty have most probably been found also by CharacterBERT.

7.4 Conclusion

We described the system we utilized to detect toxic language. In our approach, we first fine-tune CharacterBERT, a character-level pre-trained language model, on the toxic training data. Then using a simple bag-of-words model, we further improve the results of this system. The Bag-of-Words model labels the words based on their

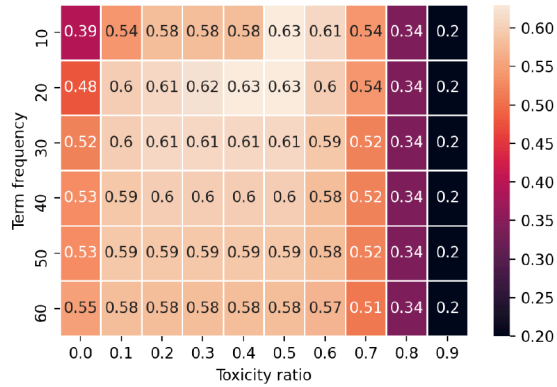


Figure 7.3: Heatmap of the results (F1 scores) with different values of term frequency and toxicity ratio before combining with CharacterBERT

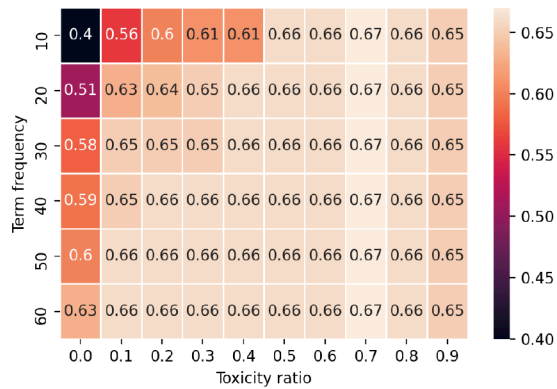


Figure 7.4: Heatmap of the results (F1 scores) with different values of term frequency and toxicity ratio after combining with CharacterBERT

frequency and the ratio of toxicity in the training data. We showed that this model, although extremely simple, gives a close performance to that of CharacterBERT with millions of parameters.

Chapter 8

AEDA: An Easier Data Augmentation Technique for Text Classification

8.1 Introduction

Text classification is a major area of study in natural language processing (NLP) with numerous applications such as sentiment analysis, toxicity detection, and question answering, to name but a few. In order to build text classifiers that perform well, the training data need to be large enough so that the model can generalize to the unseen data. However, for many machine learning (ML) applications and domains, there do not exist sufficient labeled data for training. In this situation, data augmentation (DA) can provide a solution and help improve the performance of ML systems [88, 89, 90]. DA can be carried out in many different ways such as by modifying elements of the input sequence, namely word substitution, deletion, and insertion [9, 39], and back-translation [28]. It can also be performed by noise injection in the input sequence [91] or in the embedding space utilizing a deep language model [86, 10, 92].

Using a deep language model to do DA can be complicated, while word replacement techniques with the help of a word thesaurus, even though a simple method,

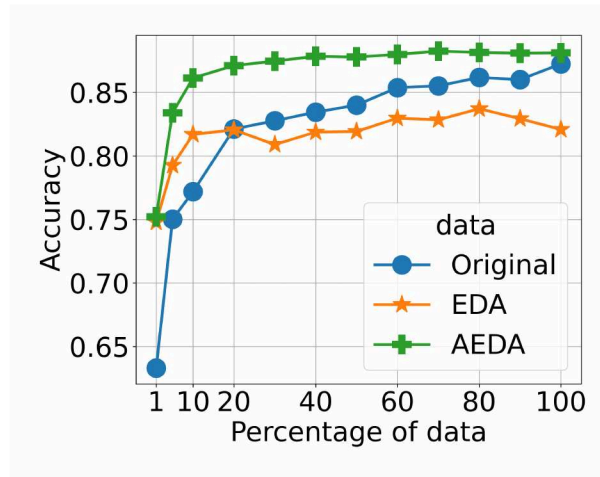


Figure 8.1: Average performance of the generated data using our proposed augmentation method (AEDA) compared with that of the original and EDA-generated data on five text classification tasks. Using both EDA and AEDA, we added 9 augmented sentences to the original training set to train the models. For each task, we ran the models with 5 different seed numbers and took the average score.

risks information loss due to the operations such as deletion and substitution. These operations can even result in changing the label of the input sequence [93], thus misleading the network.

To address these problems, we propose an extremely simple yet effective approach called AEDA (An Easier Data Augmentation) which includes only the insertion of various punctuation marks into the input sequence. AEDA preserves all the input information and does not mislead the network since it keeps the word order intact while changing their positions in that the words are shifted to the right. Our extensive experiments show that AEDA helps the models avoid overfitting (Figure 8.1).

8.2 Related Work

Although the textual content is always increasing, data augmentation is still a highly active area of research since for machine learning applications, especially the new ones, the initial annotated data are usually small. As a result, researchers are constantly coming up with innovative ideas to create new data from the available content.

Some have experimented at the input sequence level performing operations on words. For example, to improve machine translation quality, authors of [89] utilize substitution of common words with rare ones, thus providing more context for the rare words, while [28] uses back-translation where automatically translated data along with the original human-translated data are employed to train a neural machine translation system. In [94], words are replaced with their synonyms for classifying tweets. Similarly, authors of [95] replace sentence fragments from common categories with each other in order to produce new sentences.

Others have opted for using pre-trained language models such as BERT [7]. In [96], the authors utilize contextual augmentation, replacing the words with the prediction of a bidirectional language model at a desired position in the sentence. In [97] and [98], they utilize reinforcement learning with a conditional language model which is carried out by attaching the correct label to the input sequence when training [99]. Working with Transformer model [1], authors of [5] propose Mix-Transformer where two input sentences and their corresponding labels are linearly interpolated to create new samples.

In [91], the authors make use of data noising which can be considered similar to our work with the difference that they replace words choosing from the unigram frequency distribution or insert the underscore character as a placeholder, whereas we insert punctuation characters which usually occur in sentences. The related works mostly use some auxiliary data or a complicated language model to produce augmented data. Conversely, our method is extremely simple to implement and does not need any extra data. In addition, it shows superior performance to EDA in both simple models such as RNNs and CNNs and deep models such as BERT.

8.3 AEDA Augmentation

In order to insert the punctuation marks, we randomly choose a number between 1 and one-third of the length of the sequence which indicates how many insertions will be carried out. The reason is that we want to ensure there is at least one inserted mark and at the same time we do not want to insert too many punctuation marks as too much noise might have a negative effect on the model, although this effect can be investigated in future work. Then, positions in the sequence are also specified in random as many as the selected number in the previous step. In the end, for each chosen position, a punctuation mark is picked randomly from the six punctuation marks in {".", ";", "?", ":", "!", ",", ""}. Table 8.1 shows three augmentation samples by the AEDA technique.

Original	a sad , superior human comedy played out on the back roads of life .
Aug 1	a sad , superior human comedy played out on the back roads ; of life ; .
Aug 2	a , sad . , superior human ; comedy . played . out on the back roads of life .
Aug 3	: a sad ; , superior ! human : comedy , played out ? on the back roads of life .

Table 8.1: Examples of the augmented data using the AEDA technique

8.4 Experimental Setup

Since we compare our proposed method with [9], we used the same codebase as theirs with no changes in the implementation of the models. We executed the code using a GeForce RTX 2070 GPU with 8 GB of memory.

