VRIJE
UNIVERSITEIT
AMSTERDAM

mobiquity

Aspect Based Sentiment Analysis on Online Reviews using Machine and Deep Learning

Niki Vatzola

MSc Business Analytics Thesis

Aspect Based Sentiment Analysis on Online Reviews using Machine and Deep Learning

Niki Vatzola

MSc Business Analytics Thesis

December 2020

Supervisors:
Dr. Eliseo Ferrante
Dr. Alessandro Zocca
Vrije Universiteit Amsterdam
Faculty of Science
De Boelelaan 1081a
1081 HV Amsterdam

Supervisors:
Dr. Vesa Muhonen
Dr. Dennis Timmers
Mobiquity Inc.
Tommaso Albinonistraat 9
1083 HM Amsterdam

Preface

This thesis is written as the graduation project in order to obtain the MSc degree of Business Analytics at Vrije Universiteit Amsterdam. The master's program duration is two years including a six-month internship at a company. Business Analytics is an interdisciplinary degree in the fields of mathematics and computer science.

The research presented here has been conducted during the six month internship at Mobiquity, a digital consultancy Company. The focus of the research is on performing aspect based sentiment analysis on online reviews using machine and deep learning algorithms and is built on the knowledge acquired during the course of the degree.

I would like to thank my supervisors at Mobiquity, Vesa Muhonen and Dennis Timmers, who guided me through that period and helped me gain insights into the data world.

I would also like to thank my supervisor from the university, Eliseo Ferrante for his guidance and feedback and Alessandro Zocca for being the second reader of my thesis and providing feedback.

Moreover, I would like to thank the Data Science team members at Mobiquity that helped me become part of the team despite the situation with Corona virus which created unprecedented conditions.

Finally, I would like to thank my family, friends and boyfriend who have been very Supportive during the whole period of the master's program.

Summary

Problem Definition - The goal of this research is to apply machine and deep learning models in order to automate the procedure of extracting the topics of online reviews and their polarity. The models should be able to extract the topics that the reviews are talking about and the sentiment about them. During the research we were able to identify how models perform, what are the advantages and disadvantages of each model and which model is able to give the best results in terms of accuracy.

Methodology - For performing aspect based sentiment analysis we divided the procedure into two different sub-tasks. The first task is aspect extraction and the second is defining the sentiment of the extracted aspect. For both tasks we tried various machine and deep learning models and compared their output to decide which gives the most accurate results. Moreover, we tried different combinations of input encodings and models to identify which combinations work better.

Results - This report includes the results for all the machine and deep learning models for both aspect extraction and sentiment analysis. For both aspect classification and sentiment analysis the best model is a deep learning network combined with a powerful language pre-trained model.

Recommendation - The models that were created can be used in labeled online reviews to identify the topics and their sentiments. For future work it is recommended to include more complex ways to input the predicted aspect to the sentiment model.

Contents

1 Introduction **1**

2 Related work **3**

3 Data **5**

3.1 Dataset information 5

3.2 Exploratory data analysis 5

3.2.1 Restaurant domain 5

3.2.2 Laptop domain 9

3.3 Data augmentation 12

3.4 Preprocessing pipeline 13

4 Word Embeddings **15**

4.1 Count Vectorizer and Tf-Idf 15

4.2 Global vectors for word representation 17

4.3 Domain specific embedding 17

5 Methodology **19**

5.1 Latent Dirichlet allocation 19

5.2 Seeded Latent Dirichlet allocation 22

5.3 Biterm topic model 23

5.4 Neural networks 24

5.4.1 Multi-layer perceptron 27

5.5 Support vector machine 27

5.6 Naive Bayes 29

5.7 Random forest 29

5.8 Bidirectional long short-term memory . . . 30
5.9 Attention models . . . 32
5.10 Transformers Model . . . 34
5.10.1 RoBERTa . . . 36

6 Experimental Structure **38**
6.1 Aspect extraction . . . 38
6.2 Sentiment analysis . . . 40
6.3 Evaluation measures . . . 42

7 Results **44**
7.1 Aspect categorization . . . 44
7.1.1 Latent Dirichlet allocation . . . 44
7.1.2 Seeded LDA . . . 48
7.1.3 Biterm topic model . . . 49
7.1.4 Support vector machine . . . 51
7.1.5 Multi-layer perceptron . . . 54
7.1.6 Long short-term memory . . . 55
7.1.7 RoBERTa . . . 57
7.1.8 Combination of RoBERTa and LSTM . . . 58
7.2 Aspect extraction result summary . . . 60
7.3 Sentiment analysis . . . 64
7.3.1 Naive Bayes classifier . . . 65
7.3.2 Random forest . . . 66
7.3.3 Support vector machine . . . 67
7.3.4 Multi-layer perceptron . . . 71
7.3.5 Long sort-term memory . . . 72
7.3.6 RoBERTa . . . 75
7.3.7 Combination of RoBERTa and LSTM . . . 76
7.4 Comparison models . . . 80
7.5 Sentiment extraction result summary . . . 81

8 Business perspective **88**

9 Conclusion **90**

Appendix A Confusion Matrices for augmented data **101**

Appendix B Comparison Models Confusion Matrices **107**

List of Figures

3.1 Number of different aspects in the sentences 7
3.2 Sentiment distribution over the sentences in the corpus 8
3.3 Number of sentences per aspect category 9
3.4 Number of different aspects in the sentences 10
3.5 Sentiment distribution over the sentences in the corpus 11
3.6 Number of sentences per aspect category 11

5.1 LDA plated schema [6] 20
5.2 Perceptron architecture [1] 25
5.3 Multi layer neural network [2] 26
5.4 Support vector classifier [5] 28
5.5 Long Short Term Memory Cell [27] 31
5.6 Bidirectional LSTM [12] 32
5.7 Illustration of the Attention model [4] 33
5.8 Transformers model [29] 34
5.9 Scaled Dot-Product Attention (left) / Multi-Head Attention (right) [29] 35

6.1 Aspect categorization pipeline 38
6.2 Sentiment analysis pipeline 40

7.1 Example output of LDA 45
7.2 Important words for LDA for aspect classification, Restaurant domain 45
7.3 Important words for LDA for aspect classification, Laptop domain . 46
7.4 LDA Restaurant Topic distance map before augmentation 47
7.5 LDA Restaurant Topic distance map after augmentation 47

7.6 LDA Laptop Topic distance map before augmentation 48
7.7 LDA Laptop Topic distance map after augmentation 48
7.8 Bitterm Restaurant Topic distance map before augmentation 50
7.9 Bitterm Restaurant Topic distance map after augmentation 50
7.10 Topic distance map before augmentation Laptop domain 51
7.11 Topic distance map after augmentation Laptop domain 51
7.12 Seeded LDA input weights 52
7.13 Input's weights for Restaurant domain 52
7.14 Input's weights for Laptop domain 53
7.15 Best Model for Aspect Categorization 61
7.16 Restaurant domain topic distribution 61
7.17 Laptop domain topic distribution 62
7.18 Confusion matrix for Restaurants with NB 65
7.19 Confusion matrix for Laptops with NB 65
7.20 Confusion matrix for Restaurants with RF 67
7.21 Confusion matrix for Laptops for with RF 67
7.22 Confusion matrix for Restaurants with SVM 68
7.23 Confusion matrix for Laptops with SVM 68
7.24 Most important words for Restaurant sentiment 68
7.25 Most important words for Laptop sentiment 69
7.26 Confusion matrix for Restaurants with GloVe as input to SVM with averaging 70
7.27 Confusion matrix for Laptops with GloVe as input to SVM with averaging 70
7.28 Confusion matrix for Restaurants with MLP 71
7.29 Confusion matrix for Laptops with MLP 71
7.30 Confusion matrix for Restaurants LSTM with GloVe 73
7.31 Confusion matrix for Laptops LSTM with GloVe 73
7.32 Confusion matrix for Restaurants with LSTM, GloVe and domain embedding 74

7.33 Confusion matrix for Laptops with LSTM, GloVe and domain embedding . . . 74
7.34 Confusion matrix for Restaurants with Tf-Idf and LSTM . . . 75
7.35 Confusion matrix for Laptops with Tf-Idf and LSTM . . . 75
7.36 Confusion matrix for Restaurants with RoBERTa . . . 76
7.37 Confusion matrix for Laptops with RoBERTa . . . 76
7.38 Confusion matrix for Restaurants with RoBERTa and LSTM . . . 77
7.39 Confusion matrix for Laptops with RoBERTa and LSTM . . . 77
7.40 Confusion matrix for Restaurants with RoBERTa and LSTM and GloVe . . . 78
7.41 Confusion matrix for Laptops with RoBERTa and LSTM and GloVe 78
7.42 Confusion matrix for Restaurants with RoBERTa and GloVe and Domain Specific embedding as input to the LSTM . . . 79
7.43 Confusion matrix for Laptops with RoBERTa and GloVe and Domain Specific embedding as input to the LSTM . . . 79
7.44 Best model for sentiment analysis . . . 82

A.1 Restaurant domain naive Bayes . . . 101
A.2 Laptop domain naive Bayes . . . 101
A.3 Restaurant domain random forest . . . 102
A.4 Laptop domain random forest . . . 102
A.5 Confusion matrix for Restaurants with SVM . . . 102
A.6 Confusion matrix for Laptops with SVM . . . 102
A.7 Confusion matrix for Restaurants with MLP . . . 103
A.8 Confusion matrix for Laptops with MLP . . . 103
A.9 Confusion matrix for Restaurants LSTM with GloVe . . . 103
A.10 Confusion matrix for Laptops LSTM with GloVe . . . 103
A.11 Confusion matrix for Restaurant with LSTM, GloVe and domain embedding . . . 104
A.12 Confusion matrix for Laptops with LSTM, GloVe and Domain embedding . . . 104
A.13 LSTM with Tf-Idf for Restaurant domain . . . 104

A.14 LSTM with Tf-Idf for Laptop domain 104
A.15 Confusion matrix for Restaurants with RoBERTa 105
A.16 Confusion matrix for Laptops for augmented data 105
A.17 Confusion matrix for Restaurants with RoBERTa and LSTM 105
A.18 Confusion matrix for Laptops with RoBERTa and LSTM 105
A.19 Confusion matrix for Restaurants with RoBERTa and LSTM and GloVe 106
A.20 Confusion matrix for Laptops with RoBERTa and LSTM and GloVe 106
A.21 Confusion matrix for Restaurants with LSTM and RoBERTa, GloVe and domain embeddings 106
A.22 Confusion matrix for Laptops with LSTM and RoBERTa, GloVe and domain embeddings 106

B.1 Restaurant Domain AE-LSTM 107
B.2 Laptop Domain AE-LSTM 107
B.3 Restaurant Domain AT-LSTM 108
B.4 Laptop Domain AT-LSTM 108
B.5 Restaurant Domain ATAE-LSTM 108
B.6 Laptop Domain ATAE-LSTM 108
B.7 Restaurant Domain HEAT 109
B.8 Laptop Domain HEAT 109
B.9 Restaurant Domain Capsule Network 109
B.10 Laptop Domain Capsule Network 109

List of Tables

3.1 Example of the Restaurant dataset 6
3.2 No Aspect/Polarity Sentences 7
3.3 Example of the Laptop dataset 10
3.4 Augmentation example in Restaurant domain 13
3.5 Augmentation example in Laptop domain 13

5.1 Seed Words in Restaurant domain 23
5.2 Seed Words in Laptop domain 23

6.1 Word embeddings and models for aspect categorization 39
6.2 Word embeddings and models for sentiment analysis 41
6.3 Confusion matrix 43
6.4 Confusion matrix for each class 43

7.1 LDA results for Restaurant dataset 46
7.2 LDA Results for Laptop dataset 47
7.3 Seeded LDA results for Restaurants 49
7.4 Seeded LDA results for Laptops 49
7.5 Biterm model results for Restaurant dataset 50
7.6 Biterm model results for Laptop dataset 51
7.7 SVM results for Restaurant domain 53
7.8 SVM results for Laptop domain 53
7.9 SVM with GloVe in Restaurant domain 54
7.10 SVM with GloVe in Laptop domain 54
7.11 MLP results for Restaurants 55
7.12 MLP results for Laptops 55

7.13 LSTM with GloVe for Restaurants 55
7.14 LSTM with GloVe for Laptops 55
7.15 GloVe and Domain Specific as input to the LSTM for Restaurants 56
7.16 GloVe and Domain Specific as input to the LSTM for Laptops . . . 56
7.17 Tf-Idf as input to LSTM 57
7.18 RoBERTa with two dense layers for Restaurants 57
7.19 RoBERTa with two dense layers for Laptops 57
7.20 RoBERTa as input to the LSTM for Restaurants 58
7.21 RoBERTa as input to the LSTM for Laptops 58
7.22 RoBERTa and GloVe as input to the LSTM for Restaurants 58
7.23 RoBERTa and GloVe as input to the LSTM for Laptops 59
7.24 LSTM with input GloVe and Domain Specific embedding and RoBERTa for Restaurants 59
7.25 LSTM with input GloVe and Domain Specific embedding and RoBERTa for Laptops 59
7.26 Aspect categorization results without data augmentation 60
7.27 Aspect categorization results with data augmentation 63
7.28 Sentiment analysis with naive Bayes for Restaurants 65
7.29 Sentiment analysis with naive Bayes for Laptops 65
7.30 Sentiment analysis with random forest for Restaurants 66
7.31 Sentiment analysis with random forest for Laptops 66
7.32 Sentiment analysis with SVM for Restaurants 67
7.33 Sentiment analysis with SVM for Laptops 68
7.34 SVM with Glove for Restaurant domain 69
7.35 SVM with Glove for Laptop domain 70
7.36 Sentiment analysis with MLP for Restaurants 71
7.37 Sentiment analysis with MLP for Laptops 71
7.38 Sentiment analysis with LSTM and GloVe for Restaurants 72
7.39 Sentiment analysis with LSTM and GloVe for Laptops 72
7.40 LSTM with input of GloVe and Domain Specific for Restaurants . . 73
7.41 LSTM with input of GloVe and Domain Specific for Laptops 73

7.42 Tf-Idf as input to the LSTM Restaurant domain 74
7.43 Tf-Idf as input to the LSTM Laptop domain 74
7.44 RoBERTa and two dense layers for Restaurants 75
7.45 RoBERTa and two dense layers for Laptops 76
7.46 RoBERTa as input to the LSTM for Restaurants 77
7.47 RoBERTa as input to the LSTM for Laptops 77
7.48 RoBERTa and GloVe as input to the LSTM for Restaurants 78
7.49 RoBERTa and GloVe as input to the LSTM for Laptops 78
7.50 RoBERTa and GloVe and Domain Specific embedding as input to the LSTM for Restaurants 79
7.51 RoBERTa and GloVe and Domain Specific embedding as input to the LSTM for Laptops 79
7.52 Comparison models Performance 81
7.53 Sentiment results without data augmentation 82
7.54 Sentiment Results after data augmentation 84
7.54 Sentiment Results after data augmentation 85
7.55 Example of “difficult sentences” 86
7.56 Performance of the best model and comparison models in the difficult sentences 86

8.1 Number of Predicted categories 89

1. Introduction

In the past few years more and more people write their opinions about products and services online. They discuss about their purchases or services that were provided to them in order to inform other people about the advantages or the disadvantages of them. Moreover, more people tend to read these reviews and get informed before buying a product or visiting a place. Online reviews are able to determine customers' choices and affect their opinions.

In addition to that, online reviews are not only useful for the customers but also for the companies. Companies are able to gain important information about them based on customers' opinions. Being able to explore the content of the reviews could give the companies great insights and knowledge. They could identify the problems customers have with their brand, determine their flaws and try to solve them.

In that case, determining only the sentiment of a review is not enough. Knowing that customers' opinions are negative or positive without knowing the reason does not provide the necessary information. That is the reason why, apart from extracting the sentiment, it is important to know the topics the reviews are referring to. The main goal of this research is to conduct *aspect based sentiment analysis on online reviews. In order to achieve that we will try different machine and deep learning approaches. We will identify the model that gives the best results and the advantages and drawbacks of various models. Moreover, we will examine the contribution of data augmentation to our problem. Finally, we will combine different word embeddings with different models to see how they interact and perform.*

The structure of the report is the following. In the beginning we are going to talk about the datasets and analyze their characteristics. Then we are going to introduce the word embeddings that we used as input to the models. Afterwards we are going to analyze the machine and deep learning algorithms that we used to to process the sentences. The chapter after that includes the pipeline of each task and the results that we get. Then we are going to discuss the business perspective of the problem. Finally, we are going to end the report with the conclusion, the limitations during our research and the future work.

2. Related work

In the past few years aspect based sentiment analysis has been a field of natural language processing (NLP) that gains the attention of more and more researchers. The increasing percentage of people expressing their opinions about Services and products online has increased the need for those to be evaluated and have their content extracted. Some approaches that have been followed to extract the aspect categories and the sentiment about these are presented in [47]. The majority of the papers have divided this process in two different sub-tasks. The first is aspect extraction and the second is sentiment definitions of the aspects.

Hu and Liu in [19] are using a frequency based method to extract the categories of the reviews. This idea is based on the fact that some words in the text might be used more often than others to describe a topic. Their method uses only nouns and compound nouns as possible aspects as stated in [60]. In [26] the authors suggest a modification of this approach, which instead uses grammatical dependencies aiming to identify the less frequent categories. The problem of the aforementioned solutions is that the nouns and compound nouns can be mistaken by the models as aspects. In order to resolve this obstacle, [46] and [24] are proposing the use of statistics to create filters for the noun phrases. The second approach is by using the relationship between words. The models consider the grammatical and syntactical patterns hidden in the text. In [42], a method to find dependency parsing between words and their roots is presented. This method has been reproduced and modified from other papers as well but we will not further analyze it. The interested readers can find more information on [60].

Supervised approaches have been used as well for aspect categorization. Shu et

al. in [50] and [49] are using hidden Markov model and conditional random field (CRF) to perform supervised aspect extraction. Finally, unsupervised approaches have been used to perform the same task. The most used model is Latent Dirichlet allocation (LDA) and some modifications of it such as maximum entropy-LDA as presented in [6] and [63] respectively. Additionally, there are models that use deep learning approaches for topic extraction. Recurrent Neural Networks, Long Short-Term Memory are used in combination with CRF to extract the aspects of the sentences as mentioned in [15] and [28].

The second sub-task is defining the sentiment of the extracted aspect categories. Again, it can be divided in two main categories, machine and deep learning. Machine learning approaches include naive Bayes mentioned in [31], decision tree and random forest referred in [3] and Support vector machine (SVM) and its variations. Krishna et al. in [22] present a feature-based SVM that calculates the feature importance and removes those which are not important. In addition to machine learning models, researchers have also used deep learning for sentiment analysis. Long Short-Term Memory (LSTM) networks, Gated Recurrent Unit (GRU) networks and Convolutional Neural Networks (CNN) are commonly used models for sentiment analysis [32]. There are many implementation and variations of these models which were created to do that task. An example of a CNN as presented in [41] is able to achieve good results in sentiment analysis of the sentences based on the detected aspects. However, the most promising models that achieve the state of the art nowadays, are the ones which use attention mechanisms [4]. An example is the work described in [9] which uses a hierarchical attention model which is able to "pay attention" to specific words and define the polarity of a sentence based on target words. The attention mechanism is also utilized by the Transformer [55] models such as Bert and RoBERTa presented in [53] and [37]. Finally, there are approaches which combine two or more of the aforementioned models like the one presented in [57] which combines attention model and LSTM.

3. Data

In this chapter the dataset that has been used will be discussed. In the first section there will be a description of the dataset. In section 2 a data analysis will be made and in section 3 the pre-processing pipeline will be presented.

3.1 Dataset information

The datasets that were used for our research are taken from the International Workshop on Semantic Evaluation 2016 [40].

3.2 Exploratory data analysis

The first dataset is domain-specific for Restaurants and consists of approximately 2000 sentences in English. The initial dataset was provided from [14]. Furthermore, two research teams contributed in the creation of this dataset: The institute for Language and Speech Processing in Athens, Greece and the department of Informatics in Athens University of Economics and Business. The second dataset is domain-specific for Laptops and consists of approximately 2500 sentences in English. It is provided by the same organisation for the same purposes [13].

3.2.1 Restaurant domain

The dataset consist of columns which include the id of each review, the id of each sentence, the sentence itself, the targets words that would be important for defining the aspect category, the aspect categories for which the sentence is talking about, a list with subcategories of the main aspect category and the polarity of

the aspects. The SemEval workshop was divided into 3 different sub-tasks which include aspect category detection, aspect term detection and sentiment polarity detection. However, we did not conduct sentiment analysis based on aspect terms thus we did not use all of the previously mentioned columns. The aspect categories for all the sentences are Restaurant, Food, Ambience, Drinks, Location and Service and the sentiment categories are positive, negative and neutral. The following table contains a few examples of the sentences the aspect categories and their polarity.

Table 3.1: Example of the Restaurant dataset

Sentence	**Aspect Category**	**Aspect Polarity**
The pizza is yummy and I	Food	positive
like the atmosphere	Ambience	positive
The food here is rather good,	Food	positive
but only if you like to wait for it	Service	negative
After all that, they complained to me about the small tip	Service	negative

In Figure 3.1 we can see the number of different aspect categories in the collection of the sentences in the whole dataset. In the y axis we can see the number of aspect categories included in a sentence and in the x axis the number of the sentences.

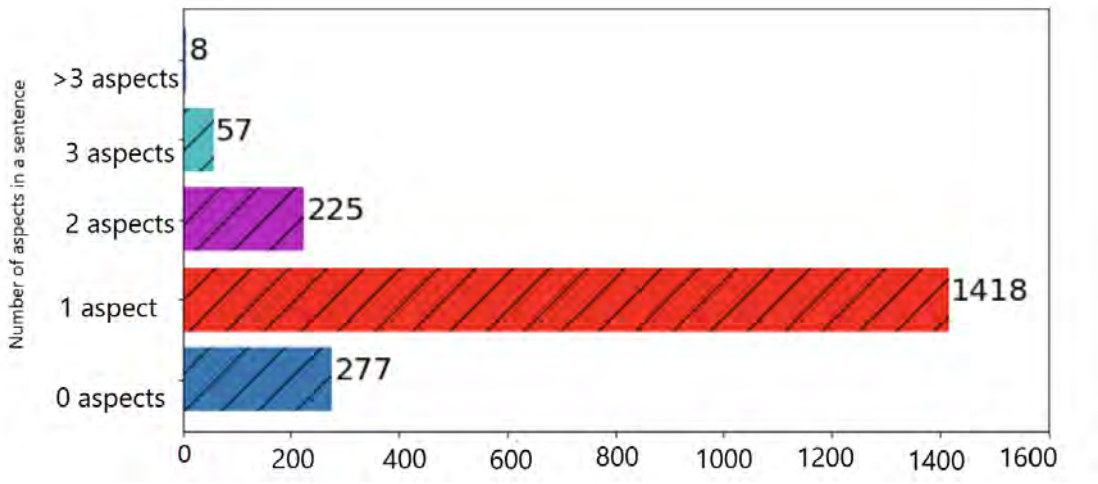


Figure 3.1: Number of different aspects in the sentences

From the dataset we removed the sentences which did not refer to any of the aspect categories and did not include polarity about a topic, which were the same. We can see some examples of such sentences in the following table.