8.4.1 Datasets

We experiment with the same five datasets as our baseline. They include **SST2** [100] Stanford Sentiment Treebank, **CR** [46, 101, 102] Customer Reviews dataset, **SUBJ** [103] Subjectivity/Objectivity dataset, **TREC** [104] Question Classification dataset, and **PC** [105] Pros and Cons dataset. Table 8.2 shows the statistics of the utilized

Dataset	N_{class}	L_{avg}	N_{train}	N_{test}	IVI
SST-2	2	19	7791	1821	15771
CR	2	19	4067	451	9048
SUBJ	2	25	9000	1000	22715
TREC	6	10	5452	500	9448
PC	2	7	40000	5806	26090

Table 8.2: Statistics of the utilized datasets. N_{class} : Number of classes, L_{avg} : Sentence average length, N_{train} : Number of training samples, N_{test} : Number of test samples, IVI : Number of unique words

datasets.

The train and test sets utilized for the experiments for these datasets were not made available by the baseline. Therefore, after collecting them, we shuffled and divided them into train and test sets with almost the same size as the ones reported by the baseline. For the CR dataset, we combined all the reviews from the three cited sources. The annotations included multiple target sentiments for each sentence. Therefore, to convert them into binary classes, we considered a sentence positive if there was no negative sentiment and negative if there was no positive sentiment. The datasets are available along the source code.

8.4.2 Models

To be consistent as well as for a fair comparison of the effects of EDA- and AEDA-augmented data, we used the same Recurrent Neural Network (RNN) [23] and Convolutional Neural Network (CNN) [2] as implemented in the baseline. For the initialization of the models, GloVe word vectors [21] were utilized.

8.5 Results

To evaluate the quality of augmented sentences, we performed experiments using the data augmented by both EDA and AEDA as well as the original data. For the results

Model	Training set size			
	500	2,000	5,000	full set
RNN	73.5	82.6	85.9	87.9
+EDA	76.1	81.3	85.2	86.5
+AEDA	77.8	83.9	87.2	88.6
CNN	76.5	83.8	87.0	87.9
+EDA	77.5	82.2	84.5	86.1
+AEDA	78.5	84.4	86.5	88.1
Average	75.0	83.2	86.5	87.9
+EDA	76.8	81.8	84.9	86.3
+AEDA	78.2	84.2	86.9	88.4

Table 8.3: Comparing average performance of EDA and AEDA across all datasets on different training set sizes. For each training sample, 16 augmented sentences were added. Scores are the average of 5 runs. Numbers are in percentages.

reported in Table 8.3, we added 16 augmentations and for the ones in Figure 8.2, 9 augmentations to be consistent with the baseline. All experiments were repeated with 5 different seed numbers and the average scores are reported.

8.5.1 AEDA Outperforms EDA

The results of the experiments with 500, 2000, 5000 and full dataset sizes for training are reported in Table 8.3. We can see that in some small datasets, EDA improves the results while for bigger ones it has a negative effect on the performance of the models. Conversely, AEDA gives a performance boost on all datasets, showing greater boosts for smaller ones. For instance, with 500 sentences, the average absolute improvement is 3.2% while for full dataset it is 0.5%. The reason why EDA does not perform well can be attributed to the operations such as deletion and substitution which insert more misleading information to the network as the number of augmentations grows. In contrast, AEDA keeps the original information in all augmentations.

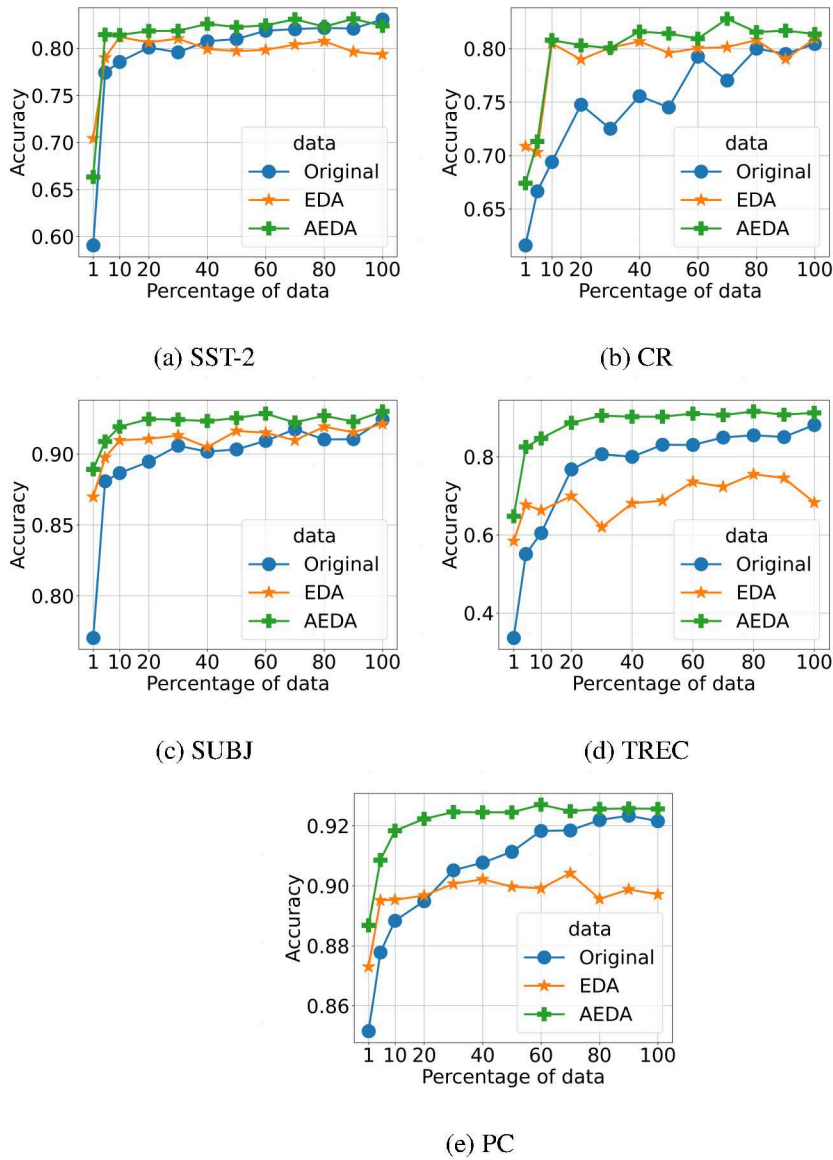


Figure 8.2: Performance of the RNN model trained on various proportions of the original, EDA-generated, and AEDA-generated training data for five text classification tasks. All the scores are the average of 5 runs.

8.5.2 Trend on Training Set Sizes

Figure 8.2 shows how both models perform on different fractions of the training set. These fractions include {1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100} percent. We can see that AEDA outperforms EDA in all tasks as well as showing improvements over the original data. One observation to point out is that also EDA works well on small datasets which can be because of lower number of augmentations compared to the ones reported in Table 8.3.

8.6 Ablation Study

In this section, we investigate how much gain there is for different number of augmentations, the effect of random initialization, and whether AEDA can improve deep models.

8.6.1 Number of Augmentations

Figure 8.3 presents the impact of adding various numbers of augmentations to the training set. We can see that only one augmentation can improve the performance by an absolute amount of 1.5% to 2.5% for all dataset sizes. However, as the augmentations increase, the smallest dataset greatly benefits from that by an improvement of almost 4% while the full dataset only gains 1%. The middle-sized ones have a gain in between (2% to 2.5%).

8.6.2 Effect of Random Initialization

When conducting the experiments, we noticed that different seed numbers produce different results. As a result, we ran the experiments for 5 times. However, in each run with the same seed number, the results can be slightly different due to the local and global generators in TensorFlow. Therefore, to ensure that 5 runs show the correct trend, we chose two of the datasets (CR and TREC) and ran the models for 21 different seeds (zero to 20). From Figure 8.4, we see that the trend is similar to Figure 8.2, which shows the average results of 5 seeds.

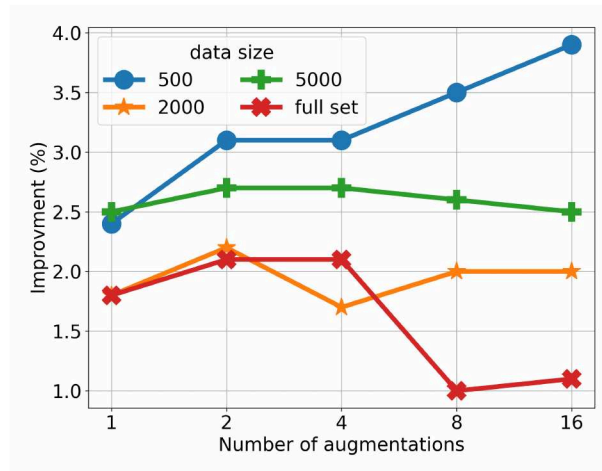


Figure 8.3: Impact of number of augmentations on the performance of the RNN model trained on various training sizes. Scores are the average of 5 runs over the five datasets. The y axis shows the percentage of improvement.

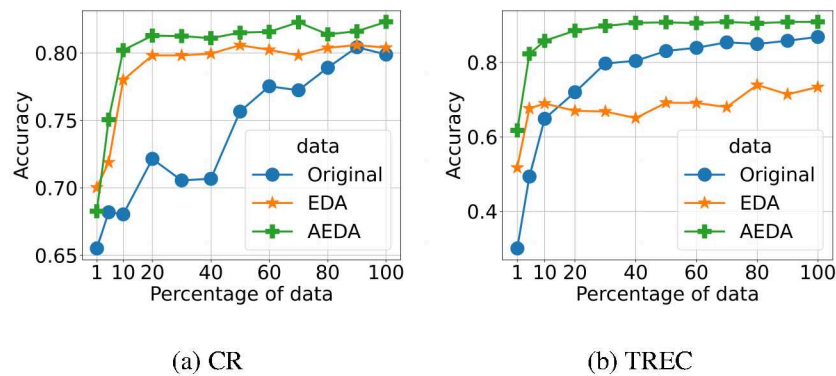


Figure 8.4: Average performance of EDA and AEDA over 21 different seed numbers. The results are in line with the experiments run over 5 seeds.

8.6.3 Using AEDA with Deep Models

The performance of AEDA on a deep model such as BERT is mixed. Table 8.4 shows the results of our experiments with the BERT model. We trained the model used in [93] for 3 epochs with its default settings and observed that adding one augmentation for each training sample increased the performance by 0.15% for SST2 and 0.76% for TREC while making it deteriorate slightly for the others. However, in all cases, except for the CR dataset, it still outperforms the EDA method. The reason why AEDA does not always help a deep model can be the fact that pre-trained models have already seen a considerable amount of data with possibly similar noises to AEDA. Nevertheless, it is worth noting that, as we saw for RNN and CNN models, adding more augmentations might be more advantageous especially for small fractions of the datasets. This can be explored in the future work.

Model	SST2	CR	SUBJ	TREC	PC
BERT	91.85	90.55	97.04	96.48	96.40
+EDA	91.85	90.55	96.24	96.84	96.08
+AEDA	92.00	90.42	96.86	97.24	96.13

Table 8.4: Comparing the impact of EDA and AEDA on the BERT model. The model was trained on the combination of the original data and one augmentation for each training sample. The scores are the average of 5 runs.

8.7 Discussion

Comparing the results that we have gained in our experiments with the ones reported in [9], we can see some discrepancy, especially in the impact of EDA on improving the performance of the models. We speculate that the difference can be caused by the inconsistency in the training and test sets. Although we obtained the datasets from the same references they have specified, some of them are not divided into train and test datasets ready to be used. As mentioned in Section 8.4.1, we randomly divided

them into train and test sets. In addition, some of them have different sizes which can produce different results.

With that said, to conduct a fair evaluation, we kept the same setting for all comparisons in terms of the utilized library and source code, train and test sets, number of augmentations, number of runs, batch size, and learning rate.

8.8 Conclusion and Future Work

We proposed an easy data augmentation technique for text classification tasks. Extensive experiments on five different datasets showed that this extremely simple method which uses punctuation marks outperforms the EDA technique which includes random deletion, insertion, and substitution of words, on all the utilized datasets. The future work will focus on exploiting the proposed method regarding which punctuation marks can have more impact, which ones to add or discard, and how many of them can be used to achieve a better performance. In addition, the question whether the punctuation marks should be inserted randomly or some positions are more effective will be investigated.

Chapter 9

Aspect-Based Emotion Analysis and Multimodal Coreference: A Case Study of Customer Comments on Adidas Instagram Posts

9.1 Introduction

Nowadays, all large companies and brands own a designated page or channel on social media platforms which allows them to directly communicate with various stakeholders, most notably the general public. This rise of social media communication has also propagated research on automatically analysing user-generated content which is often full of opinions and emotions. As a consequence, research on sentiment and emotion analysis has thrived.

Since companies not only want to know which general opinions their stakeholders hold on their company or brand, but also want to learn which products or which

features of their products are greatly appreciated or disliked by the online community. As a result, the research domain of sentiment analysis has started to focus more and more on fine-grained sentiment analysis at the feature or aspect level [106]. This is referred to as Aspect-Based Sentiment Analysis (ABSA), which focuses on the detection of all sentiment expressions within a given document and the concepts and aspects (or features) to which they refer. Following the task description in the SemEval 2015 shared task on this topic [14], aspect-based sentiment analysis can be decomposed into three subtasks, namely aspect term extraction, aspect category classification and aspect term polarity classification.

In its original sense, sentiment analysis involves classifying instances as one of the three classes *positive*, *negative*, and *neutral*, but more recently, many research objectives shifted to extracting more fine-grained emotional information like the emotional categories *anger*, *sadness*, and *joy* [107]. When performing emotion detection at the aspect level, we can analogously refer to this as Aspect-Based Emotion Analysis (ABEA) [108].

Given the rise of more visual content, which is illustrated by the popularity of platforms such as Instagram and TikTok, we want to investigate whether text-based models are sufficient to classify social media content which clearly exhibits both textual and visual information in terms of aspect and sentiment/emotion. Moreover, whenever images are presented, text is often used to pinpoint specific aspects of an image (e.g. *Love the color of these*). By closely analyzing these instances, and especially those where anaphors are used, we wish to get more insights into whether it would be helpful to include visual information in the ABEA pipeline by means of multimodal coreference resolution.