Table 3.2: No Aspect/Polarity Sentences

Sentence with no Aspect-Polarity
I went to this restaurant with a woman that I met recently
We went to Ino for Valentines.
We've lived in the area for more than 8 years.

In addition, in Figure 3.2 we can see the distribution of the polarity in the sentences in the corpus.

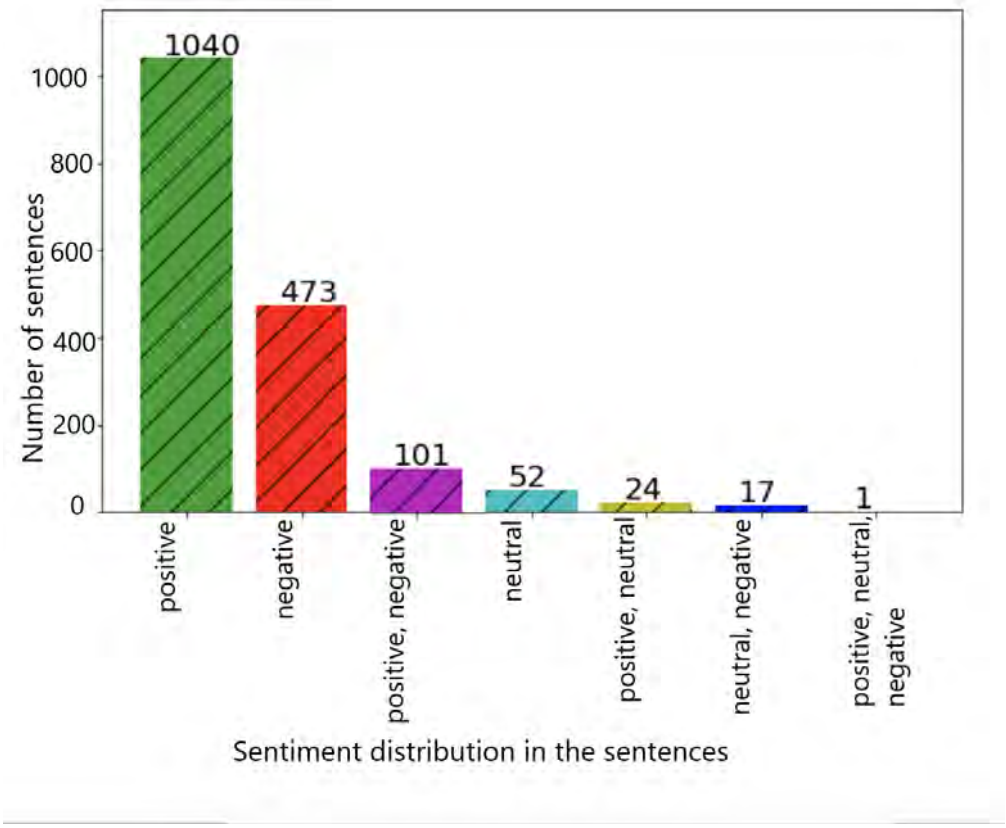


Figure 3.2: Sentiment distribution over the sentences in the corpus

The last piece of information in the dataset that we are going analyze is the distribution of the topics in the dataset. As we mentioned before the number of the topics in the sentences are six and these are Restaurant, Ambience, Location, Service, Food and Drinks. The distribution of these topics in the dataset is not balanced. To be more precise three of these categories are underrepresented as shown in the Figurebelow.

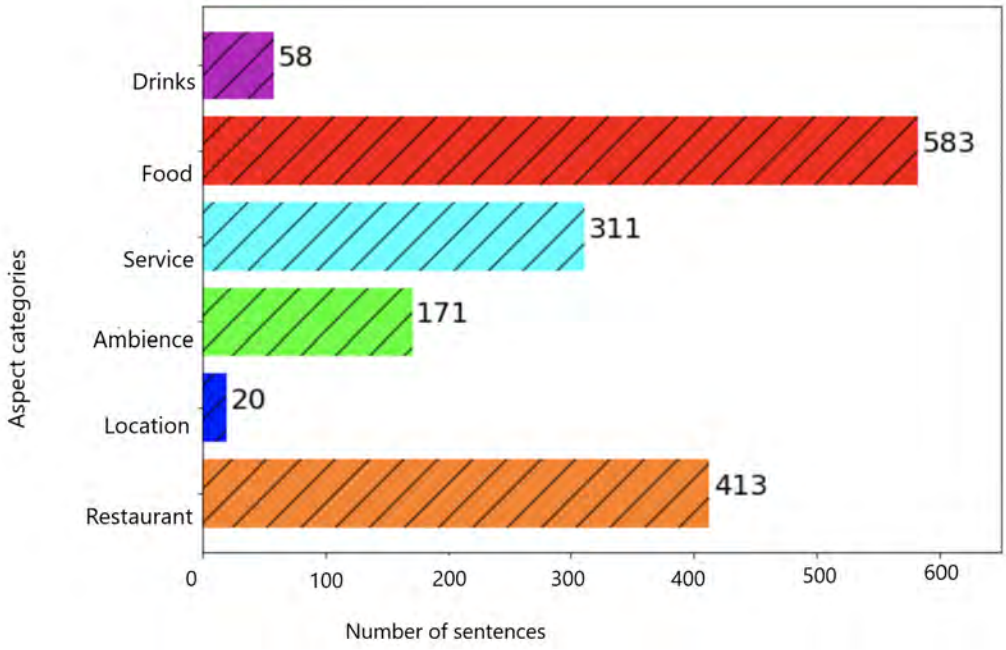


Figure 3.3: Number of sentences per aspect category

3.2.2 Laptop domain

The structure of the dataset is the same with the one mentioned above for Restaurants. This dataset has a few more aspect categories: Shipping, Laptop, Support, Company, OS, Warranty, Software and Hardware. The initial dataset contained more categories with very few sentences each, thus we decided as the guides from the competition suggested, to merge these categories to one which is Hardware. The initial categories were: Display, CPU, Motherboard, Hard disk, Memory, Battery, Keyboard, Mouse, f Fast and cooling, Optical devices and Multimedia devices.

In the table below we can see some examples of the dataset:

Table 3.3: Example of the Laptop dataset

Sentence	**Aspect Category**	**Aspect Polarity**
This computer is absolutely AMAZING!!!	Laptop	positive
The keyboard has a wonderful nature feel	Hardware	positive
Yet, HP won't do anything about the problem	Support	negative
Dell is a decently made pc	Laptop	neutral

The following Figureshows the number of different aspect categories included in the collection of the sentences. We can see that there are no sentences which do not include at least one aspect category like in the Restaurant domain dataset.

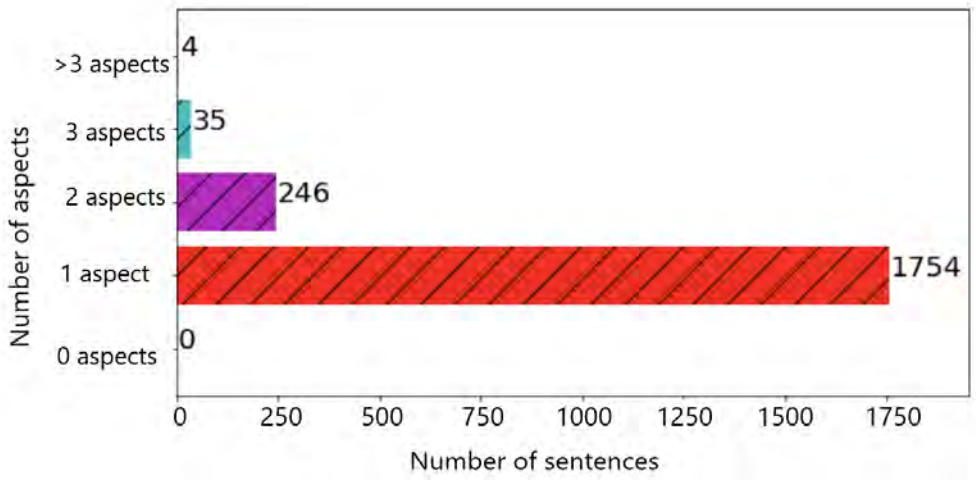


Figure 3.4: Number of different aspects in the sentences

Additionally, we can see the distribution of the opinions in the corpus in Figure 3.5 and the number of sentences for each aspect category in 3.6

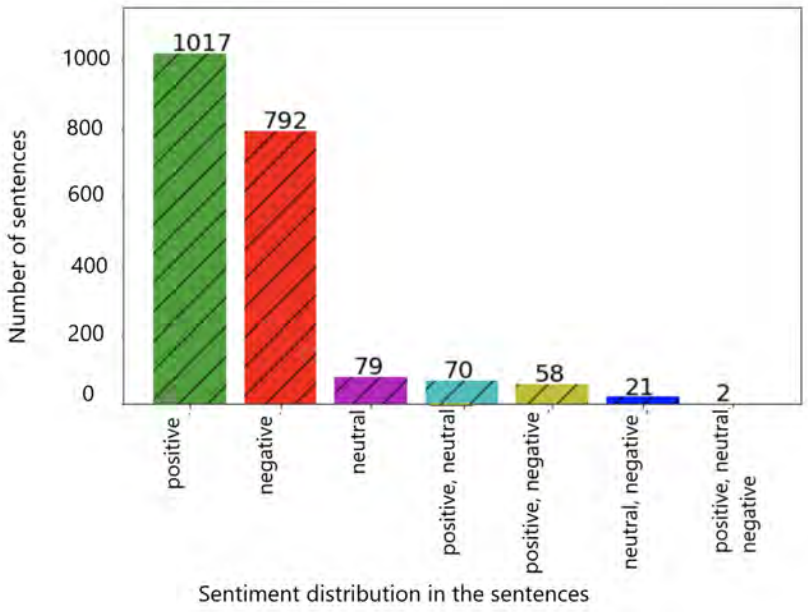


Figure 3.5: Sentiment distribution over the sentences in the corpus

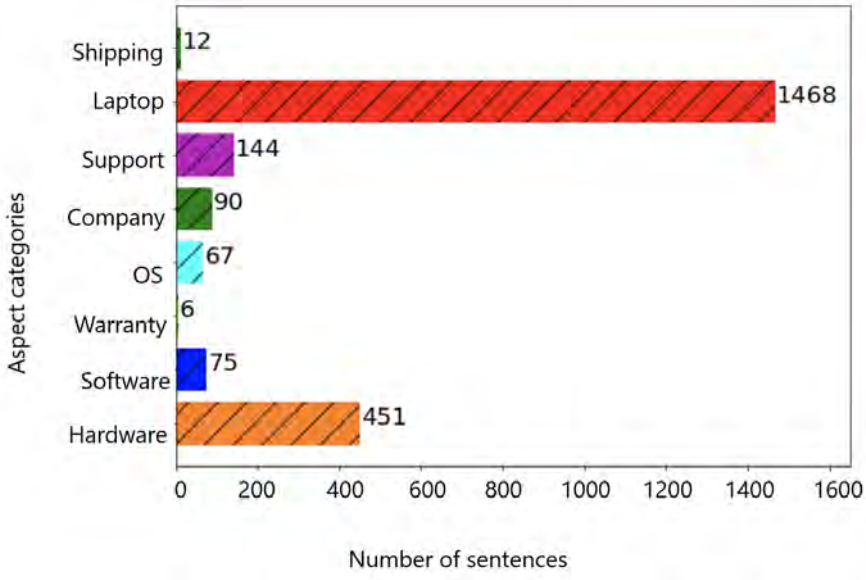


Figure 3.6: Number of sentences per aspect category

3.3 Data augmentation

It is important to mention that the size of the dataset is not as big as we wanted. That is the reason why we decided to do data augmentation on the training set. Data augmentation is the procedure in which we create copies of the sentences of the original dataset and alter them a little. There are different ways to perform that task but we chose the following. For every sentence of the dataset we identified the nouns using from the natural language toolkit (NLTK) the part of speech tagger which classifies each word according to its class e.g. verb, noun, adjective etc. and assigns labels to them. We also used another NLTK tool, a lexical database for the English language called WordNet, which contains synonyms for the vast majority of English words. For every identified noun we created a list with the synonyms provided by WordNet and randomly chose from this list so as to replace some nouns and create four copies for each sentence. That way we keep the initial sentence as is and add the copies in the dataset. There are other methods used for data augmentation in natural language processing, such as random insertion of a synonym of a randomly chosen word from the sentence or randomly deleting a word from a sentence, but we did not want to interfere with the main content of each sentence by e.g. deleting a noun or removing a word that would be important for defining the category or the sentiment of a sentence.

Not all the synonyms are suitable for the content of a sentence, however the majority is able to accurately reflect the meaning of the sentence. Furthermore, the length of the list of the synonyms for each noun is not the same and that is the reason why we did not replace each sentence with all its synonyms but we chose four at random in order to keep the same distribution of the topics and their sentiment in the dataset. The following table illustrates examples of data augmentation for both domains.

Table 3.4: Augmentation example in Restaurant domain

Initial Sentence
I recently went to this **restaurant** with some **co-workers** for lunch and had an amazing time.
Augmented Sentences
I recently went to this **eating house** with some co-workers for lunch and had an amazing time.
I recently went to this **eatery** with some co-workers for lunch and had an amazing time.
I recently went to this Restaurant with some **workfellows** for lunch and had an amazing time.
I recently went to this Restaurant with some **colleague** for lunch and had an amazing time.

Table 3.5: Augmentation example in Laptop domain

Initial Sentence
Small **screen** somewhat limiting but great for **travel**.
Augmented Sentences
Small **projection screen** somewhat limiting but great for travel.
Small **shield** somewhat limiting but great for travel .
Small screen somewhat limiting but great for **journey**.
Small screen somewhat limiting but great for **trip** .

3.4 Preprocessing pipeline

The majority of the reviews are short texts which most of the times are not well written. In order to have a clean text as an input for our model we apply some preprocessing steps, which are the following.

- Remove the white space in the beginning and end of each sentence.
- Convert all letters to lowercase
- Remove punctuation and special characters
- Remove numbers
- Remove stop words

As an exception to the above, when using the RoBERTa model, the preprocessing steps are skipped, since the pre-trained model used has been trained on raw corpus as mentioned in [43].

4. Word Embeddings

In this chapter we are going to describe the word embeddings that we used in order to transform the text data into numeric representations for the models. The machine and deep learning algorithms are not able to "read" the data the way that humans do, in physical language. That is the reason why we need to represent the text in a form that the model is able to read and process. The way to do this is by representing the words in the corpus with vectors which include numbers and use them as input for the models. Word embeddings are such techniques that are used in information retrieval and natural language processing for word representation. In order to do that in the beginning we used two simple methods that calculate the frequency of the occurrence of the words in the text. Moreover, we used Global Vectors (GloVe) that capture the semantic relationships between words as described in [35]. Finally, while researching the topic of aspect based sentiment analysis we came across am embedding that has been created aiming to be used for this problem in [59] and used that as well.

Usually, the first two methods are combined with machine learning models and the rest with deep learning models. However, we tried an unusual combination of a simple representation for the deep learning model and the more complex for a machine learning model that was able to handle such input. The results will be discussed in the specific section.

4.1 Count Vectorizer and Tf-Idf

The most usual and simple method to create word embedding is to represent each sentence with a vector with dimension the size of the vocabulary of the whole cor-

pus. If a word w occurs in the sentence the value at the position of the word in the vector is set to 1 otherwise it is set to 0. If the same word occurs again in the text then the value increases accordingly. This technique is known as Bag of Words (BoW) it is able to capture the frequency of the words that occur in a sentence. The resulting values for each word represent how many times each word appears in the text. However, most of the times it is not only useful to know the frequency of each word in a document but also the importance of a word in the text. While BoW weights all the words based only on how many times they occur there is another method that takes into account the context in which this word appears. This technique is called term frequency–inverse document frequency (Tf-Idf) and reflects the importance of each word in a sentence in the collection of the sentences. Tf-Idf weights words not only based on their frequency in a sentence but also in correlation with the collection of the other sentences in the whole dataset. First it calculates how often a word occurs in the collection of all the sentences in the dataset. In this step all words are considered to be equally important for the content of the text.

Term frequency:

$$\text{TF} = \frac{number\ of\ times\ a\ word\ appears\ in\ the\ collection\ of\ all\ the\ sentences}{total\ number\ of\ words\ in\ the\ collection\ of\ all\ the\ sentences}.$$

Then, taking into account that many words such as prepositions are more likely to appear many times in a corpus and carry no information about the content of the text, it calculates how important each word is.
Inverse Document Frequency:

$$\text{IDF}= \log \frac{Total\ number\ of\ the\ collection\ of\ the\ sentences}{Number\ of\ sentences\ with\ word\ w\ in\ it}$$

The weight of each word w contained in the collection of all the sentences in the dataset, e.g. the corpus S, is calculated by the following formula:

$$w_{x,y} = tf_{x,y}\ \text{x}\ log\left(\frac{S}{df_x}\right) \tag{4.1}$$

where

$$tf_{x,y} \tag{4.2}$$

is the frequency of the word x which appears in sentence y and

$$df_x \tag{4.3}$$

is the number of documents which contain the word x.

To summarize, words with very high frequency get lower weights because they are less likely to carry important information for the model while words that appear with lower frequency in the text might be more important for the context of a sentence and thus get higher values.

4.2 Global vectors for word representation

GloVe as presented in [36] is a way to represent the sentences and give them as input to the model. Each word in the corpus is represented with a vector. GloVe creates these vectors by counting each words' co-occurrence. By using these statistics, GloVe minimizes the least square error (LSE) and tries to distinguish the relevant words. The main difference between GloVe and other word embeddings, which is an advantage of it, is that it does not only take into account simple co-occurrence probabilities but combines them with the ratio each word appears as reported in [44]. Each word is represented with an embedding with size 300 as defined by GloVe.

4.3 Domain specific embedding

In the paper [59] a different word embedding is proposed for experimentation. The writers created a specific domain embedding for Restaurants and Laptops. In order to create this specific embedding for Laptops they gathered all the Laptop reviews from the Amazon review dataset provided by [**Laptopembed**] and for the Restaurants from the Yelp review dataset challenge (which is not available any

more). They trained the embedding using fastText, a different word embeddings method which takes into account the characters of each word, with the default hyper-parameters of the embedding model. The dimension of the embedding for both the categories is 100.

5. Methodology

In this chapter we are going to discuss the methods that we used to extract the aspect categories and perform sentiment analysis. Some of the methods were used only for one of the two tasks and the rest of them for both. The baseline method for aspect extraction is Latent Dirichlet allocation (LDA) model suggested in [6]. Then we used two modifications of this model, Seeded LDA that is presented in [20] and Biterm topic model described in [61]. Moreover, we used multi-layer perceptron (MLP) neural network and Support vector machine (SVM). Finally, we used a bidirectional LSTM model for the same purpose and the state-of-the-art model RoBERTa reported in [25]. For the second task, polarity extraction, we used as a baseline a naive Bayes model. Then we implemented random forest, MLP neural network and Support vector machine. We also used the bidirectional LSTM and RoBERTa to extract the opinion polarity of the sentences. All the models mentioned above will be described in this chapter.

5.1 Latent Dirichlet allocation

Latent Dirichlet allocation [6] is an unsupervised generative probabilistic model widely used for text data and specifically for topic modelling. In our dataset each sentence refers to one or more topics thus each sentence can be considered a combination of the topics with probability p for each of them. The objective of the LDA is to describe each sentence with a distribution of topics and then in turn with a distribution of a set of words that are most likely to refer to the topics that the sentence contains. As we can see in the following illustration, LDA performs a generative process in levels, document, topic and sentence. The plates represent the levels in which LDA performs. The exterior plate illustrates the first level, the

documents whilst the inner plate illustrates the repeated procedure of choosing topics and words within the sentences.

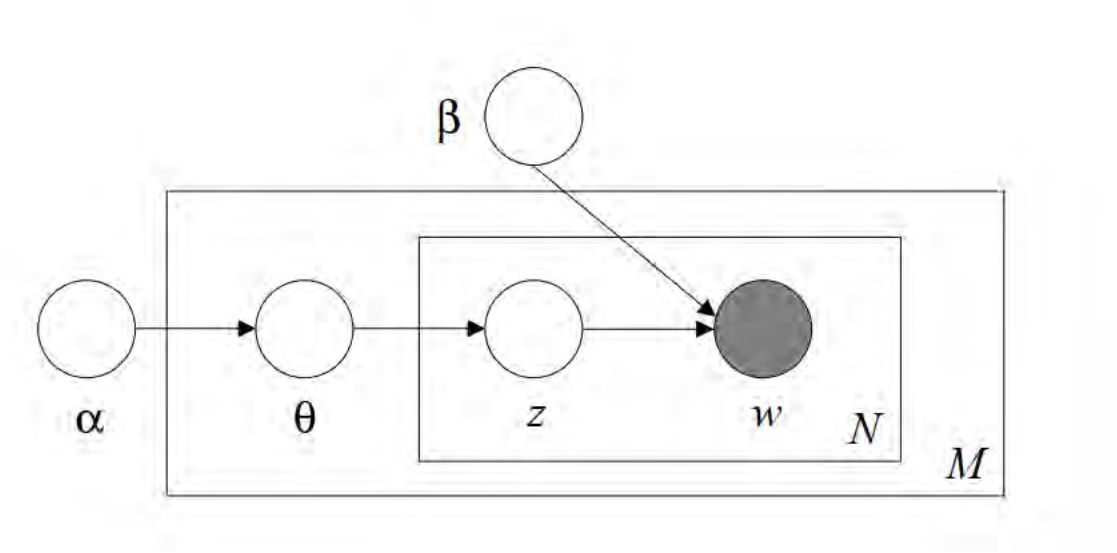


Figure 5.1: LDA plated schema [6]

The nodes represent random variables and the edges show possible dependencies among the variables. The notations M and N are used for counting the number of the sentences in a corpus and the number of words in each of them, respectively. The parameter α is refereed as Dirichlet parameter and represents the topic distribution per sentence for all the sentences in the corpus, parameter β refers to topics and specifically the distribution of words per topic as they explained in [33]. Both these parameters are sampled once while generating the corpus. The variable θ is a document level variable and refers to the distribution of the topics in each sentence/document and it is sampled once per document. The variables z and w are the topic of each word in a document and the word respectively and both of them are word-level variables and sampled once for each word in every document. Moreover, N is the number of words in a document and does not depend on any of the aforementioned variables. The probability density function of the random variable θ is calculated by the following equation explained in [8]:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} a_i)}{\prod_{i=1}^{k} \Gamma(a_i)} \theta_1^{a_1-1} .... \theta_k^{a_k-1} \tag{5.1}$$

where Γ is the Gamma function and parameter α for every index is is greater than 0.

The joint distribution of a topic mixture θ is described with the following equation and the following formulas are presented in [6]:

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \tag{5.2}$$

where parameters α and β are given, N set of topics $\mathbf{z}$ and words $\mathbf{w}$ and z is the topic of each word in the document and w the word itself.