We perform a case study using customer comments on the Adidas Instagram page by collecting 4,900 comments on 175 Instagram images, and annotating them with aspect categories and emotional information. Moreover, the annotators indicated for each comment whether the image was necessary for a full understanding of the comment. By comparing the performance of the aspect classification and emotion analysis models on both types of comments, we can assess whether comments that rely on visual information for a full understanding are more difficult to classify than

comments that do not rely on visual information.

The contributions of this work are twofold. Firstly, we present a multimodal dataset that can be used in the context of aspect-based sentiment and emotion detection, consisting of 4,900 comments on 175 images and annotated with both aspect and emotion labels. The dataset is freely available for further study¹. Moreover, we assess the utility of multimodal coreference resolution in an ABEA framework.

The next section is dedicated to the description of related work (Section 9.2). In Section 9.3, we describe the data collection and annotation process. The methodology of the experiments to assess the importance of visual content for ABEA and the results of these experiments are reported in Section 9.4. The results are further discussed in Section 9.5, followed by a conclusion in Section 9.6.

9.2 Related Work

In its original use, the goal of sentiment analysis was to classify text documents in terms of polarity (i.e. positive or negative) [109]. Through the years, the objective of sentiment analysis evolved to extracting more fine-grained insights about subjective information in texts and the need for sentiment analysis on the feature or aspect level was first expressed by [106]. ABSA received a lot of attention in the context of a shared task at SemEval 2014 [13] and 2015 [14], which provided datasets of English reviews in two domains (laptops and restaurants), annotated with aspect terms, aspect categories and sentiment labels.

The evolution of extracting more and more fine-grained subjective information also caused a switch in focus from polarity to emotion [107]. Instead of focusing on the positive/negative dichotomy, the goal in emotion analysis is to extract specific emotional states, such as the basic emotion categories of Ekman: *anger*, *disgust*, *fear*, *joy*, *sadness*, and *surprise* [110]. Recently, various studies performed emotion detection based on emotional dimensions instead of categories [111, 112]. They follow the theory of [113], who claim that every emotional state can be represented by the three dimensions *valence*, *arousal* and *dominance*.

¹<https://lt3.ugent.be/resources/multimodal-abea/>

Mostly, emotion detection is performed at the sentence or document level. Analogously to ABSA, one could analyze emotions at the aspect level as well, resulting in Aspect-Based Emotion Analysis (ABEA). Some studies have been carried out on this subject [108], but there are no publicly available datasets that have specifically been made for ABEA.

A problem in aspect-based sentiment and emotion analysis is that aspect terms are often not explicitly mentioned. This can manifest itself in various ways, e.g. by an implicit aspect that can only be inferred from the contextual meaning (e.g. *my mouth is still watering!*, which has a food-related but implicit aspect) or by an anaphor referring to an antecedent previously mentioned in the text (e.g. *They were absolutely horrible.*). In an age where visual content is becoming more and more prevalent — see also the work on multimodal NLP [114] — it is even possible that these implicit aspects can only be understood with the help of the image accompanying the text.

Looking at both ABSA and ABEA, the same first two subtasks can be defined: aspect term extraction and aspect category classification. Regarding the latter, one could hypothesize that this task might suffer from these implicit aspects and that coreference resolution is needed to overcome this. However, previous research has shown that coreference resolution does not necessarily improve aspect term classification [115]. When linking anaphors to their correct antecedent in restaurant reviews, this additional semantic information did not really help to better classify the aspects into predefined categories, suggesting that the models have enough with the contextual lexical information alone. However, in the context of multimodal coreference, where text and image appear together, this has not been investigated yet.

9.3 Dataset

9.3.1 Dataset Collection

The Adidas Instagram page (*@adidasoriginals*) has more than 37 million followers with a great number of comments on each post, which makes it a rich page for collecting opinions on the brand and its products. In order to scrape all the posts on the

Instagram page, we first obtained their shortcodes, by which each unique post can be identified. This was done using the Selenium package and the BeautifulSoup library in Python. The Selenium package automates browsing and interacting with the web. We used it to automatically open the page and scroll down until all posts were visible. Then, using the BeautifulSoup library, we extracted the shortcode for each post from the HTML source code of the fully loaded page.

The next step was to open each post and click on the *Load more comments* button in order to acquire all the comments under each post. After all comments were loaded, once again using the HTML source code, we extracted the comments for individual posts. Since our objective was to annotate only the English comments, we used the *langdetect* library² to recognize the language of the comment and filter out the non-English ones. Our tool for downloading shortcodes and comments is available on GitHub³.

In order to download the Instagram images, we used the *scraper* software package⁴. This software contains a command line utility that takes as input a file containing a list of the shortcodes and the name of the directory in which we want to store the images. The shortcode is used to build an absolute path for the post. For each post, the web page is downloaded in HTML format and then the temporary image URL path is collected from the metadata. Subsequently, for each image URL, an HTTP request is made to download the image and save it on the hard disk.

9.3.2 Dataset Annotation

The comments of 175 Instagram images were annotated by two students who were enrolled in the final year of a Bachelor's program in Applied Linguistics. The annotators were provided with an Excel file containing the image and comment. They were asked to view the comments from the perspective of the person who wrote them and to indicate whether an emotion was expressed. If that was the case, they were asked to specify this emotion. Initially, the emotions of interest were *anger, fear, joy, love,*

²<https://pypi.org/project/langdetect>

³<https://github.com/akkarimi/instascraper>

⁴<https://github.com/hachreak/scraper>

Main Category	Subcategories	Example
Company	General, reliability	Looks likes I'm Adidas fan now. Got to make daddy in law happy @jazminchristine_
Marcom	General, promotions	Thanks for the birthday coupon
Personnel	General, friendliness, service, reception, speed, information, availability, familiarity	Still no one reply to me of this wind jacket. As I've leave my phone number to your retail shop staff in Causewaybay! Even no stock or when will restock! But until now still no feedback. What the hell of customer services? Very disappointed
Product	General, price, quality, availability, variety	Hope it will be available in the philippines.
Social media	Content	I was 2013 in chicago...very nice and very nice picture ;) @adidasoriginals
Website	General, information, user-friendliness	What's wrong with the app? I've been trying to place an order 3 days already

Table 9.1: Aspect taxonomy and examples from the dataset for each main category

and *sadness*, conforming to [116]. However, after a trial annotation round, *fear* was removed from the label set as it was almost never indicated. However, the category *longing* was included instead to account for the emotion of desire, which was often expressed in the context of wanting the Adidas product that was represented in the Instagram post. Additionally, the annotators had to rate the emotional dimensions of *valence* (negative-positive) and *arousal* (calm-excited) on a 5-point scale.




Comment	Emotion	Valence	Arousal	Aspect	Image needed?
Hope it will be available in the philippines.	Longing	3	1	Product - availability	Yes  
Been rockin your shoes since the 90s! Sambas, Gazelles, Superstars, NMDs! Thank you!	Joy	4	3	Product - general	No (☹) 

Table 9.2: Examples of annotated comments.

In the next step, the annotators had to indicate the aspect term associated with the emotion (or *null* if the aspect was not mentioned explicitly in the text) and assign the appropriate aspect category (main and subcategory) for this aspect term. The aspect taxonomy is shown in Table 9.1. It was obtained in the framework of a larger project (SentEMO⁵) where aspect category taxonomies for different domains (*retail, hotel, etc*) have been drawn up in close collaboration with representative partners from the industry and on the basis of which the more generic taxonomy as presented here was derived. When multiple emotions and aspects were present in an Instagram post, the sentence was split up according to the number of aspects. It is worth noting that these comments were not taken into account for the remainder of this work. In the last step of the annotation process, the annotators were asked to indicate whether the image was necessary for understanding the comment or not.

In total, 5,140 comments were annotated in this manner. After filtering out the comments with multiple aspects or erroneous annotations, our final dataset comprises 4,900 comments, of which 2,615 were annotated as emotional and 2,285 were neutral. Two annotated examples are shown in Table 9.2.

⁵<https://lt3.ugent.be/projects/multilingual-aspect-based-sentiment-and-emotion-an/>

Class	IAA
Aspect	0.598
Emotion	0.618
Valence	0.466
Arousal	0.337

Table 9.3: Inter-annotator agreement for aspect and emotion categories (Cohens’s Kappa) and emotional dimensions (Krippendorff’s alpha).

The comments accompanying the first ten images were annotated by both annotators (trial annotation round consisting of 90 comments) in order to calculate inter-annotator agreement (in the final dataset, only Annotator 1’s annotations of the first 90 comments were taken into account). For the emotion categories and aspect categories, Cohen’s Kappa was calculated. For the emotional dimensions (*valence* and *arousal*), Krippendorff’s alpha was used. The agreement scores are shown in Table 9.3 and reveal a moderate to substantial agreement for aspect and emotion categories, and a fair to moderate agreement for emotional dimensions.

After this initial trial annotation round, the annotators sat together to align their annotation method. The remaining comments were more or less equally divided among the annotators. The annotators were free to discuss the annotations with each other when necessary in order to further guarantee consistency.

A summary of the data annotations can be found in Tables 9.4 and 9.5. Out of 2,615 emotional comments, *joy* is the most dominant category (1,198 comments), followed by *anger* and *longing* (593 and 589 comments, respectively). As regards the aspect categories, the *product* category was clearly most prevalent. We therefore decided to only keep the subcategories for this particular category, but work with the main categories for the other aspect classes. The *store/office* class was omitted, as there were no instances annotated with this category.

A notable part of the instances contained an implicit aspect term. For 695 out of 2,615 emotional comments, it was not possible to indicate the aspect term association with the emotion and they received the *null* tag as aspect term. Moreover, 883 of





Emotion (#)		Valence (#)		Arousal (#)	
Neutral	2,285	1	167	1	696
Anger	593	2	456	2	1,258
Joy	1,198	3	644	3	500
Longing	589	4	1,079	4	138
Love	191	5	269	5	23
Sadness	44				
	2,317		2,317		2,317
¶	2,583	¶	298	¶	298
Total	4,900	Total	2,615	Total	2,615

Table 9.4: Number of comments per emotion category / emotional rating.  means that the image was needed for understanding the comment, ¶ means that the image was not needed and the text alone was sufficient.

the instances, where an aspect term could be indicated, contained an anaphoric pronoun (*this, that, these, those* and *it*). We can thus say that 1,578 instances (i.e. 60%) contained an implicit aspect term.

For approximately half of the instances (2,317 out of 4,900), visual information was needed to completely understand the comment. This mainly concerned the emotional instances. When only taking the emotional comments into account (see also the numbers for *valence* and *arousal* in Table 9.4 and the aspect categories in Table 9.5), the image was needed for the vast majority of instances (2,317 out of 2,615).

9.4 Experiments & Results

In order to investigate the influence of visual content on the first two subtasks of aspect-based emotion analysis (aspect category classification and emotion analysis), we applied RoBERTa models [16] to our data and compared the performance on the comments for which the annotators indicated that no visual information was needed for a complete understanding versus the comments where the text alone sufficed. As



Aspect (#)	
Company	94
Marcom	20
Personnel	12
Product - availability	225
Product - general	1,650
Product - price	32
Product - quality	35
Product - variety	117
Social media	401
Website	29
	2,317
¶	298
Total	2,615

Table 9.5: Number of comments per aspect category.  means that the image was needed for understanding the comment, ¶ means that the image was not needed and the text alone was sufficient.

we are dealing with short social media texts, we use the RoBERTa-based language model trained on Twitter data from [117]. For aspect classification, we use the base model (twitter-roberta-base⁶) and for emotion, valence and arousal classification, we use the twitter-roberta-emotion model⁷.

For emotion classification, all instances were used (4,900 comments), while for valence, arousal and aspect classification only the non-neutral tweets were taken into account (2,615 comments). We used 10-fold cross validation to finetune the RoBERTa models for 3 epochs, with a dropout of 0.2, maximum sequence length and batch size of 64 and learning rate of 5e-05.

Table 9.6 shows the accuracy of these models. We report the performance on

⁶<https://huggingface.co/cardiffnlp/twitter-roberta-base>

⁷<https://huggingface.co/cardiffnlp/twitter-roberta-base-emotion>



Task	All		¶
Aspect	0.784	0.815	0.544
Emotion	0.675	0.838	0.528
Valence	0.559	0.574	0.450
Arousal	0.693	0.611	0.638

Table 9.6: Accuracy of aspect and emotion classification for the complete dataset (All), and for subsets of the data containing either only instances where visual information was needed for full understanding () or instances where visual information was not needed (¶).

the full subsets, but also calculate the accuracy after filtering the instances based on whether visual information was required for full understanding. On the full dataset, aspect classification is 78% accurate. For emotion classification, accuracy is 68%, and for valence and arousal classification, it amounts to 56% and 69%, respectively.

Contrary to what we expected, the aspect, emotion and valence classification models perform better on comments where visual information is needed, compared to comments where the text alone suffices. Especially for aspect and emotion classification, the difference is substantial (82% versus 54% for aspect and 84% versus 53% for emotion classification). This is counter-intuitive, as the models do not actually have access to that visual information. In the next section, this will be examined in closer detail.

9.5 Discussion

We take a closer look at the comments and predictions to get more insights into the importance of visual information for classification. More specifically, we investigate implicit aspect mentions in the comments. The pronouns *this*, *that*, *these*, *those* and *it* appear in 883 of the 2,615 non-neutral comments, and 695 comments have the *null* tag as aspect term. The vast majority of these comments are indicated by the annotators as needing the image for full understanding (869 of the comments with pronouns

and 622 of the comments with *null*, i.e. 1,491 out of 1,578 or 94%). However, given the results shown in Table 9.6, it does not seem that these implicit aspects cause problems for the classifier, as the performance of aspect classification is even higher for these instances. This suggests that aspect-based sentiment and emotion analysis would not benefit from multimodal coreference resolution.

However, a lot depends on the aspect labels of interest. In our case, the aspect labels seem broad enough to only rely on the text for extracting the aspects. Some examples are shown in Table 9.7. In the sentence *where can I get these same ones?* for example, one does need the image to know to what specific entity is being referred, but the text alone is enough to know that this comment is about a product (and the availability thereof). However, when one needs to know which specific product the comment is about, image recognition is still needed.

Most of the aspects in the dataset are subcategories of the *product* category (cf. Table 9.5). When we look at the top three aspects in the comments with implicit aspect, *product - general* is dominant as well (1,123 comments) followed by the *social media* (244 comments) and *product - availability* category (130 comments). The majority of the *product - general* and *product - availability* instances with implicit aspects are correctly classified (1,165 out of 1,253, i.e. 93%). For the *social media* category however, only 102 out of 244 instances are correctly classified (42%) and are often misclassified as *product - general*.