Taking the integral for θ and the summation over z of 5.2 the marginal distribution of a document/sentence occurs with the formula:

$$p(\mathbf{w}|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \right) d\theta \tag{5.3}$$

In the last step the probability of the whole corpus is calculated by taking the product of all the marginal probabilities of every single document.

$$p(\mathbf{D}|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn}, \beta) \right) d\theta_d \tag{5.4}$$

Despite the fact that LDA seemed to be a good candidate model for assigning topic categories to each sentence, there are some limitations that the model had to handle. LDA is a generative probabilistic model which examines each sentence separately in order to identify the patterns that are hidden in the data. The dataset consists of short review sentences which do not provide enough information to the model compared to a longer text according to [61]. Another disadvantage of LDA is that it produces unlabeled topic categories. Although we are able to determine the number of the topics which LDA will assign the sentences to, due to the fact that we have the categories in the dataset, the correspondence of the generated topics to the actual categories is not clear in the end.

5.2 Seeded Latent Dirichlet allocation

Seeded LDA is a semi supervised version of the probabilistic aforementioned Latent Dirichlet allocation. As mentioned in [20] the model is enriched with prior knowledge by giving additional information about the corpus and the topics with seed words for the aspect categories that are underrepresented in the corpus. In the section with the data statistics we saw that these categories are Location, Ambience and Drinks in the Restaurant domain and Shipping, Software, Warranty, OS, Company and Support in the Laptop domain. It is important to mention that the model is able to chose whether it will take into account these words or not. The model is able to use seed words in two different ways: to chose the words that represent each topic and also assign the topics to the sentences. It uses the seed words to improve the words that the model assigns to each topic. The LDA model assigns words to each topic using the multinomial distribution. Seeded LDA uses a combination of the words generated by the regular LDA for each topic and the words from each seed set. The words in the seed lists as well as the words in the lists that the regular model generates for the topics have non-uniform probability distribution and that means that the words do not have equal probability of being chosen by the model. Seeded LDA takes as input the number of topics of our problem and the seed words for the classes that are under-represented, which are mentioned below. The seed list should be representative of each topic. We collected the words of all the sentences that belong to the under-represented aspect categories and created lists with the most frequent adjectives and nouns. In the following table we can see some examples of these words in both domains.

Table 5.1: Seed Words in Restaurant domain

Location	**Ambience**	**Drinks**
Location	decor	wine
view	Ambience	list
river	beautiful	beer
neighborhood	staff	bottle
avenue	night	horrible
room	small	tasty
good	warm	glass

Table 5.2: Seed Words in Laptop domain

Shipping	**Software**	**Warranty**	**OS**	**Company**	**Support**
days	Software	Warranty	Linux	HP	Apple
weeks	program	manufacturer	Windows	Toshiba	Support
delivery	computer	replacement	OS	Dell	way
paperwork	addition	repair	system	Company	hours
priority	word	problem	Starter	Lenovo	Service
mail	problem	purchase	Mac	reputation	calls
UPS	files	drop	Vista	acer	day

5.3 Biterm topic model

Classifying short texts might be more challenging than having longer texts because of their characteristics. The length of the text is the main problem that algorithms have to overcome. The number of the words used to describe the context of each review are very few and the semantic relations between them are difficult to be identified as stated in [52]. Biterm topic model tries to capture the co-occurrence of words in each topic category and improve classification. Moreover, it uses the identified relationships to deal with the sparse nature of the reviews on the whole

corpus.

Biterm topic model discovers the co-occurrence patterns of the words by collecting information from the whole corpus, instead of a single sentence. The model represents the topics by a set of words that are correlated whilst this correlation has occurred by examining the complete collection of the sentences. The term Biterm refers to an unordered tuple of words that show a pattern in the corpus. The model follows a generative process in which for each topic z picks a word with a specific distribution ϕ for the specific topic. The next step is to determine the distribution for the whole collection of words for each topic. Finally, a topic class is assigned to each Biterm tuple. Thus, the joint probability of every tuple can be described with the following equation:

$$P(\beta) = \sum_{z} P(z)P(w_i|z)P(z)P(w_j|z) \tag{5.5}$$

where $\beta = (w_i, w_j)$ is the Biterm tuple, w_i and w_j are the words in the tuple and z is the distribution of the topics as shown in [61].
Comparing the two models LDA and BTM we can identify that the two main differences are the following:

- LDA considers each word separately in order to assign it to one or more classes while BTM models the co-occurrence of a sequence of two words e.g bi-gram.
- Biterms are created by examining the whole corpus instead of each sentence.

5.4 Neural networks

"*Neural Networks are algorithms that simulate the learning procedure of living organisms*" as described in [1]. The most simple neural network is the one that has one layer and is also known as perceptron. The perceptron is able to directly map the input to an output by using an activation function, calculating weights and

including bias. The following image shows a the basic architecture of a perceptron which includes input nodes x, weights w, an output node which includes the activation function ϕ, an extra neuron for bias b_k and the output y. The perceptron calculates the weights and the bias based on the given data.

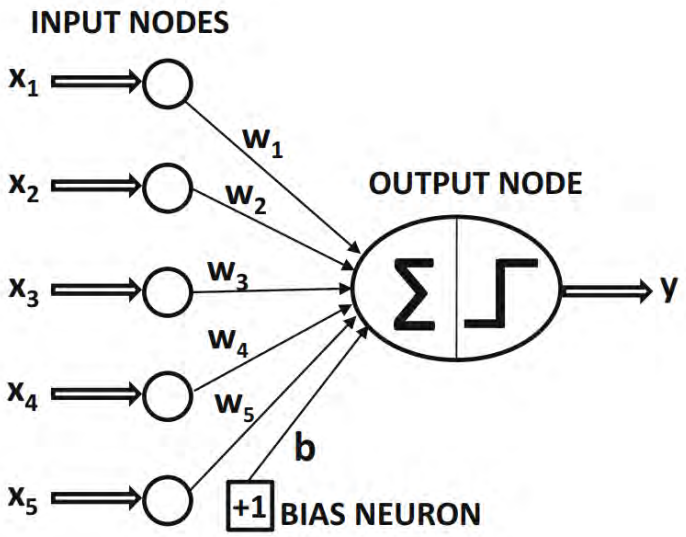


Figure 5.2: Perceptron architecture [1]

The procedure that the perceptron uses can be described with the following equation:

$$y = \phi\left(\sum_{j=1}^{m} w_{kl}x_j + b_k\right) \tag{5.6}$$

A multi layer neural network is based on the idea of a perceptron with more computation layers. These additional layers are called hidden layers and the neural networks is considered a feed forward network because successive layers feed into one another in the forward direction from input as described in [1]. The input layer gets as input the data and passes them to the hidden layers and the output of each hidden layer is fed to next layer until it reaches the output layer in which the final output is computed.

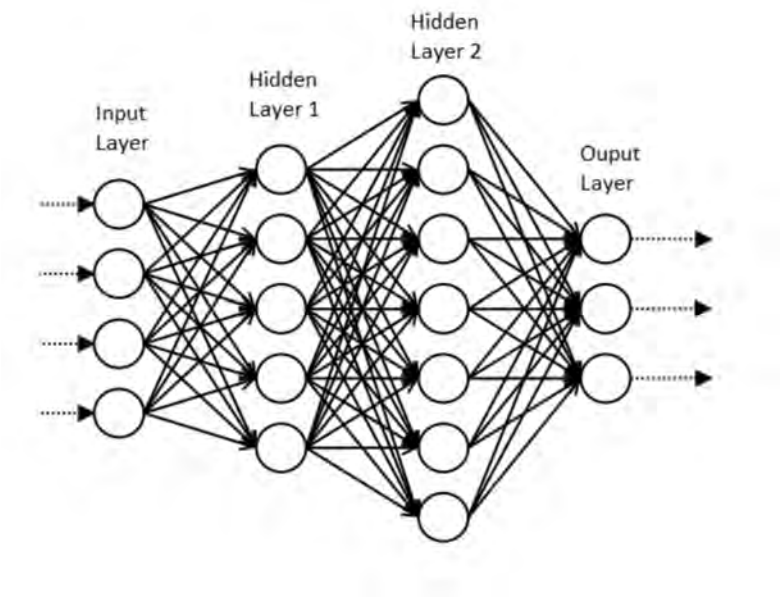


Figure 5.3: Multi layer neural network [2]

The architecture of a neural network may vary based on different number of hyper parameters that can be used. The number and the depth of the hidden layers, the variety of activation functions that can be used, the learning rate of the network, the batch size and many others are the hyper parameters of each neural network and define its performance. Finding the optimal hyper parameters for a neural network can be a challenging and time consuming procedure.
There is a variety of different activation functions that are used according to the nature of each problem. Three activation functions that we are going to use in our models are *softmax*, *sigmoid* and $ReLU$.

The *softmax* activation function turns the output to probabilities between 0 and 1. It calculates the exponent of each output and divides it with the summation of the exponents of each class. The resulted probabilities of all the classes should sum to 1. Most of the times it is used for multi-class classification problems.

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{5.7}$$

The *sigmoid* activation function, also called *Logistic*, maps the output values between 0 and 1 for each class, thus it is used for multi-label classification.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5.8}$$

The *ReLU* (Rectified Linear Unit) activation function gives as result either the input if its value is positive or 0 if negative.

$$f(x) = x^{+} = max(0, x) \tag{5.9}$$

5.4.1 Multi-layer perceptron

The first deep learning method that we chose to use is an multi-layer perceptron (MLP) which is a basic feed forward neural network. The MLP has an input layer, an output layer and at least one hidden layer. There are various activation functions which can be linear or not and we will use the aforementioned. There are also multiple other parameters such as the solver that is used or the number of hidden layers, however we are going to discuss about the parameters that we used in the implementation section.

5.5 Support vector machine

The first classifier we chose is based on the idea of Support vector machines which are well known supervised classification algorithms. The data in the model are represented as vectors with numbers which represents the coordinates of each point in the n dimensional space as it is presented in [11]. The Support vector classifier creates a set of hyper-planes to distinguish the different aspect classes while maximising the margin between them as stated in [5], proportional to all the classes. To gain an insight on how Support vectors work, we should mention that they are the points that lie on the hyper planes and affect the slope and the position of the

optimal hyper plane that will distinguish the classes. They are used as guides in creating the hyper planes having as a goal to achieve the maximum margin for all the classes. In the following representation we can see how Support vectors work for two classes. They create two planes, one for each class that guide the creation of the central plane that will distinguish the classes based on the points that lie on the side planes and has the largest possible distance to the nearest points.

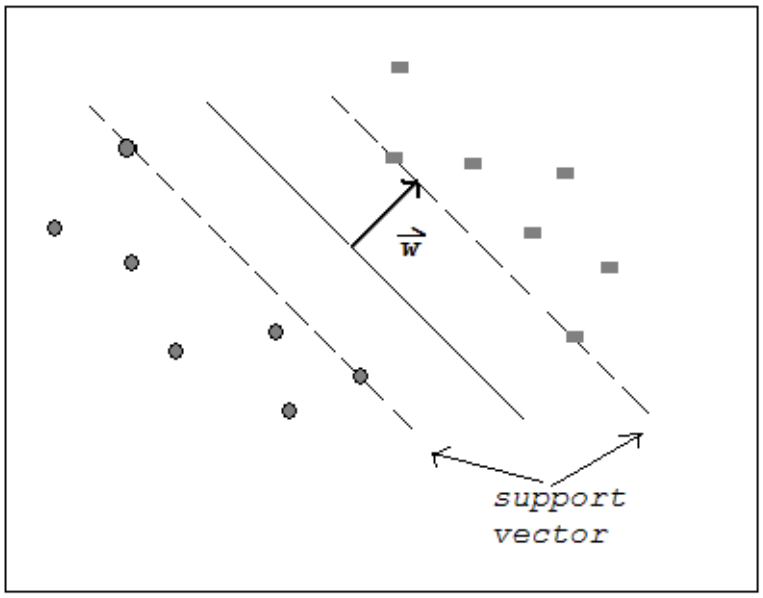


Figure 5.4: Support vector classifier [5]

In our Support vector algorithm we use a linear kernel which aims to solve the following primal problem:

$$min_{w,b}\frac{1}{2}w^T w + C\sum_{i=1} max(0, y_i(w^T\phi(x_i) + b)) \tag{5.10}$$

The best case possible can happen when the classes do not overlap and the Support vectors are able to create hyper planes that do not overlap. However, most of the times this is not possible to happen with data that come from the real world. This is the reason why Support vector machines, while trying to maximize the margin, simultaneously try to minimise the misclassification of the sentences. The parameter C is responsible for letting points to be excluded from a class but

simultaneously giving penalties to these points.

Support vector classifiers have been created for binary classification problems, so in order to be able to make one suitable for our problem, which is a multi-label problem, we needed to wrap it in one-versus-rest classifier. That means for each aspect class we create one Support vector classifier that works against all the other categories and decides whether a given sentence belongs to the aspect category or not, created by [34]. Although we used Support vector classifier for sentiment analysis as well, we did not use one-versus-rest classifier for performing this task. This alteration on the algorithm implementation happens because during aspect categorization one sentence can belong to more than one aspect categories while during sentiment analysis the outcome is either positive, negative or neutral.

5.6 Naive Bayes

Naive Bayes algorithms are a group of algorithms that are used to solve classification problems. They use Bayes theorem to calculate the probabilities of each class and chooses the one with the highest probability. The are widely used in many different domains and particularly for Natural Language Processing. Despite the fact that they are simple and we could say "naive" algorithm, they are quite popular because they are fast and reliable. Moreover, they are suitable for our problem, because they perform well with multi-class problems such as sentiment analysis with three different classes.

5.7 Random forest

A popular machine learning approach is random forest which consists of a collection of decision trees that create an ensemble. Decision trees iteratively slice the dataset to smaller subsets based on the value of a feature. The trees make the splits in order to achieve the highest possible information gain. A random forest consists of many decision trees. Each tree outputs a class prediction and the one with the

most votes becomes the model's prediction, as explained in [54].

5.8 Bidirectional long short-term memory

Long short term memory network is a modified approach of recurrent neural networks which is able to cope with keeping information from earlier stages of the model in contrast to simple RNNs that suffer from "short memory", as stated in [18]. LSTM is commonly used for text classification problems like ours, in which important information can be forgotten when they appear in the beginning. The reason for choosing LSTM instead of a RNN is because they are able to decide which information are useful for the model and retain them and which should be dismissed. LSTM contains memory blocks in every hidden layer. The memory blocks contain memory units that are called cells and gates that control the flux of the information as described in [45]. Each memory block consists of the input gate i_t, the output gate o_t and the forget gate f_t. The functions that describe each layer examine the input of that state x_t and the output of the previous hidden state h_{t-1} in combination with learned weights and various activation functions as the author explains in [44].

The following functions define the model of a LSTM cell:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{5.11}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{5.12}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{5.13}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ tahn(W_c x_t + U_c h_{t-1} + b_c) \tag{5.14}$$

$$h_t = o_t \circ tahn(c_t) \tag{5.15}$$

where σ is the sigmoid activation function, tahn the tahn activation function, $\circ$ is the element wise multiplication (Hadamard product) as mentioned in [44], W and U the weights and b different biases.

In Figure 5.5 a LSTM cell is illustrated:

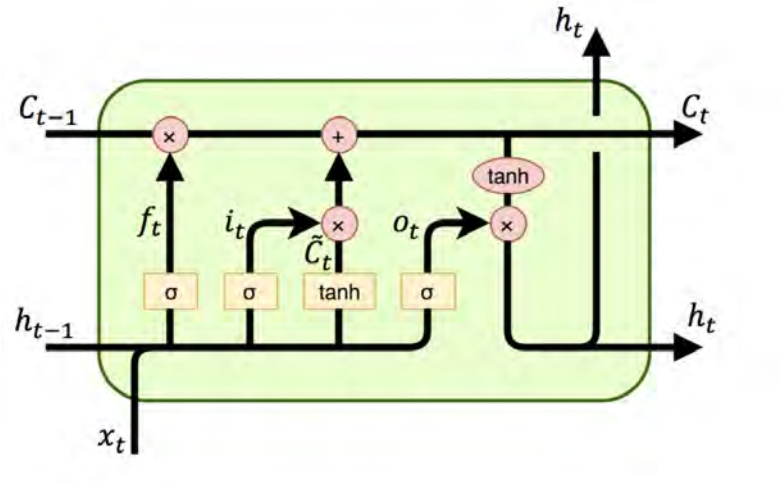


Figure 5.5: Long Short Term Memory Cell [27]

In order to boost the performance of the LSTM model, we used two LSTM networks and trained them on the input data. Each LSTM cell reads the data in a forward order e.g. take the sentence as it is and in the backward e.g. reversed copy of each sentence. The aim of this extension from a simple LSTM network to a bidirectional LSTM as seen in Figure 5.6 is to provide extra information to the model and help it learn faster and potentially better.

Finally, in order to avoid over-fitting of our model we added dropout layers. Dropout is a regularization technique that sets the output of randomly chosen hidden layers to 0.

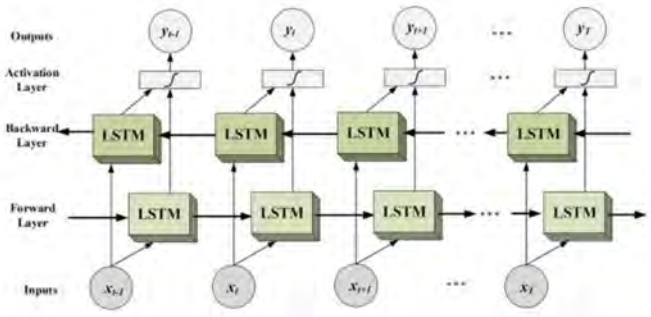


Figure 5.6: Bidirectional LSTM [12]

5.9 Attention models

Attention model was proposed in [4] as a solution to a machine learning translation implementation. The majority of the models which were trying to solve this problem are based on the idea of encoders-decoders. They try to convert an input sequence that is encoded to a vector with a fixed length to the target output which is produced by the decoder. The issue with such type of models is whether the model is able to include all the necessary information of the text in a vector which has fixed size or not. Moreover, the aforementioned issue becomes more noticeable when the model has to cope with longer sequences e.g. longer sentences as shown in [10].

Attention model was created to solve the aforementioned issue. The main difference between the encoder-decoder models and attention model is that the later is not trying to capture the whole sentence into the fixed-length matrix. Instead, it *"encodes the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively while decoding the translation"* as described in [4]. The input vector with the fixed size is called context vector. The context vector captures the information of the text and then it is transformed to the output from the decoder. This procedure can be seen in the following illustration of the model.

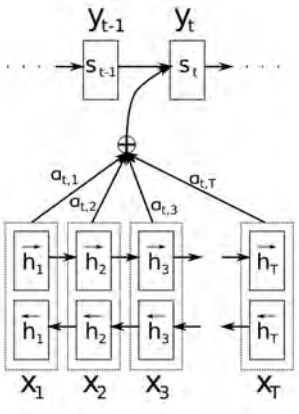


Figure 5.7: Illustration of the Attention model [4]

The encoder of the attention model is a bidirectional Recurrent Neural Notwork and that means that the model reads the input sequential data $\mathbf{x} = [x_1, x_2.....x_n]$ with length n in the usual forward order and reverse e.g backwards, as we can also see in the representation above. The forward hidden layer is represented with $\overrightarrow{\mathrm{h_i}}$ and the backward with $\overleftarrow{\mathrm{h_i}}$. The inputs of each hidden state are: the output and the hidden state of the previous decoder and the context vector c_i in the given time step. The context vector is not the same in every time step, it is calculated based on the a_{ij} which are the attention weights and are calculated by the following equation, taking into account the alignment of the the input word in the position i and the output word in the position j:

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{n=1}^{T} exp(e_{in})} \tag{5.16}$$

where

$$e_{ij} = \alpha(s_{i-1}, h_j) \tag{5.17}$$

is the alignment which is a feed forwards neural network with one hidden layer and s_i is the hidden state of the decoder. The characteristics of the attention models have made them very popular not only with translation tasks but also for other natural language processing tasks and image recognition. The are able to investigate and decide which information are important like human can do and overcome the efficiency of typical sequence to sequence models.

5.10 Transformers Model

Sequence-to-sequence architectures have been created to convert a sequence of words to another sequence. They were initially used for text translation from a language to another. The Transformers model presented in [55] has been built up on the attention mechanism. It is created based on the idea of using encoder and decoder aiming to transform the input sequence to an output sequence.

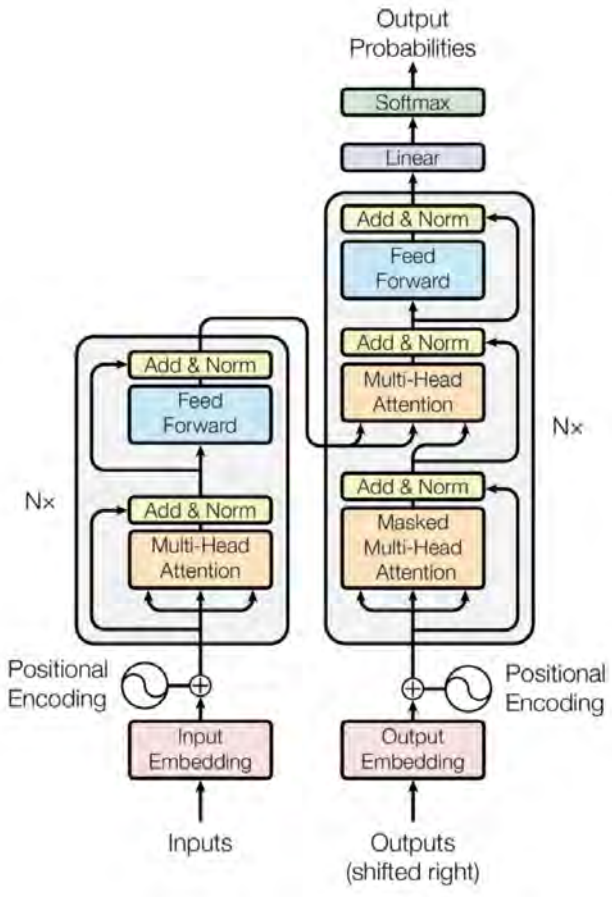


Figure 5.8: Transformers model [29]

The encoder is illustrated on the left side of Figure 5.8 and the decoder on the right and they consist of stacked modules, e.g. one on the top of the other, which

mainly are multi-head attention mechanism and a fully connected feed-forward neural networks, as the authors explain in [55].