For emotion classification, it is less surprising that the lack of visual information does not hamper performance, as emotional information is probably only present in the textual comment and not in the originally posted picture. One would expect that there is no difference in performance between the 📷 subset and the ¶ subset. However, the performance on the 📷 subset is higher than that of the ¶ subset. A possible explanation for this could be that the comments which refer to the original picture are more straightforward by themselves.







Comment	Aspect	 / ¶	F/C
Super super in love with this work of art	Product - general		C
where can I get these same ones	Product - availability		C
The same in black and I will buy this zxflux !!!	Product - variety		C
@_6079 this pic is awesome	Social media		F

Table 9.7: Examples of comments that include an anaphor.  means that the image was needed for understanding the comment, ¶ means that the image was not needed and text alone was sufficient. When the classifier made a correct prediction, it is indicated with a C; when a false prediction was made, it is indicated with an F.

9.6 Conclusion

We presented a multimodal dataset that can be used in the context of aspect-based sentiment and emotion detection, consisting of 4,900 comments on 175 images and annotated with both aspect and emotion labels. We assessed the utility of multimodal coreference resolution in an ABEA framework. Based on these preliminary experiments, we can assume that ABEA does not benefit from multimodal coreference resolution. However, when more specific information (e.g. product type, product color, etc) that is more fine-grained than the broad aspect categories, is needed, computer vision techniques will become necessary.

9.7 Acknowledgments

This work was partially funded by the Research Foundation–Flanders under a Strategic Basic Research fellowship with Grant No. 3S004019. Authors would also like to thank Leonardo Rossi for his help with downloading the images in the dataset.

Part IV

Conclusions and Future Work

The last part of the thesis consists of two chapters. In Chapter 10, we summarize and provide some conclusions for the thesis. Then, in the final chapter (Chapter 11), we highlight some of the directions that can be taken in order to address some of the problems faced in this thesis.

Chapter 10

Conclusions

In this thesis, we addressed several tasks in Natural Language Processing (NLP), namely Aspect-Based Sentiment Analysis (ABSA), Toxic Language Detection (TLD), Text Classification (TC), and Aspect-Based Emotion Analysis (ABEA).

In Chapter 5, we introduced a novel architecture called BAT that uses adversarial training with the BERT language model to tackle the ABSA task. In order to perform adversarial training, we first create adversarial examples from the input embeddings. These examples are artificial samples that are similar to the real-world examples but perturbed by a small change so that when introduced to a neural network, the network will classify them wrongly. To create them, we normalize the gradient of the loss for the forward pass of the BERT model and multiply it by a small perturbation number and add the result to the initial embeddings of the real examples. Feeding the network with these examples helps the network to become more robust and show a better performance.

In another effort to tackle the ABSA task, in Chapter 6, we introduced two modules that can be added on top of the BERT model to improve the performance over our previously proposed BAT model. These modules consist of utilizing the BERT layers put together in two parallel and hierarchical manners and take advantage of the last four layers of the BERT architecture.

In Chapter 7, we address the task of toxic language detection using another vari-

ant of the BERT model called CharacterBERT and combining it with an extremely simple yet effective bag-of-words model. While CharacterBERT can deal with out-of-domain vocabulary as well as the misspellings, the bag-of-words model makes sure to extract the toxic words that have a high toxicity ratio. This is carried out by calculating the ratio of the number of times a word is labeled toxic in the training set over the number of times it appears in total. The bag-of-words model alone, though embarrassingly simple, comes very close to the CharacterBERT model in terms of performance, showing the fact that in some cases there is no need to use large pre-trained language models in order to gain a reasonable performance.

Data augmentation is a way to deal with data scarcity in various NLP tasks. However, the cost of augmentation can sometimes be expensive in terms of the time it requires to be carried out as well as the storage it needs and its complexity. To address all these issues, in Chapter 8, we proposed a simple data augmentation technique called AEDA which is easy to implement and requires almost no time to run and no storage except for what is necessary to store the raw data. This method includes only the insertion of punctuation marks into the raw sentences. By doing such, we change the position of the words in the sentence which makes the network generalize better. At the same time, and contrary to the well-known EDA method, we do not delete the words to avoid information loss. Through an extensive set of experiments, we show that it works surprisingly well despite its simplicity and outperforms the EDA augmentation method.

Finally, in Chapter 9, we introduced a new multimodal dataset for Aspect-Based Emotion Analysis (ABEA). The dataset consists of images from the Adidas Instagram page along with their comments. We annotated the comments and took the first steps towards multimodal coreference resolution in ABEA using the annotated dataset. We argued that multimodal coreference resolution is not fruitful in our ABEA setting and provided a baseline for future work by experimenting with the dataset using RoBERTa language model, another variant of the BERT model.

To conclude, we investigated and proposed several methods to perform text analysis using deep neural networks. While the models helped to improve the previous state-of-the-art models, they can have some drawbacks. For instance, the adversarial

training method can be expensive to carry out especially when the utilized language model becomes larger. In fact, the size of the pre-trained deep language models is a major issue that needs to be addressed in order for them to be faster and more environmentally friendly. We addressed this issue by introducing the bag-of-words model for toxic language detection and the AEDA data augmentation method for text classification which do not require large models while being beneficial in improving their performance.

Chapter 11

Future Work

As we showed in the work on the AEDA data augmentation technique, it helps the networks to generalize better on the test data. However, it is not clear which augmented samples help more and which ones help less. It might also be the case that some of them change the ground-truth label. In addition, a study on the effect of different punctuation marks and their positions can shed more light on how effective the AEDA technique is. An investigation of these directions seems to be worthwhile.

Another direction can be addressing the size of the pre-trained language models. While they have achieved state-of-the-art performance in many NLP tasks, their training is time-consuming and requires a large amount of computational time and power. As a result, there is a need to find methods that can reduce this computation cost and make them faster. One possible solution is to use newly-emerged architectures called Adapters [118]. These are small modules that are inserted into the larger architecture and contain a smaller number of parameters compared to the large model. During training, the parameters of the large model are frozen and only the adapter's parameters are trained. This reduces both the training time and the utilized compute power.

Another problem in many NLP applications - especially at the time of their emergence - is the scarcity of training data. Since acquiring a large labeled data for these applications can be time-consuming and costly, we can benefit from methods that

leverage the existing data to create new ones for better training of our models. One solution is to focus on semi-supervised learning which requires a small amount of labeled data and enables us to utilize a large amount of unlabeled data. In this setting, there are two identical networks one of which is trained on labeled data and makes predictions on the unlabeled data, creating a pseudo-labeled dataset. Then, the other network is trained on the newly-created pseudo-labeled dataset. This way, we can leverage the unlabeled dataset to our benefit.

Finally, capsule networks [119] have shown promising results in the computer vision tasks. While they have been extensively explored for image and video analysis, their efficiency in NLP tasks remains somewhat unknown. For a better understanding of the relationship between aspect terms and sentiments or emotions in ABSA and ABEA, one can investigate the effects of capsule networks in conjunction with the pre-trained language models, the adapters, or the traditional word vectors.