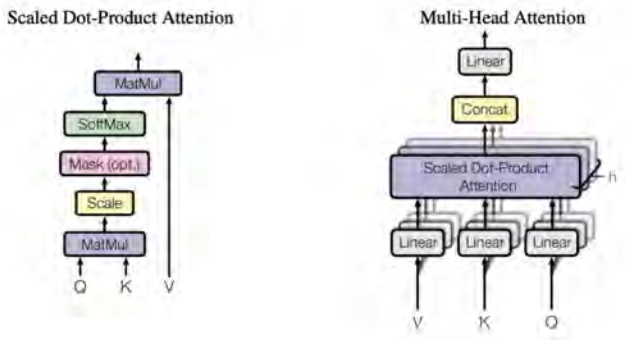


Figure 5.9: Scaled Dot-Product Attention (left) / Multi-Head Attention (right) [29]

The multi-head attention function can be described as a set of queries, keys and values that are used for describing the scaled dot-product attention as well. The scaled dot-product attention is calculated using the following equation:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{5.18}$$

where Q, K and V are the matrices of the queries, keys and values and $\frac{1}{\sqrt{d_k}}$ is the scaling factor to mainly handle the large values of d_k.
The multi-head attention uses parallel functions of queries, keys and values and the results are concatenated and projecting giving the final result. The function of this attention is calculated below:

$$Multi - HeadAttention(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{5.19}$$

where

$$head_1 = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{5.20}$$

where the projections are parameters matrices $W_i^Q \in^{d_{model} \times d_k}$, $W_i^K \in^{d_{model} \times d_k}$, $W_i^V \in^{d_{model} \times d_u}$, $W^O \in^{hd_u \times d_{model}}$.

5.10.1 RoBERTa

BERT (bidirectional encoder representations from transformers) as presented in [53] is a pre-trained language model used for natural language processing implementations. RoBERTa [37] is a robustly optimized approach of BERT pre-trained model. Specifically, the changes that have been implemented to the model the following: the model has been trained longer, on more data, the next sentence prediction objective has been removed, the model has been trained on longer sequences and finally the masking pattern has been changed. In order to explain more these modifications, we will give some information first for the BERT model.

First, we should mention that the BERT model is trained on book corpus which contains 800M words and english Wikipedia 2500M words. The input of BERT consists of three embeddings: token, segment and positional embeddings. The sentences are tokenized to words and in the beginning and in the end of each sentence two additional tokens are joined, [CLS] in the beginning and [SEP] in the end.

During training, BERT uses masked language model which randomly chooses a set of tokens and replaces them with with [MASK] tokens. The objective of the model is to predict the replaced tokens. The ration of replacing tokens is the following: the model selects 15% of the tokens and 80% of them are replaced with the token [MASK]. Afterwards, 10% of them are replaced with a random token and 10% stays the same. This procedure is performed in the beginning of the training and it is saved for the rest of it.

After, next sentence prediction (NSP) is used to define the relationship between segments of the input sentences. There are positive examples which are collected by sentences that are sequential and negative which are defined by matching sentences from different documents.

Having explained BERT model, we will now discuss the modifications that implemented in [37]. In the mask level, they used dynamic masking in which they

generated the masking pattern whenever a sequence is fed to the model. Moreover, removing the NSP gives the same or slightly better results than the original BERT. Another modification is that using large mini-batches for training can improve the performance of the model. Finally, the text is encoded with byte-pair encoder which is uses bytes rather than unicodes. RoBERTa model has been proven to give better results than the original BERT. We did not train RoBERTa from scratch. The models has approximately 355 million parameters and it is trained on a very big corpus. We used the pre-trained weights from the RoBERTa LARGE provided by [58].

6. Experimental Structure

In this chapter we are going to examine the structure of the experiments. First we are going to describe the aspect extraction and afterwards the polarity detection.

6.1 Aspect extraction

Aspect extraction pipeline contains different types of models. Each model uses different word representations and contain different layers which we are going to describe in this section.

In Figure 6.2 the aspect pipeline is illustrated.

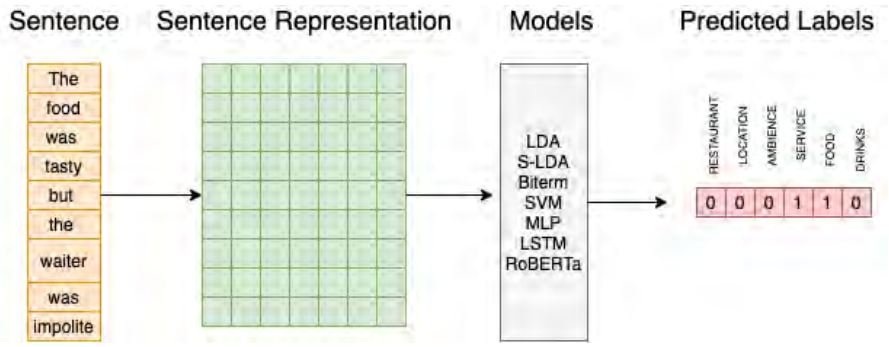


Figure 6.1: Aspect categorization pipeline

The models LDA, Seeded LDA, Biterm, SVM, MLP take as input the sentences as numeric representations using Tf-Idf or Count Vectorizer. Furthermore, in the SVM and MLP we also tried to add as features the probability vectors predicted by the best probabilistic generative model, seeded LDA, in order to boost their

performance. LSTM uses word embeddings created as described in sections 4.2 and 4.3. Moreover, we used the pre-trained model RoBERTa as an input to the LSTM and a simple and unusual word representation for the LSTM, Tf-Idf. The predicted output for each sentence is a vector with dimension 6 for Restaurant domain and 8 for Laptop domain which contains 0s when the aspect category is not predicted and 1s when the category is predicted. We can see the aforementioned combinations of word embeddings and the models in the following table:

Table 6.1: Word embeddings and models for aspect categorization

Word Representation						
		Count Vectorizer	Tf-Idf	Glove	Glove and Specific Word Emb	RoBERTa
Models	LDA		x			
	S-LDA		x			
	BTM		x			
	SVM		x	x		
	MLP	x				
	LSTM		x	x	x	x
	RoBERTa 2 dense layers					x

The LSTM model which we created consists of two bidirectional LSTM layers, the first with 128 cells and the second with 64 cells. Between these two layers we added a spatial dropout layer which sets equal to 0 the output some channels of the previous layer with probability 0.30. After the second layer we added another dropout layer which sets to 0 weights with probability 0.25. On top of the LSTM models we added two dense layers, the first one with 128 nodes and ReLU activation function and the second with a number of nodes equal to the number of aspect classes in each dataset. The second dense layer applies the sigmoid function to give the final results. We chose sigmoid because we do not want deciding one

class excluding the choice of the rest of the classes.

RoBeRTa model is a pre-trained model on raw text and that is the reason why we do not add any pre-processing steps to the text before feeding it to the model. Moreover, the model has on top a dense layer with dimensions 1024 and ReLU activation function and the output dense layer with 6 or 8 units, depending on the number of the aspect categories and sigmoid activation function.

Finally, we combine the power of the LSTM model with the structure as described above with the Transformer model RoBERTa. We use the pre-trained RoBERTa as a input to the LSTM model alone and in combination with other word embeddings. We understand that the size of the embedding that RoBERTa creates is bigger than GloVe or the specific embedding that is described in section 4.3 but we wanted to determine whether they can add information to the model.

The aspect categorization sub-task can be described as a multi-label classification problem in which the sentences can belong to more than one class.

6.2 Sentiment analysis

Polarity extraction based on the predicted labels was performed by using various models. The pipeline of the sentiment analysis is presented bellow.

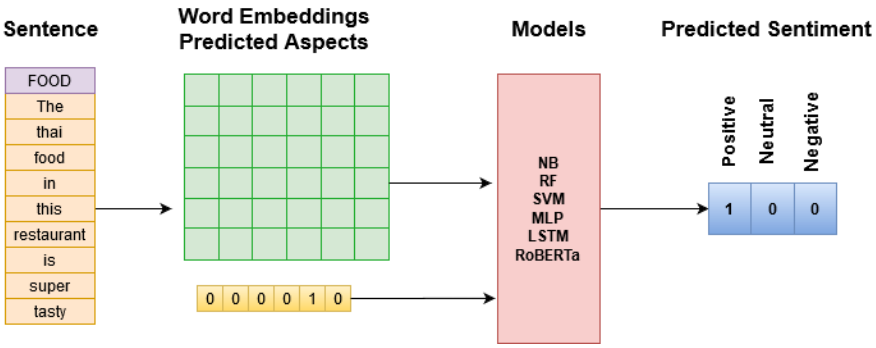


Figure 6.2: Sentiment analysis pipeline

The input of all the models is the representation of the sentences and the predicted labels. Specifically, in the machine learning models we added the predicted labels as numeric features using 0 when the category is not predicted and 1 when it is predicted. In the deep learning models we inserted the labels in two ways. First we added the name of the category in the beginning of the sentence before creating the word embeddings and in the first dense layer after the LSTM or RoBERTa as numeric features using the aforementioned representation. Naive Bayes, random forest, SVM and MLP take as input the sentences transformed with Count Vectorizer or Tf-Idf. The input of the LSTM model are the same as mentioned above in section 6.1 and the output is the predicted sentiment for all the models. In the following Figurewe can see the combinations of the word embeddings that we used with the models.

Table 6.2: Word embeddings and models for sentiment analysis

Word Representation						
		Count Vectorizer	Tf-Idf	Glove	Glove and Specific Word Emb	RoBERTa
Models	NB	x				
	RF		x			
	SVM		x	x		
	MLP		x			
	LSTM		x	x	x	x
	RoBERTa 2 dense layers					x

The architecture of the model LSTM is the same except the function that determines the output which is softmax for this task. The reason why we chose softmax is because when the model predicts one class the rest of the classes are not possible to be predicted. This sub-task can be described as a multi-class classification problem, because the classes are mutually exclusive. Additionally, we tried two unusual combinations of word embeddings and models, the SVM model

with GloVe as input and LSTM model with input created by Tf-Idf.

It is important to mention that before feeding the sentences and the predicted aspects to the models we "unfolded" the sentences. That means that in the sentences that contained more than one predicted aspect we created copies taking into account the number of the predicted aspects aiming to create sentences which have only one aspect. In that way, our models had to predict only one sentiment for each sentence.

Finally, before performing the task of sentiment analysis we chose the best results we got from the previous task of aspect extraction, aiming to give the same information to the models and be able to compare their outcomes. The result in the sentiment analysis are based on the correctly predicted aspect only.

6.3 Evaluation measures

In order to evaluate the performance of our models we are going to use accuracy and F1-micro score for aspect categorization. In this problem, by accuracy we mean the correctly predicted set of labels for each sentence. If one label category is missing, or is wrongly predicted the set is considered wrong. Accuracy is calculated by the following equation:

$$accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \tag{6.1}$$

Moreover, we are using the F1-micro score as proposed in [38]. The dataset as presented in section 3.2 is not balanced for both domains. That is the reason why we did not choose simple F1-score or F1-macro which do not take into account the imbalance of the dataset. In order to define F1-micro, the precision and recall are calculated globally for all the labels by counting all the true positives, false negatives and false positives as explained in [34].

The calculation of the aforementioned measures is based on the confusion matrix for each class.

Table 6.3: Confusion matrix

		Predicted Labels	
		Positive	*Negative*
True Labels	*Positive*	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

That can be translated to the following confusion matrix for each class:

Table 6.4: Confusion matrix for each class

		Predicted Labels	
		Class i Predicted	*Class i Not Predicted*
True Labels	*Class i*	True Predicted	False Not Predicted
	Not class i	False Predicted	True Not Predicted

Then the F1-micro is calculated by the following formula:

$$F1_{micro} = 2 * \frac{Micro - precision \ * \ Micro - recall}{Micro - precision \ + \ Micro - recall} \tag{6.2}$$

where micro-recall and micro-precision are shown below:

$$Micro - Recall = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} TP_i + FN_i} \tag{6.3}$$

$$Micro - Precision = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} TP_i + FP_i} \tag{6.4}$$

In order to measure the performance of our models for the sentiment analysis task, we are going to use accuracy which calculated with the equation 6.5 and the confusion matrix for the three sentiment classes.

$$accuracy = \frac{TP \ + \ TN}{TP \ + TN \ + FP \ + FN} \tag{6.5}$$

7. Results

In this section the procedure of getting our results will be presented explicitly. The parameters of each model and the results will be presented for aspect extraction and sentiment analysis. All the results were calculated with the same random state aiming to make the comparison as fair as possible. Moreover, we used a validation set which is 25% of the train set e.g. 427 sentences in the Restaurant domain and 510 for the Laptop domain. The validation set shares the same topic distribution with the train set. The test set that we used is the one that provided by the SemEval 2016 workshop. The size of the test set is 587 sentences in the Restaurant domain and 573 in the Laptop domain.

7.1 Aspect categorization

7.1.1 Latent Dirichlet allocation

The first model that we examined was LDA. The input of the model is the text data transformed in a way that the model is able to understand. Moreover, we added the number of topics, which is the number of the aspect categories we have in out dataset. There are other parameters as well, such as $\alpha = 0.1$ and the number of iterations which is set to 20000. The output of the model is a vector with length the number of topics we have set. The numbers on this vector represent the probabilities that each sentence belongs to each topic and the summation is equal to 1.

```
[3.15576099e-03 9.06557595e-01 1.44203759e-03 9.52363848e-03
 5.75588943e-02 2.17620739e-02]
```

Figure 7.1: Example output of LDA

After having these probabilities for each sentence we set a threshold and if the probabilities are higher than the threshold the topic is assigned to the sentence, if not the topic is rejected. In order to determine the threshold we used a validation set. We tried different threshold values from 0 to 1 with step 0.05 and we kept the value for which we get the highest accuracy. Accuracy in aspect categorization means that the complete set of aspect categories was predicted correctly. Without data augmentation the best threshold for the evaluation set is 0.45 whilst for augmented data it was 0.4.

Although LDA is considered a good model for topic classification, the sparse nature of the data and the limited size of our dataset did not interact well with the model and the results were not as good as we expected. In Figure 7.2, we can see which words LDA used to assign the topics to each sentence. It is easy to notice that different aspect categories use the same words and thus the model might be confused. This may as well have been an obstacle for LDA to give good results.

```
Topic 0: get like service make table dont come staff never wait
Topic 1: place great try love sushi like special must go roll
Topic 2: food best restaurant one place experience good new ive great
Topic 3: go would back place restaurant recommend dinner nice time never
Topic 4: food great service good price excellent really wine well friendly
Topic 5: food good pizza dish delicious taste fresh also chicken appetizer
```

Figure 7.2: Important words for LDA for aspect classification, Restaurant domain

```
Topic 0: laptop buy toshiba would product computer one purchase get nt
Topic 1: great screen price keyboard good use quality expect well love
Topic 2: battery life long enough fast give good last nt hour
Topic 3: computer use time get laptop macbook best love pro thing
Topic 4: great work software come computer program like use processing love
Topic 5: nt laptop like would want get use work even recommend
Topic 6: computer laptop easy use make need machine get worth much
Topic 7: work month get day drive take hard go one button
```

Figure 7.3: Important words for LDA for aspect classification, Laptop domain

The results of the LDA model are presented below.

Table 7.1: LDA results for Restaurant dataset

	Accuracy	**F1-micro**
Without data augmentation	0.24	0.35
With data augmentation	0.23	0.32

As we know, LDA model does not assign labels to the topics. For that reason we conducted a permutation to the order of the probabilities output and used the validation set in order to find the best match, in combination with the appropriate threshold, between the known aspects and the topics that LDA exported. From the permutation we concluded that topic 3 represents the category Restaurant, topic 1 the category Location, topic 2 the category Ambience, topic 0 the category Service, topic 5 the category Food and topic 4 the category Drinks for not augmented data. We can see from the words in Figure 7.2 that LDA exported that topic 4 is the only one that contains a word that is related to the category Drinks. There are more words that LDA uses but these are the top 10. Moreover, topic 0 contains words that can be related to the category Service like Service, staff and wait. The rest of the topics seem to be more confusing, however almost all the words included in topic 5 are related to Food. After data augmentation the topics changed: topic 2 talks about Restaurant, topic 3 about Location, 4 about Ambience, 1 about Service, 5 about Food and 0 about Drinks.

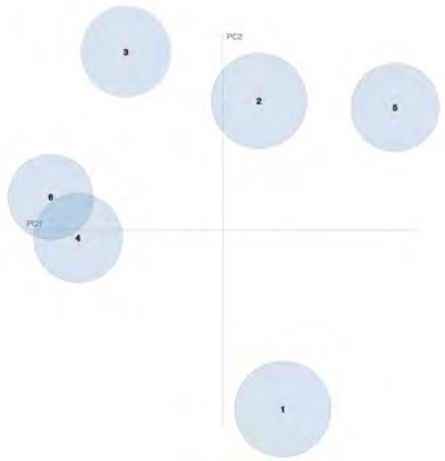


Figure 7.4: LDA Restaurant Topic distance map before augmentation

Figure 7.5: LDA Restaurant Topic distance map after augmentation

For the Laptop dataset we can see that model performs poorer. For not augmented the best threshold is 0.45 and the best permutation is $(0, 6, 1, 2, 3, 5, 4, 7)$ for the topics Hardware, Software, Warranty, OS, Company, Support, Laptop and Shipping. For augmented data the best threshold is 0.35 and the best permutation is $(6, 4, 5, 1, 2, 7, 0, 3)$.

Table 7.2: LDA Results for Laptop dataset

	Accuracy	**F1-micro**
Without data augmentation	0.12	0.19
With data augmentation	0.14	0.22

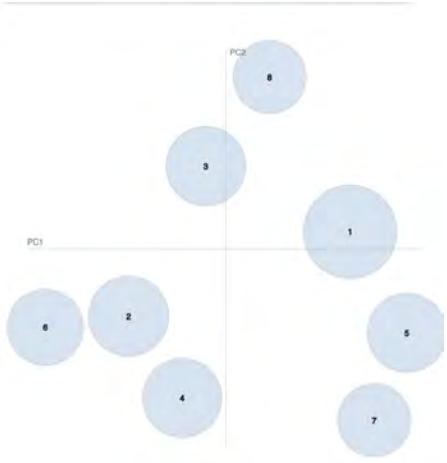


Figure 7.6: LDA Laptop Topic distance map before augmentation

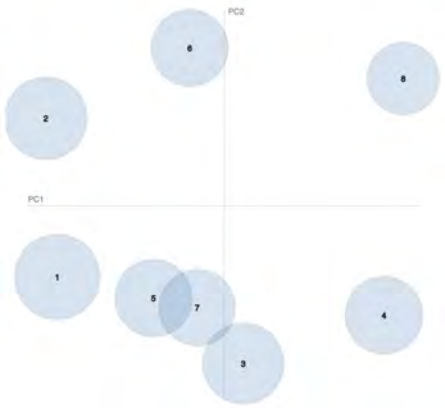


Figure 7.7: LDA Laptop Topic distance map after augmentation

We can see in Figure 7.7, using the package provided by [51] that after data augmentation the topics seem to overlap more and that might be the reason why the predictive power of the model is reducing.

7.1.2 Seeded LDA

The next model we implemented is seeded LDA, which is a semi supervised implementation of the regular LDA. Seeded LDA, as presented in the model description [56], takes as input the sentences and the seed topic list of the underrepresented topics in our dataset which are Location, Ambience and Drinks for the Restaurant domain and Shipping, Software, Warranty, OS, Company, and Support for the Laptop domain. Moreover, we set the number of iterations to 20000. The output is the same with the output of LDA. After having extracted the probabilities, we set a threshold and with the same procedure, if the probability is higher than this threshold the topic or topics are assigned to each sentence. The threshold that gives the best results in the validation set is 0,4 for not augmented data and 0.35

for augmented. We can see that again, data augmentation did not help.

Table 7.3: Seeded LDA results for Restaurants

	Accuracy	**F1-micro**
Without data augmentation	0.28	0.35
With data augmentation	0.26	0.39

For the Laptop dataset this model gives worse results. The best threshold for not augmented data is 0.5 and for augmented 0.45.

Table 7.4: Seeded LDA results for Laptops

	Accuracy	**F1-micro**
Without data augmentation	0.15	0.20
With data augmentation	0.16	0.22

7.1.3 Biterm topic model

The third machine learning model that we tried is Biterm Topic Model. This model has been created for short text classification. However, this model also did not give the results that we expected. This model does not output the labels for the topics and that is why we also used permutation. The best permutation for not augmented data is the following: topic 4 is Restaurant, topic 0 is Location, topic 5 is Ambience, topic 1 is Food, topic 2 is Drinks and topic 3 is Service. The best threshold is 0.3. In 7.8 we can see the distance between the topics as defined by Biterm model before augmentation and in 7.9 after. The best threshold for augmented data is 0.35 and the topic matching is the following: Restaurant topic 4, Location topic 1, Ambience topic 0, Service topic 5, Food topic 3 and Drinks topic 2.

Figure 7.8: Bitterm Restaurant Topic distance map before augmentation

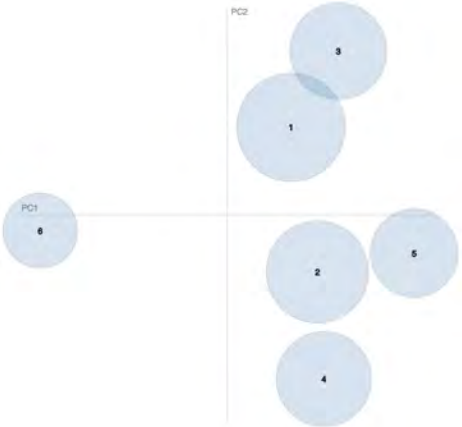


Figure 7.9: Bitterm Restaurant Topic distance map after augmentation

We can see in Figure 7.9 that after data augmentation the distance between the topics differentiates. After that procedure, the topics seem to be more discreet and do not overlap as much as before. Augmentation helped Biterm topic model give better results.

In the following table the results of Biterm model are presented. We can see that for that model data augmentation had positive results and increased the performance of the model.

Table 7.5: Biterm model results for Restaurant dataset

	Accuracy	**F1-micro**
Without data augmentation	0.16	0.30
With data augmentation	0.23	0.33

For the Laptop dataset before data augmentation the best threshold is 0.35 and the best permutation is (3, 2, 0, 6, 4, 7, 1, 5). After data augmentation the best

threshold is 0.3 and the best permutation is (1, 5, 0, 3, 2, 6, 4, 7).

Table 7.6: Biterm model results for Laptop dataset

	Accuracy	**F1-micro**
Without data augmentation	0.12	0.20
With data augmentation	0.10	0.19

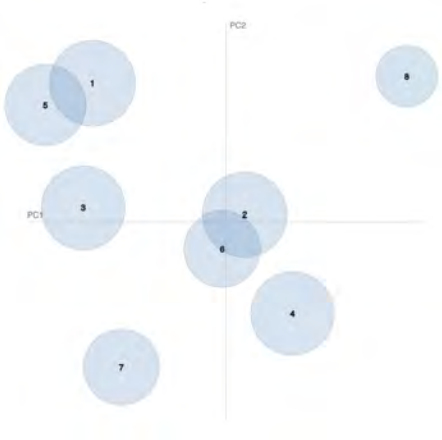


Figure 7.10: Topic distance map before augmentation Laptop domain

Figure 7.11: Topic distance map after augmentation Laptop domain

The main problem with the dataset that we are using is that the aspect category Restaurant and Laptops, are very general and to some extend they may contain information that could be easily misclassified from an algorithm. Furthermore, the aforementioned models use words to categorize the sentences and these words can be included in multiple aspect categories.

7.1.4 Support vector machine

SVM takes as input the features of the dataset. In order to decide if we are going to use the simple Count Vectorizer or Tf-Idf we tried them both on the valida-

tion set and chose the second which gave better results. Furthermore, we used as additional input combining it with Tf-Idf, the output probabilities of the seeded LDA model in order to provide more information to the model. In more detail, the output of the seeded LDA is a vector with length 6 with the predicted probabilities for each class. We concatenated this vector with the vector that Tf-Idf produced.

The following Figurecreated using the Python package ELI5 [30], portraying the learned weights of the SVM, we can see that the additional input from the Seeded LDA helped the model to give better results. The weights are referring to the categories with the following order: Restaurant, Location, Ambience, Service, Food and Drinks. The positive weights in green mean that the specific predicted probability pushes the model towards deciding that the sentence belongs to that column's category. On the other hand, negative weights push the model away from choosing this category. This indicates that a more complex approach to the probability outputs instead of a simple threshold can give better accuracy results.

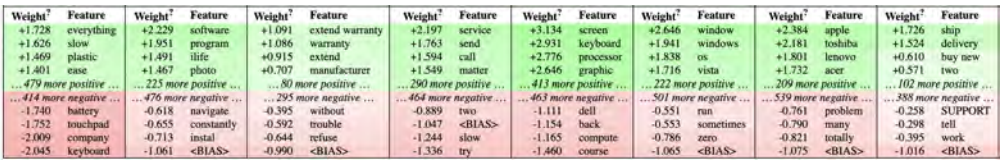
Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature
+0.370	RESTAURANT	+0.042	AMBIENCE	+0.049	AMBIENCE	+0.486	SERVICE	+0.907	FOOD	+0.094	DRINKS
+0.021	AMBIENCE	+0.033	LOCATION	+0.038	LOCATION	+0.157	DRINKS	-0.264	LOCATION	+0.024	AMBIENCE
-0.140	SERVICE	-0.181	DRINKS	+0.012	RESTAURANT	+0.105	AMBIENCE	-0.306	AMBIENCE	-0.054	SERVICE
-0.180	LOCATION	-0.216	RESTAURANT	-0.069	DRINKS	-0.171	RESTAURANT	-0.337	DRINKS	-0.157	RESTAURANT
-0.332	DRINKS	-0.245	SERVICE	-0.224	FOOD	-0.192	LOCATION	-0.479	RESTAURANT	-0.257	FOOD
-0.734	FOOD	-0.435	FOOD	-0.294	SERVICE	-0.469	FOOD	-0.591	SERVICE	-0.360	LOCATION

Figure 7.12: Seeded LDA input weights

Moreover, below we can see an example of the weights that the model gives to the words in the dataset. Specifically, the table shows 4 top words have the highest and lowest weights for each category.

Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature
+1.634	restaurant	+2.585	view	+3.292	decor	+5.237	service	+4.564	food	+2.764	wine
+1.537	get table	+1.501	location	+3.119	atmosphere	+3.142	staff	+2.376	sushi	+2.149	drink
+1.479	spend	+1.167	right	+2.617	ambience	+2.391	waiter	+1.977	meal	+1.894	sake
+1.414	price	+0.982	end	+2.365	beautiful	+1.931	manager	+1.910	delicious	+1.813	beer
... 316 more positive ...		*... 147 more positive ...*		*... 323 more positive ...*		*... 347 more positive ...*		*... 408 more positive ...*		*... 185 more positive ...*	
... 435 more negative ...		*... 404 more negative ...*		*... 418 more negative ...*		*... 405 more negative ...*		*... 347 more negative ...*		*... 463 more negative ...*	
-1.528	cuisine	-0.435	FOOD	-0.725	im	-0.840	disappoint	-1.269	restaurant	-0.653	try
-1.557	sushi	-0.445	would expect	-0.758	leave	-0.907	cash	-1.311	day	-0.735	great place
-1.567	order	-0.463	joint	-0.807	far	-0.968	worth	-1.449	apology	-0.751	dessert
-1.577	lot	-1.002	<BIAS>	-0.898	<BIAS>	-1.091	crowd	-2.094	prepare	-0.942	<BIAS>

Figure 7.13: Input's weights for Restaurant domain



Weight	Feature	Weight	Feature	Weight	Feature	Weight	Feature	Weight	Feature	Weight	Feature	Weight	Feature	Weight	Feature
+1.728	everything	+2.229	software	+1.091	extend warranty	+2.197	service	+3.134	screen	+2.646	window	+2.384	apple	+1.726	ship
+1.626	slow	+1.951	program	+1.086	warranty	+1.763	send	+2.931	keyboard	+1.941	windows	+2.181	toshiba	+1.524	delivery
+1.469	plastic	+1.491	ilife	+0.915	extend	+1.594	call	+2.776	processor	+1.838	os	+1.801	lenovo	+0.610	buy new
+1.401	ease	+1.467	photo	+0.707	manufacturer	+1.549	matter	+2.646	graphic	+1.716	vista	+1.732	acer	+0.571	two
...479 more positive ...		...225 more positive ...		...80 more positive ...		...290 more positive ...		...413 more positive ...		...222 more positive ...		...209 more positive ...		...102 more positive ...	
...414 more negative ...		...476 more negative ...		...295 more negative ...		...464 more negative ...		...463 more negative ...		...501 more negative ...		...539 more negative ...		...388 more negative ...	
-1.740	battery	-0.618	navigate	-0.395	without	-0.889	two	-1.111	dell	-0.551	run	-0.761	problem	-0.258	SUPPORT
-1.752	touchpad	-0.655	constantly	-0.592	trouble	-1.047	<BIAS>	-1.154	back	-0.553	sometimes	-0.790	many	-0.298	tell
-2.009	company	-0.713	instal	-0.644	refuse	-1.244	slow	-1.165	compute	-0.786	zero	-0.821	totally	-0.395	work
-2.045	keyboard	-1.061	<BIAS>	-0.990	<BIAS>	-1.336	try	-1.460	course	-1.065	<BIAS>	-1.075	<BIAS>	-1.016	<BIAS>

Figure 7.14: Input's weights for Laptop domain

Finally, the results of the model are presented below.

Table 7.7: SVM results for Restaurant domain

	Accuracy	**F1-micro**
Without data augmentation	0.58	0.73
With data augmentation	0.56	0.72

Table 7.8: SVM results for Laptop domain

	Accuracy	**F1-micro**
Without data augmentation	0.58	0.71
With data augmentation	0.60	0.73

Usually, Support vector algorithms are combined with word embeddings created by simple methods that are based on counting the frequency of the words occurring in the text. However, we decided to perform an extra experiment and use the GloVe word embedding as input to the SVM model. We noticed that SVM was able to perform relatively well with the simpler inputs, taking into account how simple is its. That is the reason why we chose to combine it with this more complex word representation than includes semantic relationships between the words. Moreover, another characteristic of SVM is that it is able to handle data with higher dimensions such as GloVe. We used three techniques in order to be able to use the GloVe embeddings as an input. The first one is to take the summation of all the word Glove vectors in a sentence. The second is to use feature flattening

and stack one vector under the other. Finally, we used the average of the GloVe vectors. The results are the following:

Table 7.9: SVM with GloVe in Restaurant domain

	Before data augmentation		After data augmentation	
	Accuracy	F1-score	Accuracy	F1-score
Summation	0.51	0.71	0.51	0.71
Flattening	0.49	0.69	0.58	0.76
Average	**0.57**	**0.73**	0.64	0.79

Table 7.10: SVM with GloVe in Laptop domain

	Before data augmentation		After data augmentation	
	Accuracy	F1-score	Accuracy	F1-score
Summation	0.51	0.70	0.51	0.68
Flattening	0.51	0.69	0.52	0.70
Average	**0.58**	**0.70**	0.58	0.72

We can see that the best results in both domains are almost the same achieved when we used Tf-Idf as input. However, Tf-Idf is much simpler than GloVe since it only takes into account the frequency of the words. That is why we concluded that Tf-Idf is a better input to the SVM model for this task.

7.1.5 Multi-layer perceptron

We followed the same procedure to decide the input of the model. Count Vectorizer gave better results on the validation set and we chose that instead of Tf-Idf. Moreover, we again added the predicted output of the Seeded LDA. The results of the MLP are given in the following tables.

Table 7.11: MLP results for Restaurants

	Accuracy	**F1-micro**
Without data augmentation	0.55	0.69
With data augmentation	0.51	0.64

Table 7.12: MLP results for Laptops

	Accuracy	**F1-micro**
Without data augmentation	0.54	0.68
With data augmentation	0.58	0.70

MLP is a simple feed forward neural network and the results that it gave were slightly worse than those from SVM and not as good as we expected. That is the reason why the next experiments we conducted utilized a more complex and sophisticated network, a Long short-term memory network.

7.1.6 Long short-term memory

In the first implementation the input embedding of the LSTM model was created by using GloVe. The results are shown in the following tables:

Table 7.13: LSTM with GloVe for Restaurants

	Accuracy	**F1-micro**
Without data augmentation	0.70	0.8
With data augmentation	0.69	0.79

Table 7.14: LSTM with GloVe for Laptops

	Accuracy	**F1-micro**
Without data augmentation	0.61	0.71
With data augmentation	0.58	0.70

Following, we used as input the combination of the word embedding created by GloVe and the Domain Specific embedding mention in 4.3. The results are slightly better.

Table 7.15: GloVe and Domain Specific as input to the LSTM for Restaurants

	Accuracy	**F1-micro**
Without data augmentation	0.71	0.79
With data augmentation	0.70	0.79

Table 7.16: GloVe and Domain Specific as input to the LSTM for Laptops

	Accuracy	**F1-micro**
Without data augmentation	0.62	0.73
With data augmentation	0.60	0.72

We decided to use both the embedding from GloVe alone and in combination with the Domain Specific embedding to see if could provide more information to the model and increase the results. However, we can see that there was a slight raise in the order of one percent which is not important.

Furthermore, we tried a combination that might be considered unusual, as we have not encountered it in the related work, and we used Tf-Idf as an input to the LSTM neural network. We did the following experiment because we wanted to confirm that indeed this combination is not used because the results are not good.

Table 7.17: Tf-Idf as input to LSTM

	Accuracy	**F1-micro**
Restaurant Domain	0.54	0.67
Laptop Domain	0.59	0.67

We can see that in the Restaurant domain the results of this combination are much lower compared to the results the LSTM gave with the aforementioned more complicated word embeddings. On the contrary, in the Laptop domain the results are better, approximately the same as when we used the other embeddings. This result probably has occurred due to the predictive power of the LSTM network because the model does not outperform the previous ones. However, the results were not as bad as we expected.

7.1.7 RoBERTa

RoBERTa takes as input the sentences from our dataset. On the top of the pre-trained model we use two dense layers for performing the classification. The results are presented in the following tables.

Table 7.18: RoBERTa with two dense layers for Restaurants

	Accuracy	**F1-micro**
Without data augmentation	0.57	0.71
With data augmentation	0.57	0.72

Table 7.19: RoBERTa with two dense layers for Laptops

	Accuracy	**F1-micro**
Without data augmentation	0.58	0.70
With data augmentation	0.57	0.70

We can see that the results are slightly worse than those of the LSTM network in both domains.

7.1.8 Combination of RoBERTa and LSTM

Finally, we used the RoBERTa pre-trained model as an input to the LSTM model. Following the idea of combining different embedding provided by [59] we tried different combinations to find the one which gives the best results. In the beginning we tried RoBERTa alone and then we combined it with the embeddings created by Glove and the results are presented below.

Table 7.20: RoBERTa as input to the LSTM for Restaurants

	Accuracy	**F1-micro**
Without data augmentation	0.76	0.83
With data augmentation	0.74	0.83

Table 7.21: RoBERTa as input to the LSTM for Laptops

	Accuracy	**F1-micro**
Without data augmentation	0.66	0.77
With data augmentation	0.64	0.75

Table 7.22: RoBERTa and GloVe as input to the LSTM for Restaurants

	Accuracy	**F1-micro**
Without data augmentation	0.76	0.85
With data augmentation	0.76	0. 85

Table 7.23: RoBERTa and GloVe as input to the LSTM for Laptops

	Accuracy	**F1-micro**
Without data augmentation	0.67	0.77
With data augmentation	0.66	0.77

Finally, we tried to maximize the information fed to the LSTM network, by combining RoBERTa, the embeddings created from GloVe and the specific domain embedding in order to see if the later would be able to improve the results as it has been trained on these specific domains.

Table 7.24: LSTM with input GloVe and Domain Specific embedding and RoBERTa for Restaurants

	Accuracy	**F1-micro**
Without data augmentation	0.78	0.86
With data augmentation	0.74	0.82

Table 7.25: LSTM with input GloVe and Domain Specific embedding and RoBERTa for Laptops

	Accuracy	**F1-micro**
Without data augmentation	0.66	0.76
With data augmentation	0.67	0.77

In the Restaurant domain we were able to achieve the highest accuracy and F1-score, by using as input RoBERTa, word embedding created by GloVe and the specific domain embedding. In the Laptop domain the results were almost the same across the three experiments but typically the best results were achieved with RoBERTa as input to the LSTM.

7.2 Aspect extraction result summary

In this section we are going to present and analyze the results of aspect classification. The following table contains the outcome of each model.

Table 7.26: Aspect categorization results without data augmentation

Models	**Restaurant Accuracy**	**Restaurant F1-micro**	**Laptop Accuracy**	**Laptop F1-micro**
LDA	0.24	0.35	0.12	0.19
Seeded LDA	0.28	0.35	0.15	0.20
BTM	0.16	0.30	0.12	0.20
SVM	0.58	0.73	0.58	0.71
MLP	0.55	0.69	0.54	0.68
LSTM with GloVe	0.70	0.80	0.61	0.71
LSTM with GloVe and Domain Emb.	0.71	0.79	0.62	0.73
RoBERTa	0.57	0.71	0.58	0.70
LSTM with RoBERTa	0.76	0.83	**0.66**	**0.77**
LSTM with GloVe and RoBERTa	0.76	0.85	**0.66**	**0.77**
LSTM with GloVE and RoBERTa and Domain Emb.	**0.78**	**0.86**	0.66	0.76

The first three probabilistic models do not give the results that we expected. We already mentioned that the main cause of it is the sparsity of the data, the small size of the training set, the overlapping nature of the topics and the length of the sentences. The machine learning model SVM performed much better and achieved accuracy 58 percent and F1-micro score 0.73 and in the Restaurant domain and 58 percent and 0.71 in the Laptop domain respectively. MLP neural network performed slightly worse than the two aforementioned models. For LSTM model

we tried different inputs and the results were very similar for all the different inputs. LSTM managed to increased the correctly predicted aspect categories from almost 60 percent given by SVM to 70 percent. RoBERTa model with two dense layers shows a decrease on the results, comparing it to the LSTM model. However, this decrease is probably due to the dense layers because when combining the pre-trained Transformers model with the LSTM model the results are the best possible in our research. We managed to achieve 78 percent accuracy and 86 percent F1-micro score in aspect extraction task on the Restaurant domain and 66 and 77 percent respectively on the Laptop domain using the model that is illustrated in Figure 7.15

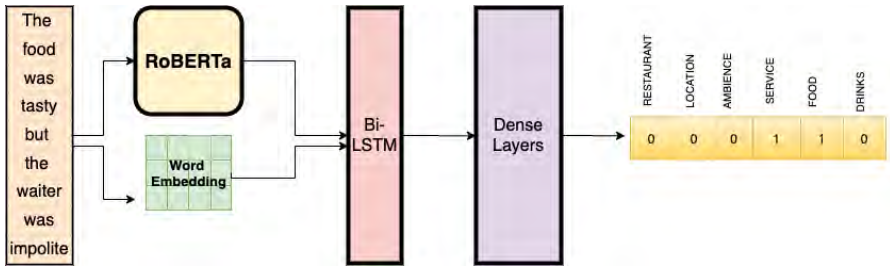


Figure 7.15: Best Model for Aspect Categorization

Even though there is misclassification we can see that the distribution of the

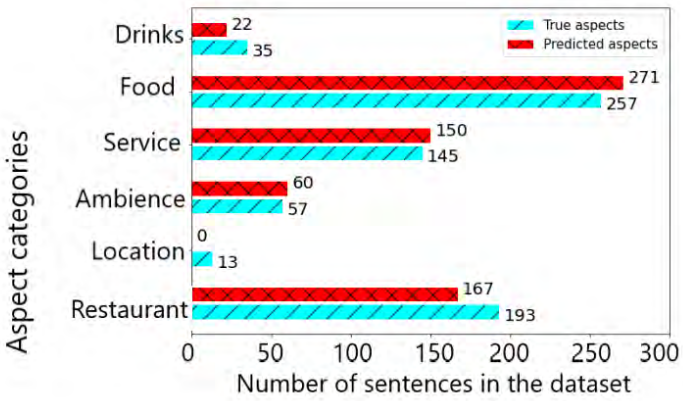


Figure 7.16: Restaurant domain topic distribution

predicted topics is approximately the same with the distribution of the topics in the test set as we can see in the Figures 7.16 and 7.17. That output is important as we will later discuss in section 8.

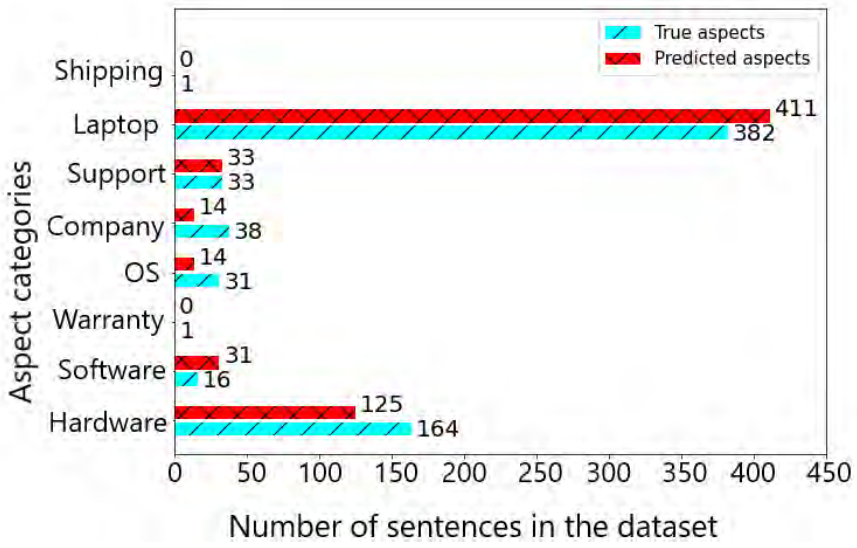


Figure 7.17: Laptop domain topic distribution

During the implementation of our models we used data augmentation to increase the size of our datasets. We noticed that data augmentation helps the models achieve higher accuracy in very few cases. On the contrary, we generally see a slight decrease in the performance of them. In the following table we can see all the results.

Table 7.27: Aspect categorization results with data augmentation

Models	**Restaurant Accuracy**	**Restaurant F1-micro**	**Laptop Accuracy**	**Laptop F1-micro**
LDA	0.23	0.32	0.14	0.22
Seeded LDA	0.26	0.39	0.16	0.22
BTM	0.23	0.33	0.10	0.19
SVM	0.56	0.72	0.60	0.73
MLP	0.51	0.64	0.58	0.70
LSTM with GloVe	0.69	0.79	0.58	0.70
LSTM with GloVe and Domain Emb	0.70	0.79	0.60	0.72
RoBERTa	0.57	0.72	0.57	0.70
LSTM with RoBERTa	0.74	0.83	0.64	0.75
LSTM with GloVe and RoBERTa	**0.76**	**0.85**	0.66	0.77
LSTM with GloVE and RoBERTa and Domain Emb	0.74	0.82	**0.67**	**0.77**

Although, it does not help with accuracy and F1-score we noticed that in the deep learning models, the minimum validation loss is achieved in earlier epoch. That means that the models learn faster when data augmentation is implemented and are able to achieve slightly worse results with much fewer epochs of training.

In general the results in the Laptop domain are poorer than these in the Restaurant domain. That might be because in the Laptop domain there are more technical terms about the devices and the models might not be able to identify the categories in which each term belongs. Both the datasets include a category which is fairly broad and that might have been a problem for the models. Moreover, these two categories are included in the majority of the sentences and that as well might

have been an obstacle.

Finally, we are going to discuss the approximate training time of the models on a Laptop with a 2,2 GHz 6-Core Intel Core i7 and 16GB RAM. The fastest models to train were MLP and SVM which took approximately 1 to 2 minutes before data augmentation. The generative models LDA and seeded LDA took almost 5 minutes to train. The slowest generative model is Bittern topic model that had to be trained for almost ten minutes. The deep learning model LSTM was slower to train and took more time. For each epoch the LSTM took approximately forty seconds and that means that for 50 epochs it took almost 35 minutes. Furthermore, even though we did not train the full RoBERTa model, even using it as an input with the dense layers on top increased the training time to approximately 270 seconds i.e. 4.5 minutes for each epoch on a CPU. That means that for 50 epochs training took almost 4 hours. Last but not least, when using the pre-trained model RoBERTa as an input to the LSTM, the model took approximately the sum of the amount of time when training RoBERTa plus the time for the LSTM, e.g. 300 seconds per epoch and that means more than 4 hours. The aforementioned time slots were calculated when training the models before data augmentation. After data augmentation the training of the models increased proportionally to their initial duration.

7.3 Sentiment analysis

In this section we are going to present and discuss the results of sentiment analysis based on the predicted labels. The confusion matrices for not augmented data will be included in that section. The confusion matrices after data augmentation have been placed in Appendix A in case the reader wants to see in detail how the models perform in each sentiment class.

7.3.1 Naive Bayes classifier

Naive Bayes classifier takes as input the words represented with Count Vectorizer and the predicted aspects. We compared in the validation set the results with Count Vectorizer and Tf-Idf and the first gave better scores. The results are shown below.

Table 7.28: Sentiment analysis with naive Bayes for Restaurants

	Accuracy
Without data augmentation	0.80
With data augmentation	0.80

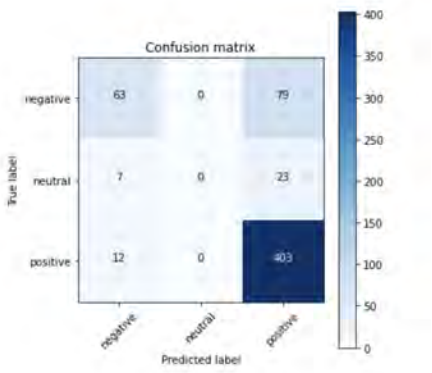


Figure 7.18: Confusion matrix for Restaurants with NB

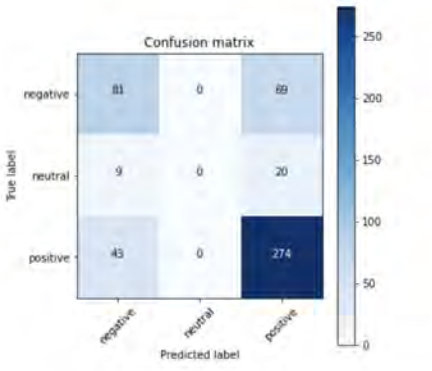


Figure 7.19: Confusion matrix for Laptops with NB

Table 7.29: Sentiment analysis with naive Bayes for Laptops

	Accuracy
Without data augmentation	0.72
With data augmentation	0.69

We can see that the naive Bayes approach, even though it is a very simple algorithm and easy to implement, is able to give high accuracy on predicting the sentiment of the sentences. However, it has bad predicting power on the class neutral in both domains.