Publications

Conference Papers:

- [1] **Akbar Karimi**, Leonardo Rossi, and Andrea Prati. Adversarial Training for Aspect-Based Sentiment Analysis with BERT. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021.
- [2] **Akbar Karimi**, Leonardo Rossi, and Andrea Prati. Improving BERT Performance for Aspect-Based Sentiment Analysis. In *Proceedings of The Fourth International Conference on Natural Language and Speech Processing (IC-NLSP)*. Association for Computational Linguistics, 2021.
- [3] **Akbar Karimi**, Leonardo Rossi, and Andrea Prati. AEDA: An Easier Data Augmentation Technique for Text Classification. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. 2021.
- [4] Luna De Bruyne, **Akbar Karimi**, Orphée De Clercq, Andrea Prati, and Véronique Hoste. Aspect-Based Emotion Analysis and Multimodal Coreference: A Case Study of Customer Comments on Adidas Instagram Posts. In *13th International Conference on Language Resources and Evaluation (LREC), European Language Resources Association, 2022*.

Workshop Paper:

- [5] **Akbar Karimi**, Leonardo Rossi, and Andrea Prati. UniParma at SemEval-2021 Task 5: Toxic Spans Detection Using CharacterBERT and Bag-of-Words

Model. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, Association for Computational Linguistics, 2021.

Published Collaborations:

- [6] Leonardo Rossi, **Akbar Karimi**, and Andrea Prati. A novel region of interest extraction layer for instance segmentation. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021.
- [7] Leonardo Rossi, **Akbar Karimi**, and Andrea Prati. Recursively Refined R-CNN: Instance Segmentation with Self-RoI Rebalancing. In *Computer Analysis of Images and Patterns (CAIP)*, Springer International Publishing, 2021.
- [8] Leonardo Rossi, **Akbar Karimi**, and Andrea Prati. Improving Localization for Semi-Supervised Object Detection. In *21st International Conference on Image Analysis and Processing*, 2022.

Submitted Collaborations:

- [9] Leonardo Rossi, **Akbar Karimi**, and Andrea Prati. Self-Balanced R-CNN for Instance Segmentation. In *Journal of Visual Communication and Image Recognition*, 2022.

Bibliography

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [2] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014.
- [3] Hu Xu, Bing Liu, Lei Shu, and S Yu Philip. Double embeddings and cnn-based sequence labeling for aspect extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 592–598, 2018.
- [4] Zheng Li, Ying Wei, Yu Zhang, Xiang Zhang, and Xin Li. Exploiting coarse-to-fine task transfer for aspect-level sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4253–4260, 2019.
- [5] Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, S Yu Philip, and Lifang He. Mixup-transformer: Dynamic data augmentation for nlp tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3436–3440, 2020.

-
- [6] Akbar Karimi, Leonardo Rossi, and Andrea Prati. Aeda: An easier data augmentation technique for text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2748–2754, 2021.
- [7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.
- [8] Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun’ichi Tsujii. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915, 2020.
- [9] Jason Wei and Kai Zou. 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)*, pages 6383–6389, 2019.
- [10] Akbar Karimi, Leonardo Rossi, and Andrea Prati. Adversarial training for aspect-based sentiment analysis with bert. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 8797–8803. IEEE, 2021.
- [11] Akbar Karimi, Leonardo Rossi, and Andrea Prati. Improving BERT performance for aspect-based sentiment analysis. *arXiv preprint arXiv:2010.11731*, 2020.
- [12] Akbar Karimi, Leonardo Rossi, and Andrea Prati. UniParma at SemEval-2021 task 5: Toxic spans detection using CharacterBERT and bag-of-words model. In *Proceedings of the 15th International Workshop on Semantic Evaluation*

- (*SemEval-2021*), pages 220–224, Online, August 2021. Association for Computational Linguistics.
- [13] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland, August 2014. Association for Computational Linguistics.
- [14] Maria Pontiki, Dimitrios Galanis, Harris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 486–495, 2015.
- [15] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 19–30, 2016.
- [16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [17] Hu Xu, Bing Liu, Lei Shu, and S Yu Philip. Bert post-training for review reading comprehension and aspect-based sentiment analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2324–2335, 2019.
- [18] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.

- [19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [20] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.
- [21] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [22] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.
- [23] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2873–2879, 2016.
- [24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [25] Benjamin Sapp, Ashutosh Saxena, and Andrew Y Ng. A fast data collection and augmentation procedure for object recognition. In *AAAI*, pages 1402–1408. Chicago, IL, 2008.
- [26] David Janiszek, Renato De Mori, and E Bechet. Data augmentation and language model adaptation. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, volume 1, pages 549–552. IEEE, 2001.

- [27] David A Van Dyk and Xiao-Li Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.
- [28] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, 2016.
- [29] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.
- [30] Ramit Sawhney, Megh Thakkar, Shivam Agarwal, Di Jin, Diyi Yang, and Lucie Flek. Hypmix: Hyperbolic interpolative data augmentation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9858–9868, 2021.
- [31] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, 2018.
- [32] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097, 2019.
- [33] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [34] Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. Nrcanada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442, 2014.

- [35] Pablo Gamallo and Marcos Garcia. Citius: A naive-bayes strategy for sentiment analysis on english tweets. In *Proceedings of the 8th international Workshop on Semantic Evaluation (SemEval 2014)*, pages 171–175, 2014.
- [36] Shruti Wakade, Chandra Shekar, Kathy J Liszka, and Chien-Chung Chan. Text mining for sentiment analysis of twitter data. In *Proceedings of the International Conference on Information and Knowledge Engineering (IKE)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2012.
- [37] Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, volume 1, pages 61–67, 1999.
- [38] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.
- [39] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.
- [40] Pengfei Liu, Shafiq Joty, and Helen Meng. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443, 2015.
- [41] Yequan Wang, Minlie Huang, Li Zhao, et al. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.
- [42] Yukun Ma, Haiyun Peng, and Erik Cambria. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

- [43] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.
- [44] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna Estrach, Dumitru Erhan, Ian Goodfellow, and Robert Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.
- [45] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016.
- [46] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.
- [47] Ivan Titov and Ryan McDonald. A joint model of text and aspect ratings for sentiment summarization. In *proceedings of ACL-08: HLT*, pages 308–316, 2008.
- [48] Tun Thura Thet, Jin-Cheon Na, and Christopher SG Khoo. Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of information science*, 36(6):823–848, 2010.
- [49] Soujanya Poria, Erik Cambria, and Alexander Gelbukh. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108:42–49, 2016.
- [50] Alexander Rietzler, Sebastian Stabinger, Paul Opitz, and Stefan Engl. Adapt or get left behind: Domain adaptation through bert language model finetuning for aspect-target sentiment classification. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4933–4941, 2020.

- [51] Pinlong Zhao, Linlin Hou, and Ou Wu. Modeling sentiment dependencies with graph convolutional networks for aspect-level sentiment classification. *Knowledge-Based Systems*, 193:105443, 2020.
- [52] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [53] Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. Attentional encoder network for targeted sentiment classification. *arXiv preprint arXiv:1902.09314*, 2019.
- [54] Xin Li, Lidong Bing, Piji Li, Wai Lam, and Zhimou Yang. Aspect term extraction with history attention and selective transformation. *arXiv preprint arXiv:1805.00760*, 2018.
- [55] Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 388–397, 2017.
- [56] Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for natural language understanding. In *International Conference on Learning Representations*, 2019.
- [57] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.
- [58] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pages 2137–2146, 2018.
- [59] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

- [60] Abd Samad Hasan Basari, Burairah Hussin, I Gede Pramudya Ananta, and Junta Zeniarja. Opinion mining of movie review using hybrid method of support vector machine and particle swarm optimization. *Procedia Engineering*, 53:453–462, 2013.
- [61] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.
- [62] Richard O Duda, Peter E Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.
- [63] Liviu P Dinu and Iulia Iuga. The naive bayes classifier in opinion mining: in search of the best feature set. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 556–567. Springer, 2012.
- [64] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
- [65] Yoav Kantor, Yoav Katz, Leshem Choshen, Edo Cohen-Karlik, Naftali Liberman, Assaf Toledo, Amir Menczel, and Noam Slonim. Learning to combine grammatical error corrections. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 139–148, 2019.
- [66] Joe Davison, Joshua Feldman, and Alexander M Rush. Commonsense knowledge mining from pretrained models. 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)*, pages 1173–1178, 2019.
- [67] Yifan Peng, Shankai Yan, and Zhiyong Lu. Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 58–65, 2019.