7.3.2 Random forest

Having used the simplest model first for predicting the sentiment of the sentiment, we tried the random forest algorithm that is slightly more complicated and hopefully can give better results in the neutral class. Random forest takes as input the words represented with Tf-Idf and the predicted aspects. We decided to use Tf-Idf instead of Count Vectorizer using the same experiment on the validation set. The results are shown below.

Table 7.30: Sentiment analysis with random forest for Restaurants

	Accuracy
Without data augmentation	0.77
With data augmentation	0.74

Table 7.31: Sentiment analysis with random forest for Laptops

	Accuracy
Without data augmentation	0.69
With data augmentation	0.69

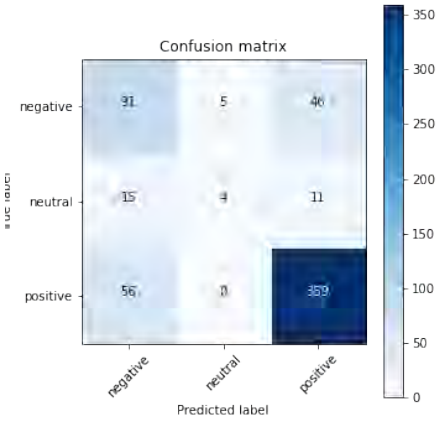


Figure 7.20: Confusion matrix for Restaurants with RF

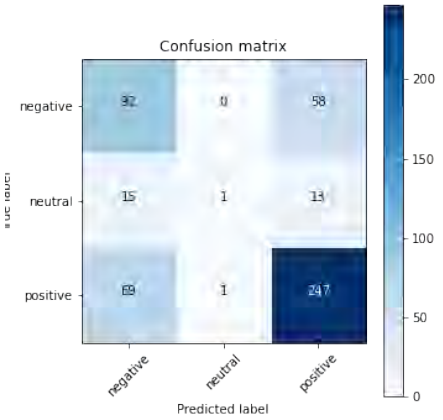


Figure 7.21: Confusion matrix for Laptops for with RF

Random Forest was not able to predict the neutral sentiment before data augmentation. However, we noticed that using that technique helped the model perform slightly better in that class.

7.3.3 Support vector machine

The SVM takes as input the words represented with Tf-Idf and the predicted aspects. We decided that representation using the aforementioned procedure. The results are presented below in the tables 7.32 and 7.33.

Table 7.32: Sentiment analysis with SVM for Restaurants

	Accuracy
Without data augmentation	0.76
With data augmentation	0.75

Table 7.33: Sentiment analysis with SVM for Laptops

	Accuracy
Without data augmentation	0.69
With data augmentation	0.65

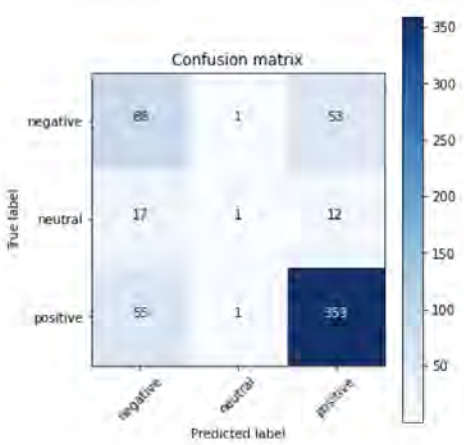


Figure 7.22: Confusion matrix for Restaurants with SVM

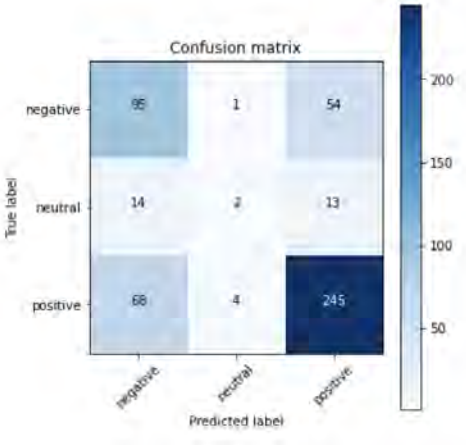


Figure 7.23: Confusion matrix for Laptops with SVM

Finally, we can see an example of the words that affect the model towards each category in the following figures. The first column represents the negative class, the second the neutral and finally the positive.

Weight?	Feature	Weight?	Feature	Weight?	Feature
+1.724	soggy	+1.471	close	+2.069	great
+1.699	worst	+1.381	upscale	+1.747	delicious
+1.697	horrible	+1.283	cash	+1.426	always
+1.466	return	+1.257	environment	+1.402	excellent
+1.461	told	+1.191	nothing	+1.394	nice
... 750 more positive ...		*... 369 more positive ...*		*... 1064 more positive ...*	
... 1049 more negative ...		*... 1373 more negative ...*		*... 729 more negative ...*	
-1.346	mean	-0.719	know	-1.565	done
-1.410	excellent	-0.724	DRINKS	-1.651	worst
-1.442	always	-0.765	dinner	-1.720	horrible
-1.534	delicious	-0.789	great	-1.741	either
-1.829	great	-0.846	<BIAS>	-2.026	return

Figure 7.24: Most important words for Restaurant sentiment

Weight?	Feature	Weight?	Feature	Weight?	Feature
+1.911	slow	+1.695	ok	+2.903	love
+1.609	plastic	+1.551	camera	+2.687	great
+1.551	break	+1.474	cheap	+2.524	apple
+1.406	sometimes	+1.448	may	+2.302	fast
+1.394	dont like	+1.378	get use	+2.018	nice
... 608 more positive ...		*... 381 more positive ...*		*... 694 more positive ...*	
... 745 more negative ...		*... 925 more negative ...*		*... 657 more negative ...*	
-1.651	enough	-0.822	nice	-1.400	bad
-1.779	fast	-0.903	<BIAS>	-1.444	slow
-2.347	apple	-0.923	pc	-1.459	dont like
-2.476	great	-0.926	slow	-1.498	camera
-2.581	love	-0.929	everything	-1.549	plastic

Figure 7.25: Most important words for Laptop sentiment

We can see in figures 7.24 and 7.25 that for the positive and negative classes the words are representative and they express the sentiment category in which they belong. However, in the neutral class the words seem to be irrelevant and most of them do not lead the model to the proper direction. This is represented in the results where it is noticeable that the model is not able to correctly assign this label to the sentences.

Is this sub-task we also decided to use as input the more complex GloVe word embedding and examine the results. We used the same methods that we described in 7.1.4. The results are presented below:

Table 7.34: SVM with Glove for Restaurant domain

	Accuracy before data augmentation	Accuracy after data augmentation
Summation	0.72	0.78
Flattening	0.71	0.71
Average	**0.81**	0.80

Table 7.35: SVM with Glove for Laptop domain

	Accuracy before data augmentation	Accuracy after data augmentation
Summation	0.65	0.71
Flattening	0.65	0.63
Average	**0.70**	0.70

We can see that SVM with a more complex input is able to perform better. GloVe is able to capture the semantic relationships between words and provide more information to the model. Moreover, SVM model was able to handle these information because it it suitable for input with bigger dimension. The results are unexpectedly good and the model was able to gain 5 percent in the Restaurant domain and achieve the best performance. In the Laptop domain the increase was not so important and the results are approximately the same. The confusion matrices for the best input is presented below.

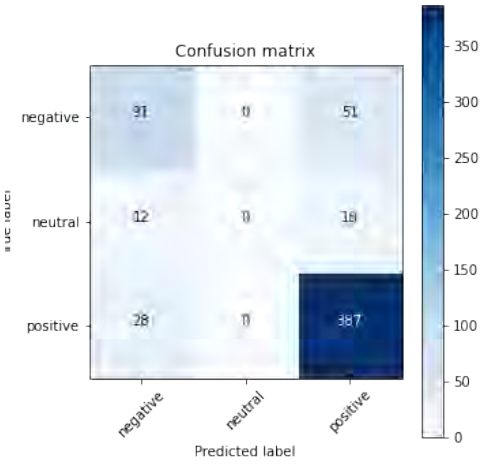


Figure 7.26: Confusion matrix for Restaurants with GloVe as input to SVM with averaging

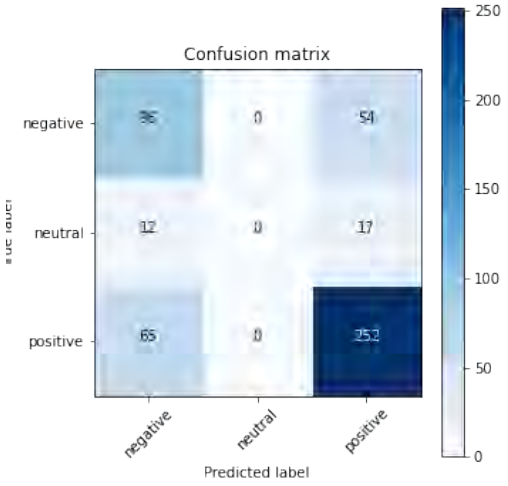


Figure 7.27: Confusion matrix for Laptops with GloVe as input to SVM with averaging

7.3.4 Multi-layer perceptron

MLP takes as input the words represented with Tf-Idf which was decided by the same method applied in the validation set and the predicted aspects. The results are shown below.

Table 7.36: Sentiment analysis with MLP for Restaurants

	Accuracy
Without data augmentation	0.74
With data augmentation	0.73

Table 7.37: Sentiment analysis with MLP for Laptops

	Accuracy
Without data augmentation	0.63
With data augmentation	0.64

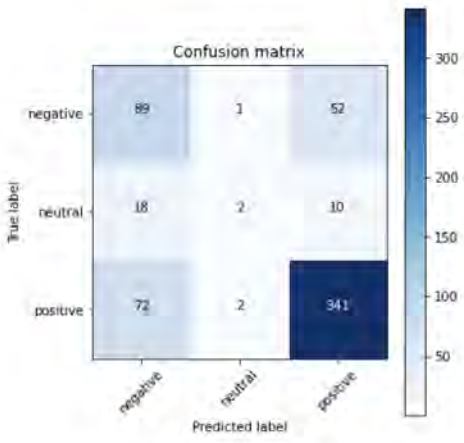


Figure 7.28: Confusion matrix for Restaurants with MLP

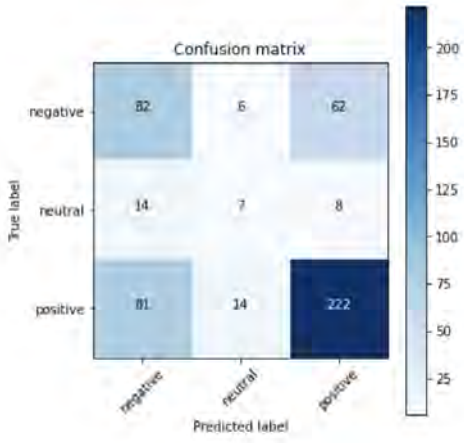


Figure 7.29: Confusion matrix for Laptops with MLP

The model is able to correctly predict very few sentences which belong in the neutral category. In general, MLP is not able to perform very well.

7.3.5 Long sort-term memory

For the LSTM model we tried different word embeddings and examined the results of the model. First we tried word embedding created with GloVe.

Table 7.38: Sentiment analysis with LSTM and GloVe for Restaurants

	Accuracy
Without data augmentation	0.82
With data augmentation	0.80

Table 7.39: Sentiment analysis with LSTM and GloVe for Laptops

	Accuracy
Without data augmentation	0.70
With data augmentation	0.70

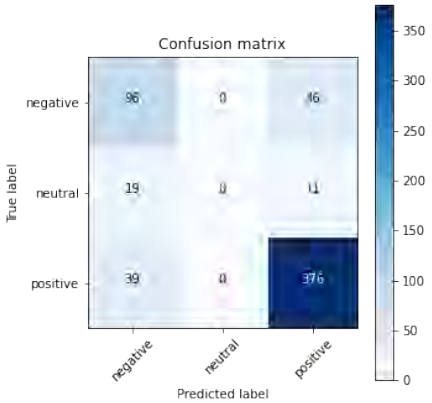


Figure 7.30: Confusion matrix for Restaurants LSTM with GloVe

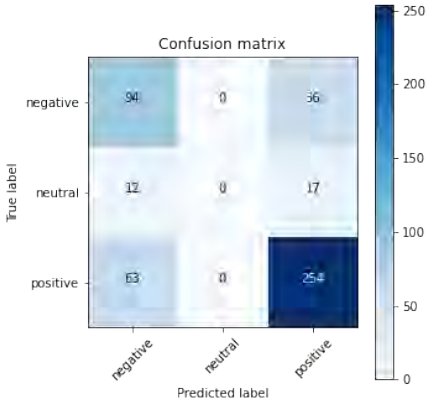


Figure 7.31: Confusion matrix for Laptops LSTM with GloVe

Then we combined the embedding created from GloVe and the Domain Specific embedding.

Table 7.40: LSTM with input of GloVe and Domain Specific for Restaurants

	Accuracy
Without data augmentation	0.82
With data augmentation	0.82

Table 7.41: LSTM with input of GloVe and Domain Specific for Laptops

	Accuracy
Without data augmentation	0.72
With data augmentation	0.71

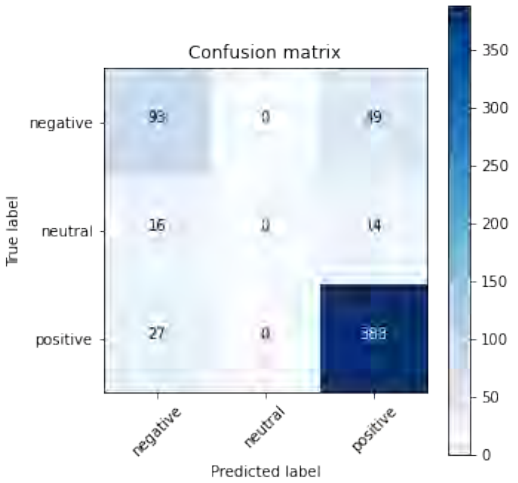


Figure 7.32: Confusion matrix for Restaurants with LSTM, GloVe and domain embedding

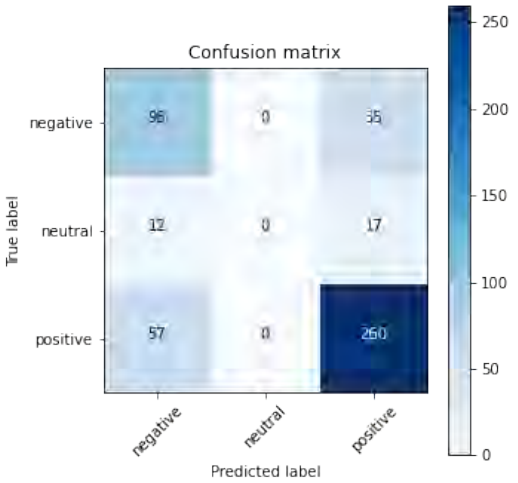


Figure 7.33: Confusion matrix for Laptops with LSTM, GloVe and domain embedding

Finally, we conducted the same experiment using as input the word embedding created with Tf-Idf, which is as we explained above an unusual choice. The results for both domains are the following:

Table 7.42: Tf-Idf as input to the LSTM Restaurant domain

	Accuracy
Without data augmentation	0.79
With data augmentation	0.75

Table 7.43: Tf-Idf as input to the LSTM Laptop domain

	Accuracy
Without data augmentation	0.70
With data augmentation	0.69

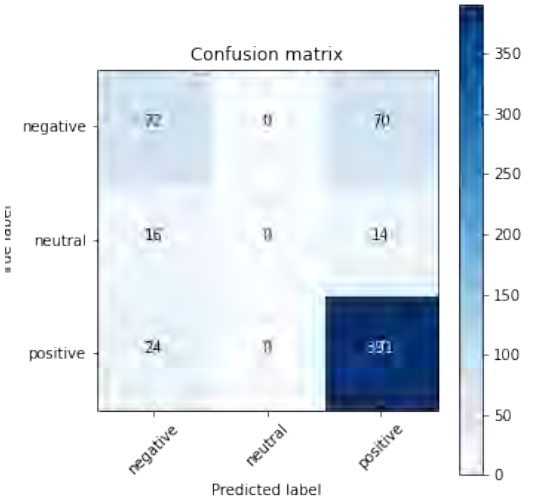


Figure 7.34: Confusion matrix for Restaurants with Tf-Idf and LSTM

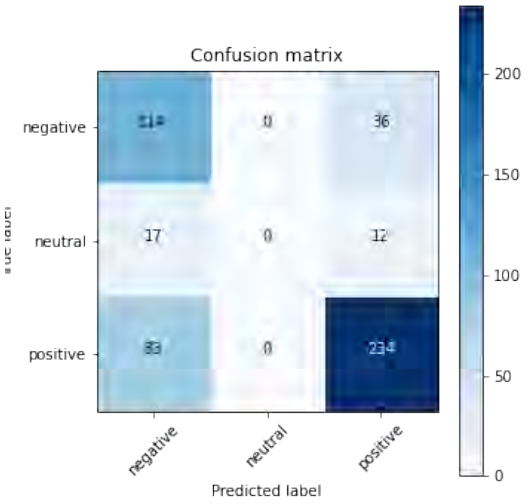


Figure 7.35: Confusion matrix for Laptops with Tf-Idf and LSTM

The results when using Tf-Idf as input to the LSTM model are slightly worse in the Restaurant domain, comparing them with the results LSTM gave with other embeddings. The same stands for the Laptop domain.

7.3.6 RoBERTa

RoBERTa is a model that includes a lot of information because it has been trained on a very big corpus of Wikipedia and books. In order to perform the classification we added two dense layers on top of it. The results are presented below:

Table 7.44: RoBERTa and two dense layers for Restaurants

	Accuracy
Without data augmentation	0.83
With data augmentation	0.83

Table 7.45: RoBERTa and two dense layers for Laptops

	Accuracy
Without data augmentation	0.80
With data augmentation	0.82

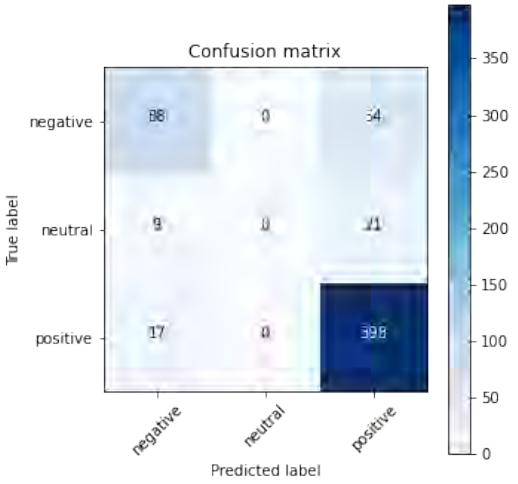


Figure 7.36: Confusion matrix for Restaurants with RoBERTa

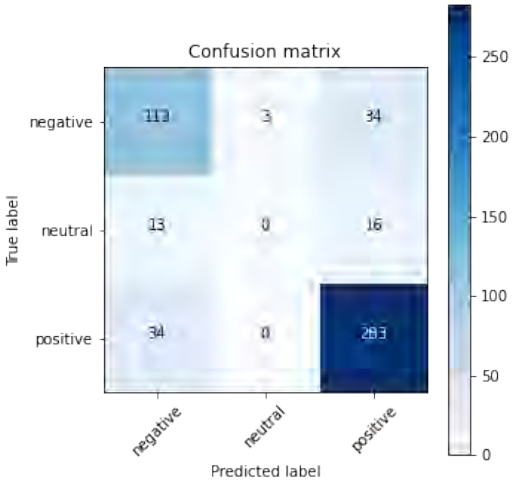


Figure 7.37: Confusion matrix for Laptops with RoBERTa

The accuracy of the correctly predicted sentiment of the sentences increased when comparing these results with the results of the aforementioned models.

7.3.7 Combination of RoBERTa and LSTM

During these experiments we used RoBERTa as input to the LSTM model. RoBERTa has been pre-trained on a very big corpus and contains a lot of information. That is the reason why we chose to combine it with LSTM and not only use it with two dense layers on top because we believed that LSTM would be able to handle the input better. We can see the results in the following tables:

Table 7.46: RoBERTa as input to the LSTM for Restaurants

	Accuracy
Without data augmentation	0.87
With data augmentation	0.88

Table 7.47: RoBERTa as input to the LSTM for Laptops

	Accuracy
Without data augmentation	0.85
With data augmentation	0.84

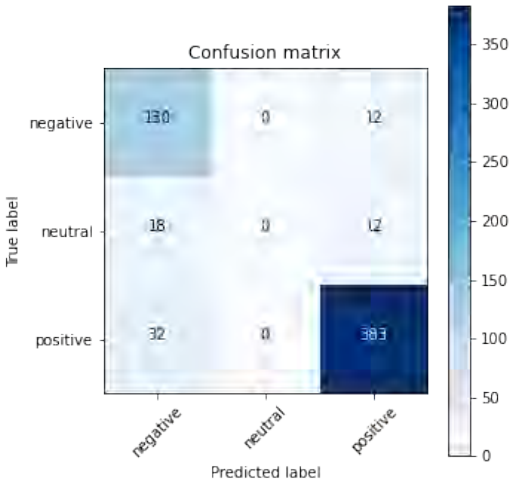


Figure 7.38: Confusion matrix for Restaurants with RoBERTa and LSTM

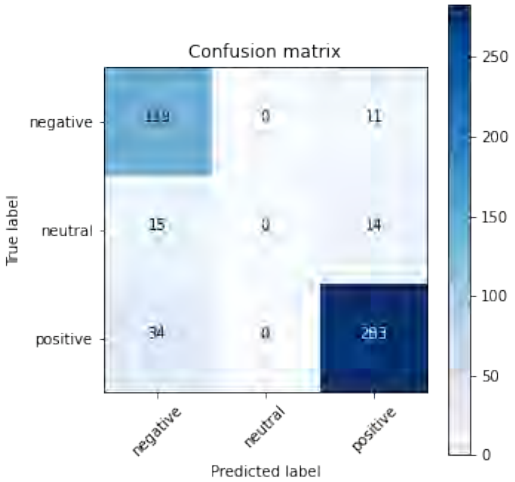


Figure 7.39: Confusion matrix for Laptops with RoBERTa and LSTM

The results confirm our hypothesis that the combination of these two would outperform the LSTM model with simpler input and the dense layers that we used in the initial RoBERTa model.

Furthermore, we decided to add another input to the LSTM model in addition to RoBERTa in order to identify whether they will be able to provide extra information to the model and give better results. For that reason we tried RoBERTa with GloVe and we also added the Domain Specific embedding.