- [68] Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, 2019.
- [69] Jesse Vig and Yonatan Belinkov. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, 2019.
- [70] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune BERT for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019.
- [71] Wei Yang, Yuqing Xie, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. Data augmentation for BERT fine-tuning in open-domain question answering. *arXiv preprint arXiv:1904.06652*, 2019.
- [72] Yang Liu. Fine-tune BERT for extractive summarization. *arXiv preprint arXiv:1903.10318*, 2019.
- [73] Mickel Hoang, Oskar Alija Bihorac, and Jacobo Rouces. Aspect-based sentiment analysis using bert. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 187–196, 2019.
- [74] Chi Sun, Luyao Huang, and Xipeng Qiu. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *Proceedings of NAACL-HLT*, pages 380–385, 2019.
- [75] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, 2019.
- [76] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

- [77] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537, 2015.
- [78] Leonardo Rossi, Akbar Karimi, and Andrea Prati. A novel region of interest extraction layer for instance segmentation. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2203–2209. IEEE, 2021.
- [79] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.
- [80] Maria Pontiki, Dimitrios Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. Semeval-2014 task 4: Aspect based sentiment analysis. *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 27–35, 01 2014.
- [81] John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*, 2021.
- [82] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus.
- [83] Yoon Kim, Yacine Jernite, David Sontag, and Alexander Rush. Character-aware neural language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.
- [84] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, 2018.

- [85] Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1054–1064, 2020.
- [86] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4163–4174, 2020.
- [87] Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Gralinski. ApplicaAI at semeval-2020 task 11: On RoBERTa-CRF, span CLS and whether self-training helps them. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1415–1424, 2020.
- [88] Anton Ragni, Kate M Knill, Shakti P Rath, and Mark JF Gales. Data augmentation for low resource languages. In *INTERSPEECH 2014: 15th Annual Conference of the International Speech Communication Association*, pages 810–814. International Speech Communication Association (ISCA), 2014.
- [89] Marzieh Fadaee, Arianna Bisazza, and Christof Monz. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, 2017.
- [90] Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. Daga: Data augmentation with a generation approach for low-resource tagging tasks. *arXiv preprint arXiv:2011.01549*, 2020.
- [91] Ziang Xie, Sida I Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y Ng. Data noising as smoothing in neural network language

- models. In *5th International Conference on Learning Representations, ICLR 2017*, 2019.
- [92] Siddhant Garg and Goutham Ramakrishnan. Bae: Bert-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, 2020.
- [93] Varun Kumar, Ashutosh Choudhary, and Eunah Cho. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Lifelong Learning for Spoken Language Systems*, pages 18–26, 2020.
- [94] William Yang Wang and Diyi Yang. That’s so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563, 2015.
- [95] Jacob Andreas. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, 2020.
- [96] Sosuke Kobayashi. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, 2018.
- [97] Zhiting Hu, Bowen Tan, Russ R Salakhutdinov, Tom M Mitchell, and Eric P Xing. Learning data manipulation for augmentation and weighting. *Advances in Neural Information Processing Systems*, 32:15764–15775, 2019.
- [98] Ruibo Liu, Guangxuan Xu, Chenyan Jia, Weicheng Ma, Lili Wang, and Soroush Vosoughi. Data boost: Text data augmentation through reinforcement

- learning guided conditional generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9031–9041, 2020.
- [99] Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. Conditional bert contextual augmentation. In *International Conference on Computational Science*, pages 84–95. Springer, 2019.
- [100] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, 2013.
- [101] Xiaowen Ding, Bing Liu, and Philip S Yu. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 international conference on web search and data mining*, pages 231–240, 2008.
- [102] Qian Liu, Zhiqiang Gao, Bing Liu, and Yuanlin Zhang. Automated rule selection for aspect extraction in opinion mining. In *Twenty-Fourth international joint conference on artificial intelligence*, 2015.
- [103] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 271–278, 2004.
- [104] Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.
- [105] Murthy Ganapathibhotla and Bing Liu. Mining opinions in comparative sentences. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 241–248, 2008.
- [106] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

- [107] Saif M Mohammad. Sentiment analysis: Detecting valence, emotions, and other affectual states from text. In *Emotion measurement*, pages 201–237. Elsevier, 2016.
- [108] Swapnil Bhagaji Padme and Pallavi V. Kulkarni. Aspect based emotion analysis on online user-generated reviews. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–5, 2018.
- [109] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics, July 2002.
- [110] Paul Ekman. An argument for basic emotions. *Cognition & Emotion*, 6(3-4):169–200, 1992.
- [111] Sven Buechel and Udo Hahn. Emotion analysis as a regression problem - dimensional models and their implications on emotion representation and metrical evaluation. In *ECAI 2016 - Proceedings of the 22nd European Conference on Artificial Intelligence*, pages 1114–1122, The Hague, The Netherlands, 2016. IOS Press.
- [112] Saif Mohammad and Svetlana Kiritchenko. Understanding emotions: A dataset of tweets to study interactions between affect categories. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 198–209, Miyazaki, Japan, 2018. European Language Resources Association (ELRA).
- [113] Albert Mehrabian and James A Russell. *An Approach to Environmental Psychology*. MIT Press, 1974.
- [114] Julia Kruk, Jonah Lubin, Karan Sikka, Xiao Lin, Dan Jurafsky, and Ajay Divakaran. Integrating text and image: Determining multimodal document intent

- in Instagram posts. 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)*, pages 4622–4632, Hong Kong, China, November 2019. Association for Computational Linguistics.
- [115] Orphée De Clercq and Veronique Hoste. It’s absolutely divine! can fine-grained sentiment analysis benefit from coreference resolution? In *Proceedings of the Third Workshop on Computational Models of Reference, Anaphora and Coreference (CRAC 2020)*, pages 11–21. Association for Computational Linguistics (ACL), 2020.
- [116] Luna De Bruyne, Orphée De Clercq, and Véronique Hoste. An emotional mess! deciding on a framework for building a Dutch emotion-annotated corpus. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1643–1651, Marseille, France, May 2020. European Language Resources Association.
- [117] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online, November 2020. Association for Computational Linguistics.
- [118] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.
- [119] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International conference on artificial neural networks*, pages 44–51. Springer, 2011.

Acknowledgments

Grazie ad Andrea Prati per la guida ed il supporto durante questi tre anni. Grazie ai miei colleghi Leonardo, Luca, Eleonora, Federico e Tomaso. Grazie a tutti i tesisti e a tutti i Professori del Dipartimento di Ingegneria e Architettura.