Table 7.48: RoBERTa and GloVe as input to the LSTM for Restaurants

	Accuracy
Without data augmentation	0.87
With data augmentation	0.87

Table 7.49: RoBERTa and GloVe as input to the LSTM for Laptops

	Accuracy
Without data augmentation	0.84
With data augmentation	0.81

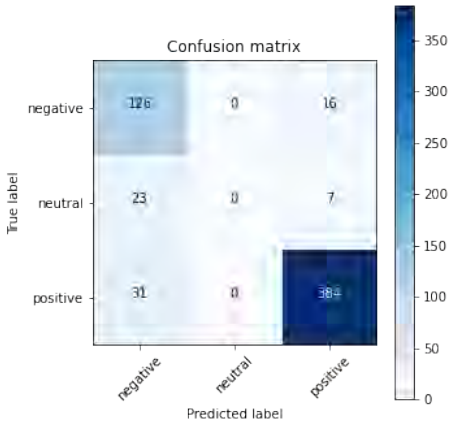


Figure 7.40: Confusion matrix for Restaurants with RoBERTa and LSTM and GloVe

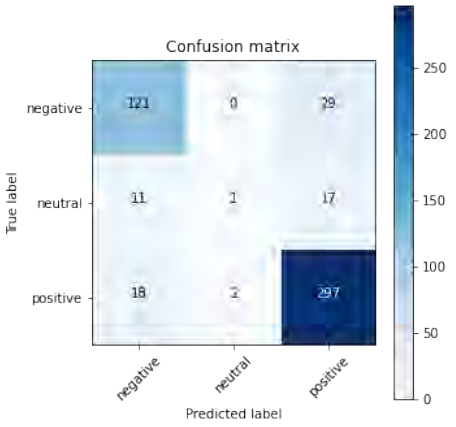


Figure 7.41: Confusion matrix for Laptops with RoBERTa and LSTM and GloVe

Table 7.50: RoBERTa and GloVe and Domain Specific embedding as input to the LSTM for Restaurants

	Accuracy
Without data augmentation	0.85
With data augmentation	0.87

Table 7.51: RoBERTa and GloVe and Domain Specific embedding as input to the LSTM for Laptops

	Accuracy
Without data augmentation	0.85
With data augmentation	0.83

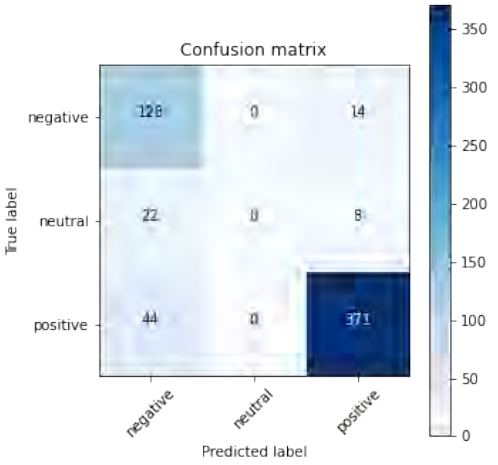


Figure 7.42: Confusion matrix for Restaurants with RoBERTa and GloVe and Domain Specific embedding as input to the LSTM

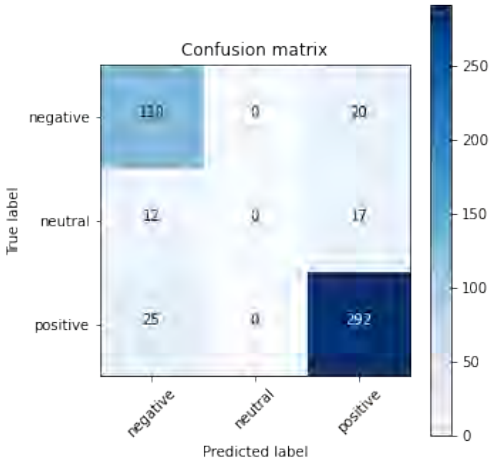


Figure 7.43: Confusion matrix for Laptops with RoBERTa and GloVe and Domain Specific embedding as input to the LSTM

We can see that the model performed almost the same when using the aforementioned input. Depending on the dataset and the specific combination the results slightly differ but the difference is not important i.e. about one percent.

7.4 Comparison models

In this section we are going to discuss a few models that we investigated during our research which use the state of the art methods like attention mechanism and achieve very good results. We were able to reproduce these models and get the results using the same train, test and validation set that we used for our models.

Attention based LSTM

The three models in [57] are a combination of an attention mechanism and a LSTM network for aspect level sentiment classification. As they are described the models are able to pay attention to different parts of each sentence given the predicted aspect categories. The first of these three models is LSTM with aspect Embedding and it is able to create and learn different embedding vector for each aspect class. The second model is an attention based LSTM in which the attention mechanism tries to identify the most important words of the sentence again taking into account the given aspect. The third and last model is a combination of the two aforementioned models. It attaches the embedding vector to the word input vector in order to pass this information to the attention model and capture the dependencies between the words and the aspects.

Hierarchical attention network

Hierarchical Attention Network (HEAT) [9] is an attention model with two-layer attention mechanisms, one for the aspect categories and one for the polarity of these aspects. The first layer gives emphasis on the aspect information. The second layer tries to identify the features that give the sentiment information of the text in order to determine the polarity.

Capsule network

Capsule networks as they are presented in [17] are a type of artificial neural networks which use capsules to store the important information of the data. Usually, they are used for image recognition, however the implementation in [21] showed very good results in the sentiment analysis problem. The results of this model is good and capsule networks seem to be promising for text data.

The results of these models are presented in the following table. Their confusion matrices will be placed in Appendix B in case the reader wants to see in detail how these models perform in each category. In the following section we are going to discuss them in comparison with the results of our models.

Table 7.52: Comparison models Performance

Models	**Restaurants Accuracy**	**Laptops Accuracy**
AE-LSTM	0.74	0.75
AT-LSTM	0.81	0.72
AT-AE-LSTM	0.78	0.77
HEAT	0.82	0.75
Capsule network	0.81	0.75

7.5 Sentiment extraction result summary

In this section we are going to gather and discuss the results from our models. In table 7.53 we have gathered the results of all the aforementioned models, before data augmentation. The machine learning models that we used for performing the sentiment analysis give pretty good results, taking into account how simple and easy to implement they are. Naive Bayes model gave the best results among the models for both the Restaurant and Laptop domain by being able to achieve 80 percent and 72 percent accuracy respectively. The deep learning LSTM model achieved 82 percent accuracy for the Restaurant domain. It is easy to notice that

taking into consideration the complexity of the model and the difference between the simple word representation used by the machine learning models and the word embedding that include semantic relationships between words the increase is not so significant. RoBERTa with two dense layers performed almost the same for the Restaurant domain as the LSTM model. On the contrary, on the Laptop domain RoBERTa provided to the model 8 percent more accurate results. Finally, we achieved the best results when using RoBERTa as an input to the LSTM model as illustrated in Figure 7.44. The model achieved 87 percent accuracy on the Restaurant domain and 85 percent on the Laptop domain.

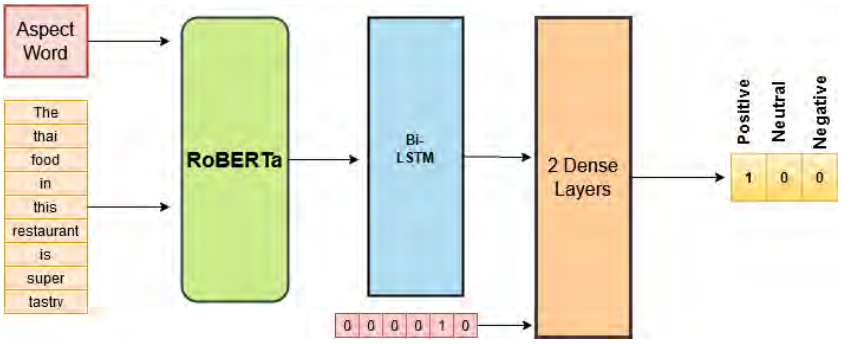


Figure 7.44: Best model for sentiment analysis

In the following table we can see the aforementioned results concentrated.

Table 7.53: Sentiment results without data augmentation

Models	**Restaurant Accuracy**	**Laptops Accuracy**
Naive Bayes	0.80	0.72
Random Forest	0.77	0.68
SVM with Tf-Idf	0.76	0.69
SVM with GloVe	0.81	0.70
MLP	0.74	0.63
LSTM with GloVe	0.82	0.70

LSTM with GloVe and Domain Embedding	0.82	0.72
RoBERTa	0.83	0.80
LSTM with RoBERTa	**0.87**	**0.85**
LSTM with GloVe and RoBERTa	0.87	0.84
LSTM with GloVe and RoBERTa and Domain Embedding	**0.87**	**0.85**

The main problem during performing sentiment analysis was that the dataset is imbalanced and the sentiment category positive has many more examples than the other two. Most of the times, the models predicted wrong the category negative and neutral whereas the category positive had the best predicting ratio. Additionally, the models were unable to identify the category neutral in almost all the sentences that belong to that polarity class. This result was expected because in both the datasets the majority of the sentences belong in the category positive, fewer in the class negative and those which are neutral are the fewest, their number being negligible compared to the other two categories.

Furthermore, we are going to analyze how long the models need to be trained and give the output, in the same setup as mentioned in section 7.2. The fastest models to train were naive Bayes, random forest, SVM and MLP which took approximately 1 to 3 minutes before data augmentation. The deep learning model LSTM was slower to train and took more time. Its training time is almost the same with the one we mentioned in the aspect categorization results, i.e. 35 minutes. Even though we did not train the full RoBERTa model, even using it as an input with the dense layers on top took approximately 4 hours on a CPU. Last but not least, when using the pre-trained model RoBERTa as an input to the LSTM the model

took the same amount of time when training RoBERTa with the dense layers separately plus the time for the LSTM. The training time after data augmentation increased proportionally to the size of the datasets for both the machine and deep learning models.

Last but not least, we tested our models before and after data augmentation. We can see these results in the table 7.54. Again, like in aspect extraction task the results were approximately the same as before augmentation. Although the results did not change much, the models achieved the minimum validation loss during training much faster than before data augmentation. However, we should mention that the data augmentation slightly increased the accuracy in the Laptop domain.

Table 7.54: Sentiment Results after data augmentation

Models	**Restaurant Accuracy**	**Laptops Accuracy**
Naive Bayes	0.80	0.69
Random Forest	0.74	0.69
SVM with Tf-Idf	0.75	0.65
SVM with GloVe	0.80	0.71
MLP	0.73	0.64
LSTM with GloVe	0.80	0.70
LSTM with GloVe and Domain Embedding	0.82	0.71
RoBERTa	0.83	0.82
LSTM with RoBERTa	**0.88**	**0.84**
LSTM with GloVe and RoBERTa	0.87	0.81

Table 7.54: Sentiment Results after data augmentation

Models	**Restaurant Accuracy**	**Laptops Accuracy**
LSTM with GloVe and RoBERTa and Domain Embedding	0.87	0.83

In the previous section 7.4 we introduced a few models that we found online and reproduced in order to gain information about the performance of our models. All of them are using deep learning approaches and specifically, four of them are using attention mechanisms and one of them a capsule network. In the table 7.52 we can see the accuracy these models achieved. They were able to perform as well as the machine learning approaches that we used, and sometimes even better. However, the deep learning approaches we implemented were able to outperform their results in both domains. It is important to mention again that in order to be fair we used the exact same train, test and validation set on both the comparison and our models. Furthermore, we would like to mention that the external models as well as our models were not able to correctly predict the sentiment class neutral. Also, they had the best predictive power on the class positive. These results reinforce our assumption that this happened because of the nature of the datasets we used.

Having seen and discussed the results of sentiment analysis of our models and the comparison models and identified the best models and the advantages and disadvantages of each algorithm, now we are going to analyze another important element that we examined during our research. Using the accuracy as evaluation measure gives information about the performance of the models. In addition to that, we are going to examine the effectiveness of the algorithms on a specific category of sentences that we can characterize as “difficult”. They are those which contain more than one aspect category and the sentiment of the predicted aspect is not the same. We can see an example of such sentences in the table 7.55.

Table 7.55: Example of "difficult sentences"

Sentence	**Aspect**	**Polarity**
I liked the atmosphere very much but	Ambience	positive
the Food does not worth a price	Food	negative
It's a great Laptop, but right now it's	Laptop	positive
annoying to use the touch pad	Hardware	negative

When only one polarity category is included, it is easier for the models to predict it. But how do the models perform when the mentioned sentiment are conflicting? We noticed that the models were not able to perform very well on these sentences. Specifically, in the following table we can see how the best model performed in this category of sentences. We can see that only 60 percent of these sentences were correctly predicted in the Restaurant domain whereas in the Laptop domain the model was able to achieve only 44 percent accuracy.

Table 7.56: Performance of the best model and comparison models in the difficult sentences

	Accuracy rest	**Accuracy lap**
RoBERTa with LSTM	0.60	0.44
AE-LSTM	0.63	0.61
AT-LSTM	0.61	0.53
ATAE-LSTM	0.61	0.53
HEAT	0.51	0.61
Capsule network	0.60	0.53

We can see that the performance of the comparison models is approximately the same and some times slightly better in the Restaurant domain. However, in the Laptop domain the external models were able to perform much better. Most of the models are able to outperform ours in the category "difficult sentences" because they use more sophisticated ways to take into account the predicted aspect categories. However, all the models, ours and the comparison, were not able to

perform very well in that category of sentences. That might have been because in the training set the number of these sentences is not high, thus the models might have not been able to learn how to correctly classify them.

8. Business perspective

Nowadays, more and more people write their opinions about products and Services online. Companies are able to use these reviews to gain insights about customers' opinions. Businesses might spend a lot of time and human resources to manually extract these information. The goal of this research is to conduct aspect based sentiment analysis using machine and deep learning algorithms. In that way we can automate this procedure and help save time and money.

We can see in our datasets that the majority of the sentences contain one aspect. This is the case in real word as well. Extracting the aspect and the sentiment from sentences which contain only one statement would have been easier for our models. We could have used approximately the 80 percent of the sentences we have in our datasets and we would be able to provide more accurate results. The correctly predicted aspect categories are those mentioned in the section 7 by using the aforementioned models. However, apart from the one hundred percent correctly predicted aspects, we noticed that there was a significant, consistently steady number of sentences for which the models were able to successfully predict one less category than the true ones, without giving wrongly predicted labels. That means that, when for example a sentence contained two aspect categories the models was able to correctly predict one of them without predicting other categories in the same sentences. Those sentences could also be included in the useful output of the models and give insights about the content of a review without giving false information to those who would be responsible for interpreting them.

Table 8.1: Number of Predicted categories

	100% Correct	**One less**	**Rest**
LSTM	415	67	105
RoBERTa + LSTM	447	50	90

In the test set of the Restaurant domain there are 587 sentences. 484 of them are talking about one aspect, 86 for two and the rest of them, 3 or more. We have not calculated this extra category on one less accurate label in our results and only took into account the 100 percent predicted labels, but a Company could gain information from this extra category. In table 8.1 we can see some examples.

Moreover, another important conclusion to a business is the one we discussed in section 7.2. The best model for the aspect extraction was able to maintain the same distribution of the predicted topics as the one in the test set in both domains as shown in Figures 7.17 and 7.16. Even though the model was not able to perfectly predict the classes the Company is still able to gain information about the topics that the reviewers are talking about.

Furthermore, we mentioned that data augmentation was not able to help our models get better results. Yet, the models were able to reach the lowest validation loss much faster and as a result it would save time and money while training the models. That could save time to a Company in delivering the results.

An additional feature the machine learning models have and could be useful to a Company is their explainability. It is easier to understand how machine learning models work and how they produce their results. Additionally, there are some Python packages that are used to explain various models such the one we used for SVM and is presented in Figures 7.13, 7.14, 7.24 and 7.25. On the contrary deep learning models are not easily explainable and it is more complex to justify how they work.

9. Conclusion

Nowadays, online reviews have dramatically increased all over the internet. People write reviews about hotels, Services, Restaurants and all kind of products online aiming to inform other people. However, these reviews can be used in favor of companies which want to gain insights about customers opinion about them and correct their flaws.

During this research we are trying to extract important information using review datasets in the domain of Restaurants and Laptops. Sentiment analysis is not enough if we do not know the topics in which the reviews are referring to. That is the reason why we want to extract both the topic and the sentiment of each sentence in the dataset.

In order to achieve that we divided the problem of aspect based sentiment analysis into two tasks. The first one is aspect categorization and the second sentiment analysis based on the predicted aspects. For aspect categorization we tried probabilistic models for text classification but they did not give the results that we expected because of the nature of or dataset. Reviews are short and thus they do not contain many words. As a result the data that we feed to the models are sparse. The best results for this task were given by combining an LSTM neural network which is generally used for sequential data like text and the transformer model, RoBERTa, which is a language representation model, pre-trained on a very big corpus.

Sentiment analysis was the second sub-task in which machine learning models gave good results when accounting for their simplicity and easy implementation.

Nevertheless, the state of the art pre-trained model RoBERTa given as input to an LSTM neural network again was able to result in the highest accuracy. The main problem during polarity extraction was that the datasets that we used were imbalanced. Finally, for both tasks the size of the datasets played an important role to the results.

We showed that aspect based sentiment analysis, is a challenging task that can be approached with various machine learning, deep learning models and language models. Machine learning models are easier to implement and faster to train and their results are as not as bad as we expected but the deep learning and pre-trained models are able to achieve the highest accuracy.

Limitations during the research

The main limitations that we came across during out research were the imbalanced dataset. The aspect categories did not have the same size. Moreover, one of the included categories is very broad and general and thus the models misclassified sentences to this category. We faced the same problem during performing sentiment analysis. The majority of the sentences belonged to the category positive, fewer in the category negative and the class neutral had very few examples compared to the two other classes. That is the reason why all the models were unable to properly classify the sentences to this category. Additionally, the size of the datasets is not big enough to help the model to learn better. Despite the drawbacks of the datasets, it is one of the best labeled datasets that someone could use to perform aspect based sentiment analysis.

Future work

Online reviews contain opinions about products and Services. When people write online in order to express themselves it is very likely that they use emojis. Even though in our datasets we did not came across any, searching this aspect of the topic might be interesting. We could include the emojis as a combination of the

symbols that are used to create them, or even replace each emoji with its meaning. Even though we did not examine that topic during our research it might have been an interesting idea to explore during aspect based sentiment analysis.

The size of our datasets was not that large. That is the reason why we conducted data augmentation. However, we only used one method to perform it. This method was able to reduce the training time but not give better results. We could investigate other ways of data augmentation that would be able to achieve both better results and less training time. Moreover, data augmentation could be used in order to balance the categories in our dataset and examine how the models perform when that imbalance is removed.

Finally, in future research, more complex ways to insert the predicted labels as input to the sentiment classification models, for example with a capsule network, could be investigated in order to boost the algorithms and help them gain more information about it.

Bibliography

[1] Charu C. Aggarwal. *Neural Networks and Deep Learning. A Textbook*. Cham: Springer, 2018, p. 497. ISBN: 978-3-319-94462-3. DOI: 10.1007/978-3-319-94463-0.

[2] Jayesh Bapu Ahire. *The Artificial Neural Networks handbook: Part 1*. 2018. URL: https://medium.com/coinmonks/the-artificial-neural-networks-handbook-part-1-f9ceb0e376b4 (visited on 10/19/2020).

[3] Md Shad Akhtar, Asif Ekbal, and Pushpak Bhattacharyya. "Aspect Based Sentiment Analysis: Category Detection and Sentiment Classification for Hindi". In: *Computational Linguistics and Intelligent Text Processing*. Ed. by Alexander Gelbukh. Cham: Springer International Publishing, 2018, pp. 246–257. ISBN: 978-3-319-75487-1.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1409.0473.

[5] Atreya Basu, Carolyn Watters, and Michael Author. "Support Vector Machines for Text Categorization." In: Jan. 2003, p. 103. DOI: 10.1109/HICSS.2003.1174243.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation". In: *J. Mach. Learn. Res.* 3 (2003), pp. 993–1022. ISSN: 1532-4435. DOI: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993. URL: http://portal.acm.org/citation.cfm?id=944937.

[7] Piotr Bojanowski et al. “Enriching Word Vectors with Subword Information”. In: *CoRR* abs/1607.04606 (2016). arXiv: 1607.04606. URL: http://arxiv.org/abs/1607.04606.

[8] Qiuxing Chen, Lixiu Yao, and Jie Yang. “Short Text Classification Based on LDA Topic Model”. In: *2016 International Conference on Audio, Language and Image Processing (ICALIP)*. 2016, pp. 749–753. DOI: 10.1109/icalip.2016.7846525. URL: https://app.dimensions.ai/details/publication/pub.1093588426.

[9] Jiajun Cheng et al. “Aspect-Level Sentiment Classification with HEAT (HiErarchical ATtention) Network”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM ’17. Singapore, Singapore: Association for Computing Machinery, 2017, pp. 97–106. ISBN: 9781450349185. DOI: 10.1145/3132847.3133037. URL: https://doi.org/10.1145/3132847.3133037.

[10] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

[11] Corinna Cortes and Vladimir Vapnik. “Support-Vector Networks”. In: *Mach. Learn.* 20.3 (Sept. 1995), pp. 273–297. ISSN: 0885-6125. DOI: 10.1023/A:1022627411411. URL: https://doi.org/10.1023/A:1022627411411.

[12] *Deep Dive into Bidirectional LSTM*. 2019. URL: https://www.i2tutorials.com/deep-dive-into-bidirectional-lstm/ (visited on 10/29/2020).

[13] Gayatree Ganu, Noemie Elhadad, and Amélie Marian. “Beyond the Stars: Improving Rating Predictions using Review Text Content.” In: Jan. 2009.

[14] Gayatree Ganu, Noemie Elhadad, and Amélie Marian. “Beyond the Stars: Improving Rating Predictions using Review Text Content.” In: Jan. 2009.

[15] Athanasios Giannakopoulos et al. “Unsupervised Aspect Term Extraction with B-LSTM & CRF using Automatically Labelled Datasets”. In: *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity,*

Sentiment and Social Media Analysis. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 180–188. DOI: 10.18653/v1/W17-5224. URL: https://www.aclweb.org/anthology/W17-5224.

[16] 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*. WWW '16. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, 2016, pp. 507–517. ISBN: 9781450341431. DOI: 10.1145/2872427.2883037. URL: https://doi.org/10.1145/2872427.2883037.

[17] Geoffrey E. Hinton, A. Krizhevsky, and S. Wang. "Transforming Auto-Encoders". In: *ICANN*. 2011.

[18] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[19] Minqing Hu and Bing Liu. "Mining Opinion Features in Customer Reviews". In: *Proceedings of the 19th National Conference on Artifical Intelligence*. AAAI'04. San Jose, California: AAAI Press, 2004, pp. 755–760. ISBN: 0262511835.

[20] Jagadeesh Jagarlamudi, Hal Daumé, and Raghavendra Udupa. "Incorporating Lexical Priors into Topic Models". In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. EACL '12. Avignon, France: Association for Computational Linguistics, 2012, pp. 204–213. ISBN: 9781937284190.

[21] Qingnan Jiang et al. "A Challenge Dataset and Effective Models for Aspect-Based Sentiment Analysis". In: *EMNLP/IJCNLP*. 2019.

[22] M. Hari Krishna, K. Rahamathulla, and Ali Akbar. "A feature based approach for sentiment analysis using SVM and coreference resolution". In: *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)* (2017), pp. 397–399.

[23] Edda Leopold and Jörg Kindermann. "Text Categorization with Support Vector Machines. How to Represent Texts in Input Space?" In: *Mach. Learn.* 46.1-3 (2002), pp. 423–444. URL: http://dblp.uni-trier.de/db/journals/ml/ml46.html#LeopoldK02.

[24] Zhichao Li et al. "Automatic Extraction for Product Feature Words from Comments on the Web". In: *Information Retrieval Technology*. Ed. by Gary Geunbae Lee et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 112–123. ISBN: 978-3-642-04769-5.

[25] Y. Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *ArXiv* abs/1907.11692 (2019).

[26] Chong Long, Jie Zhang, and Xiaoyan Zhu. "A Review Selection Approach for Accurate Feature Rating Estimation". In: *Coling 2010: Posters*. Beijing, China: Coling 2010 Organizing Committee, Aug. 2010, pp. 766–774. URL: https://www.aclweb.org/anthology/C10-2088.

[27] dProgrammer lopez. *RNN, LSTM GRU*. 2019. URL: http://dprogrammer.org/rnn-lstm-gru (visited on 10/18/2020).

[28] Huaishao Luo et al. "Improving Aspect Term Extraction With Bidirectional Dependency Tree Representation". In: *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 27.7 (July 2019), pp. 1201–1212. ISSN: 2329-9290. DOI: 10.1109/TASLP.2019.2913094. URL: https://doi.org/10.1109/TASLP.2019.2913094.

[29] Maxime. *What is a Transformer?* 2019. URL: https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04 (visited on 10/25/2020).

[30] Konstantin Lopuhin Revision 2497ec37 Mikhail Korobov. *Welcome to ELI5's documentation!* 2016. URL: https://eli5.readthedocs.io/en/latest/index.html#welcome-to-eli5-s-documentation (visited on 11/16/2020).

[31] Mohamad Syahrul Mubarok and Muhammad Dwi Adiwijaya Aldhi. "Aspect-based sentiment analysis to review products using Naıve Bayes". In: *International Conference on Mathematics: Pure, Applied and Computation: Empowering Engineering using Mathematics*. Vol. 1867. American Institute of Physics Conference Series. Aug. 2017, p. 020060. DOI: 10.1063/1.4994463.

[32] A. Nazir et al. "Issues and Challenges of Aspect-based Sentiment Analysis: A Comprehensive Survey". In: *IEEE Transactions on Affective Computing* (2020), pp. 1–1. DOI: 10.1109/TAFFC.2020.2970399.

[33] A. Onan, Serdar Korukoglu, and H. Bulut. "LDA-based Topic Modelling in Text Sentiment Classification: An Empirical Analysis". In: *Int. J. Comput. Linguistics Appl.* 7 (2016), pp. 101–119.

[34] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[35] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[36] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[37] Minh Hieu Phan and Philip O. Ogunbona. "Modelling Context and Syntactical Features for Aspect-based Sentiment Analysis". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 3211–3220. DOI: 10.18653/v1/2020.acl-main.293. URL: https://www.aclweb.org/anthology/2020.acl-main.293.

[38] Maria Pontiki et al. "SemEval-2014 Task 4: Aspect Based Sentiment Analysis". In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Lin-

guistics, Aug. 2014, pp. 27–35. DOI: 10.3115/v1/S14-2004. URL: https://www.aclweb.org/anthology/S14-2004.

[39] Maria Pontiki et al. "SemEval-2015 Task 12: Aspect Based Sentiment Analysis". In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, June 2015, pp. 486–495. DOI: 10.18653/v1/S15-2082. URL: https://www.aclweb.org/anthology/S15-2082.

[40] Maria Pontiki et al. "SemEval-2016 Task 5: Aspect Based Sentiment Analysis". In: *10th International Workshop on Semantic Evaluation (SemEval 2016)*. San Diego, United States, June 2016. URL: https://hal.archives-ouvertes.fr/hal-02407165.

[41] Soujanya Poria, Erik Cambria, and Alexander Gelbukh. "Aspect extraction for opinion mining with a deep convolutional neural network". In: *Knowledge-Based Systems* 108 (2016). New Avenues in Knowledge Bases for Natural Language Processing, pp. 42–49. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2016.06.009. URL: http://www.sciencedirect.com/science/article/pii/S0950705116301721.

[42] Peng Qi et al. *Universal Dependency Parsing from Scratch*. 2019. arXiv: 1901.10457 [cs.CL].

[43] *roberta-large - Hugging Face*. URL: https://huggingface.co/roberta-large (visited on 12/10/2020).

[44] Sebastian Ruder. "Neural Transfer Learning for Natural Language Processing". PhD thesis. National University of Ireland, Galway, 2019.

[45] Afan Salman et al. "Single Layer Multi-layer Long Short-Term Memory (LSTM) Model with Intermediate Variables for Weather Forecasting". In: *Procedia Computer Science* 135 (Jan. 2018), pp. 89–98. DOI: 10.1016/j.procs.2018.08.153.

[46] Christopher Scaffidi et al. "Red Opal: Product-Feature Scoring from Reviews". In: *Proceedings of the 8th ACM Conference on Electronic Commerce*. EC '07. San Diego, California, USA: Association for Computing Machinery, 2007, pp. 182–191. ISBN: 9781595936530. DOI: 10.1145/1250910.1250938. URL: https://doi.org/10.1145/1250910.1250938.

[47] K. Schouten and F. Frasincar. "Survey on Aspect-Level Sentiment Analysis". In: *IEEE Transactions on Knowledge and Data Engineering* 28.3 (2016), pp. 813–830. DOI: 10.1109/TKDE.2015.2485209.

[48] K. Schouten and F. Frasincar. "Survey on Aspect-Level Sentiment Analysis". In: *IEEE Transactions on Knowledge and Data Engineering* 28 (2016), pp. 813–830.

[49] Lei Shu, Hu Xu, and Bing Liu. "Lifelong Learning CRF for Supervised Aspect Extraction". In: *CoRR* abs/1705.00251 (2017). arXiv: 1705.00251. URL: http://arxiv.org/abs/1705.00251.

[50] Lei Shu et al. "Supervised Opinion Aspect Extraction by Exploiting Past Extraction Results". In: *CoRR* abs/1612.07940 (2016). arXiv: 1612.07940. URL: http://arxiv.org/abs/1612.07940.

[51] Carson Sievert and Kenneth Shirley. "LDAvis: A method for visualizing and interpreting topics". In: June 2014. DOI: 10.13140/2.1.1394.3043.

[52] Ge Song et al. "Short Text Classification: A Survey". In: *Journal of Multimedia* 9 (May 2014). DOI: 10.4304/jmm.9.5.635-643.

[53] Chi Sun, Luyao Huang, and Xipeng Qiu. *Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence*. 2019. arXiv: 1903.09588 [cs.CL].

[54] Tin Kam Ho. "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1. 1995, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994.

[55] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[56] Vikash Singh. *GuidedLDA: Guided Topic modeling with latent Dirichlet allocation*. http://https://guidedlda.readthedocs.io/en/latest/#license.

[57] Yequan Wang et al. "Attention-based LSTM for Aspect-level Sentiment Classification". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 606–615. DOI: 10.18653/v1/D16-1058. URL: https://www.aclweb.org/anthology/D16-1058.

[58] Thomas Wolf et al. "HuggingFace's Transformers: State-of-the-art Natural Language Processing". In: *ArXiv* abs/1910.03771 (2019).

[59] Hu Xu et al. "Double Embeddings and CNN-based Sequence Labeling for Aspect Extraction". In: *ACL*. 2018.

[60] Kaustubh Yadav. *A Comprehensive Survey on Aspect Based Sentiment Analysis*. 2020. arXiv: 2006.04611 [cs.CL].

[61] Xiaohui Yan et al. "A Biterm Topic Model for Short Texts". In: *Proceedings of the 22nd International Conference on World Wide Web*. WWW '13. Rio de Janeiro, Brazil: Association for Computing Machinery, 2013, pp. 1445–1456. ISBN: 9781450320351. DOI: 10.1145/2488388.2488514. URL: https://doi.org/10.1145/2488388.2488514.

[62] Hsiang-fu Yu et al. *LibShortText: A Library for Short-text Classification and Analysis LibShortText: A Library for Short-text Classification and Analysis*.

[63] Wayne Xin Zhao et al. "Jointly Modeling Aspects and Opinions with a MaxEnt-LDA Hybrid". In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. EMNLP '10. Cambridge, Massachusetts: Association for Computational Linguistics, 2010, pp. 56–65.

A. Confusion Matrices for augmented data

In this section of appendix we are going to include the resulting confusion matrices from sentiment analysis sub-task for augmented data.

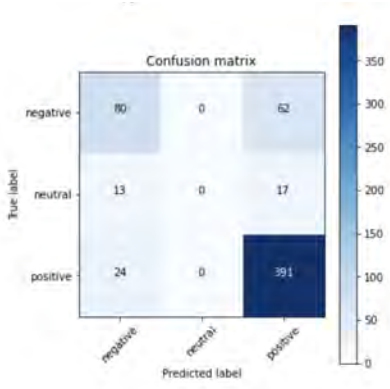


Figure A.1: Restaurant domain naive Bayes

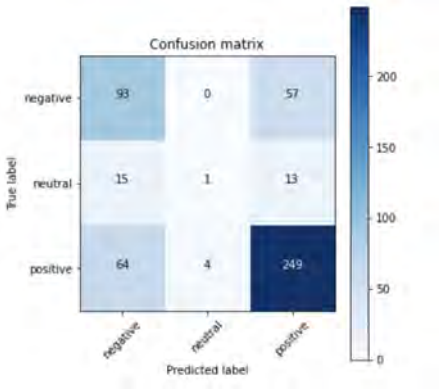


Figure A.2: Laptop domain naive Bayes

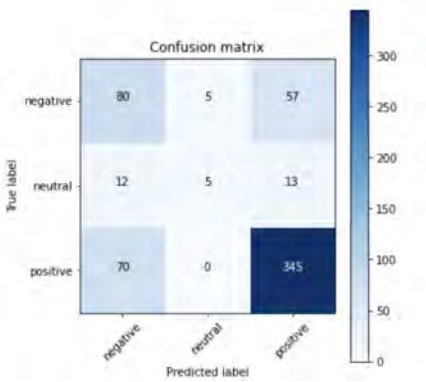


Figure A.3: Restaurant domain random forest

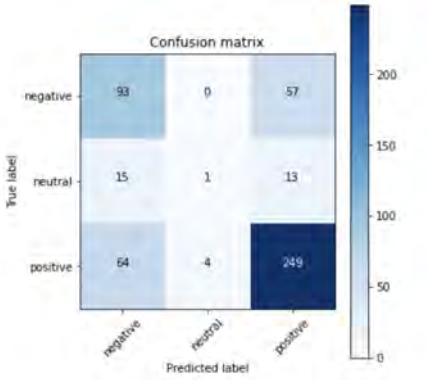


Figure A.4: Laptop domain random forest

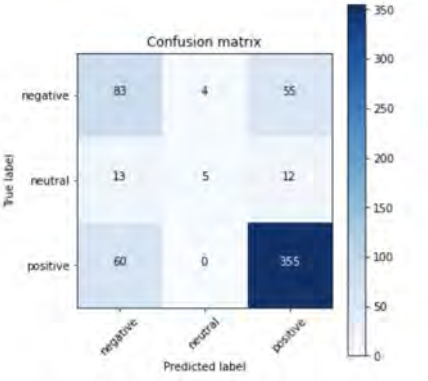


Figure A.5: Confusion matrix for Restaurants with SVM

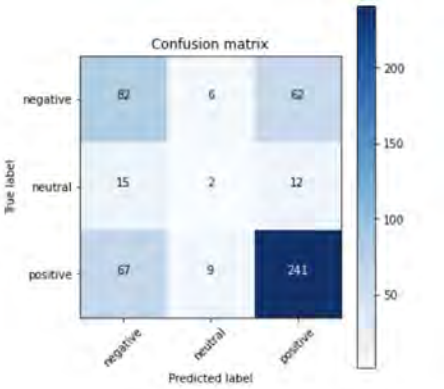


Figure A.6: Confusion matrix for Laptops with SVM

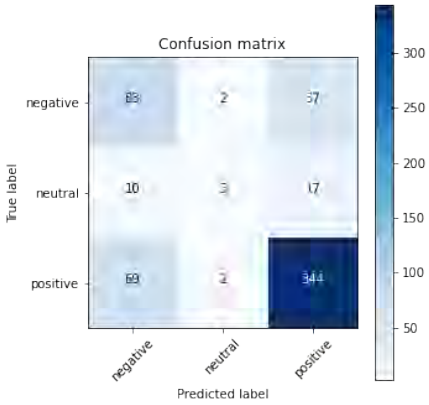


Figure A.7: Confusion matrix for Restaurants with MLP

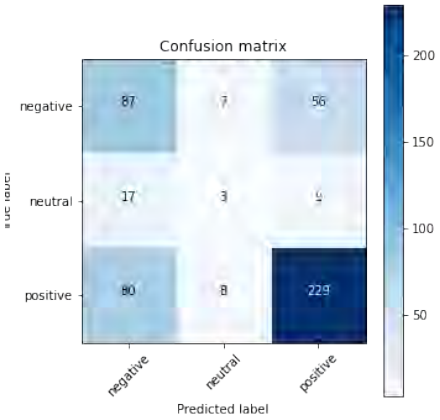


Figure A.8: Confusion matrix for Laptops with MLP

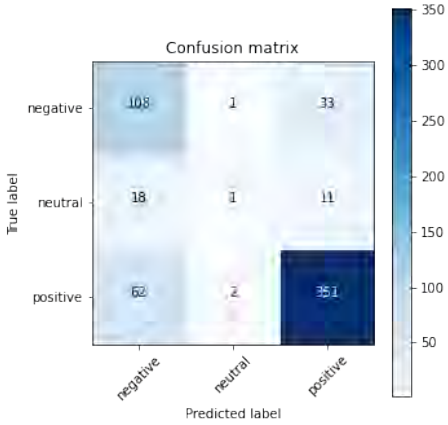


Figure A.9: Confusion matrix for Restaurants LSTM with GloVe

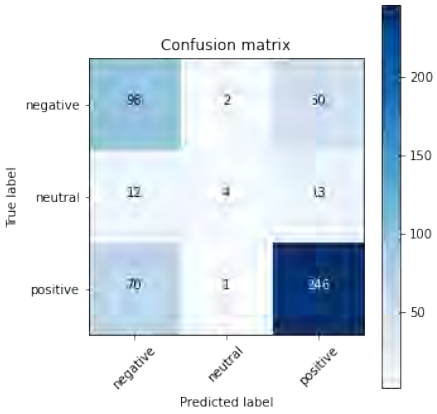


Figure A.10: Confusion matrix for Laptops LSTM with GloVe

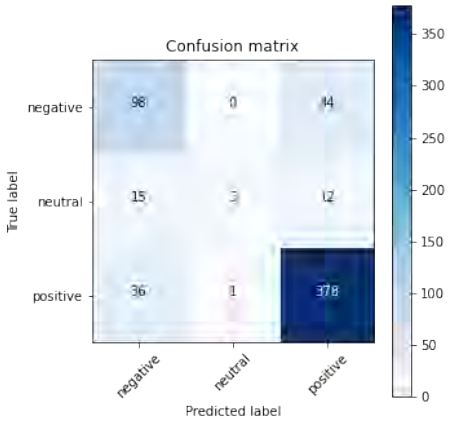


Figure A.11: Confusion matrix for Restaurant with LSTM, GloVe and domain embedding

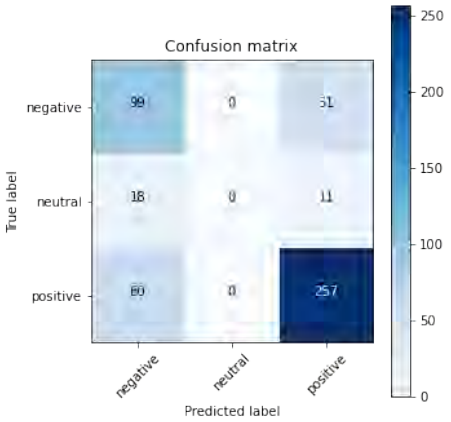


Figure A.12: Confusion matrix for Laptops with LSTM, GloVe and Domain embedding

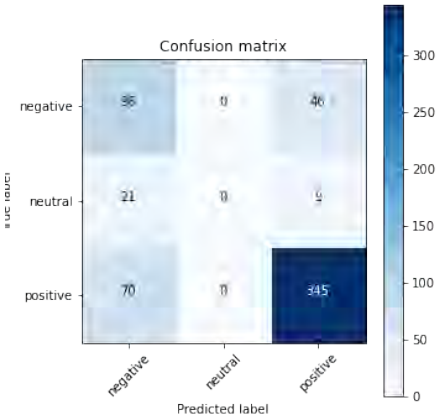


Figure A.13: LSTM with Tf-Idf for Restaurant domain

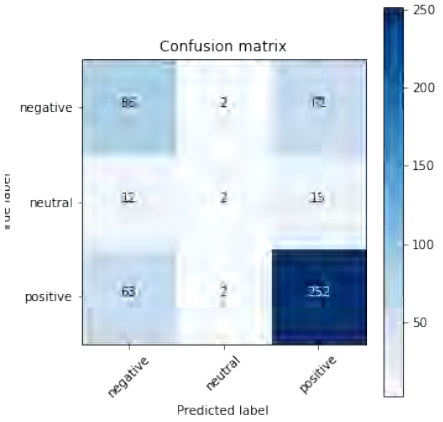


Figure A.14: LSTM with Tf-Idf for Laptop domain

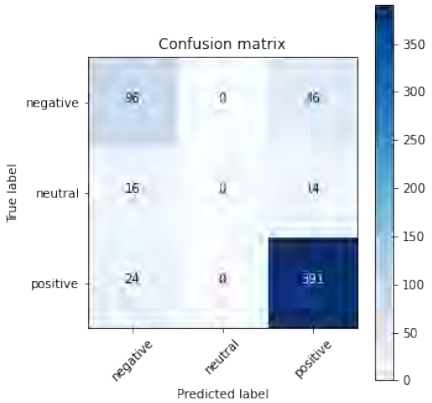


Figure A.15: Confusion matrix for Restaurants with RoBERTa

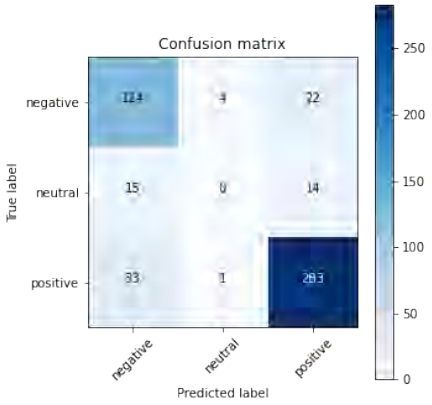


Figure A.16: Confusion matrix for Laptops for augmented data

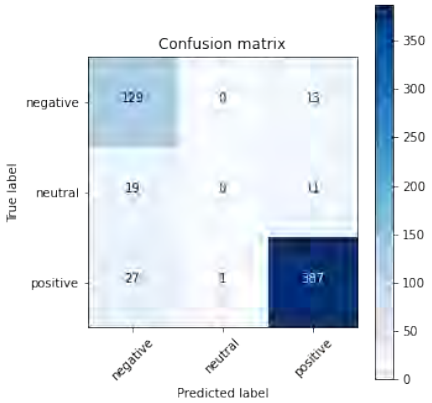


Figure A.17: Confusion matrix for Restaurants with RoBERTa and LSTM

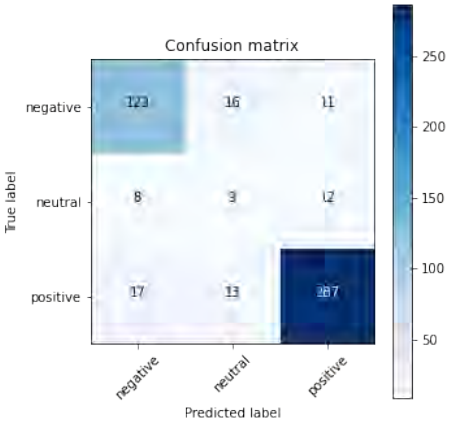


Figure A.18: Confusion matrix for Laptops with RoBERTa and LSTM

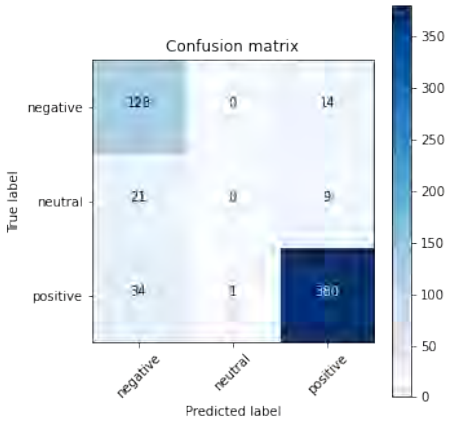


Figure A.19: Confusion matrix for Restaurants with RoBERTa and LSTM and GloVe

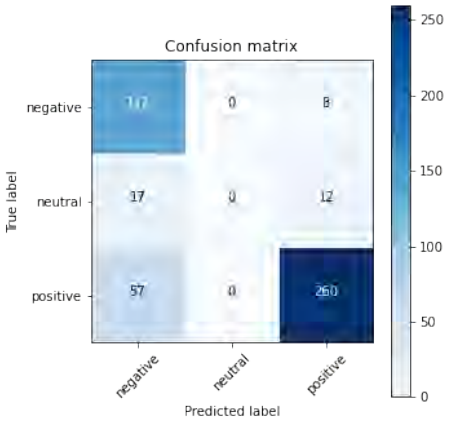


Figure A.20: Confusion matrix for Laptops with RoBERTa and LSTM and GloVe

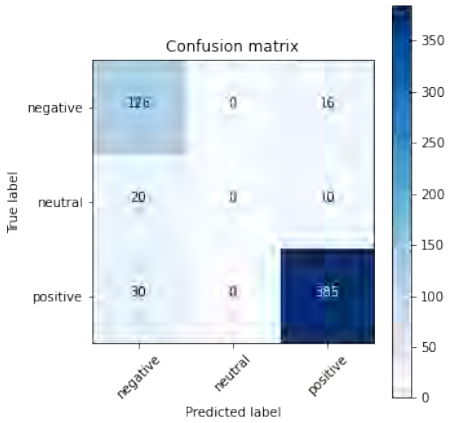


Figure A.21: Confusion matrix for Restaurants with LSTM and RoBERTa, GloVe and domain embeddings

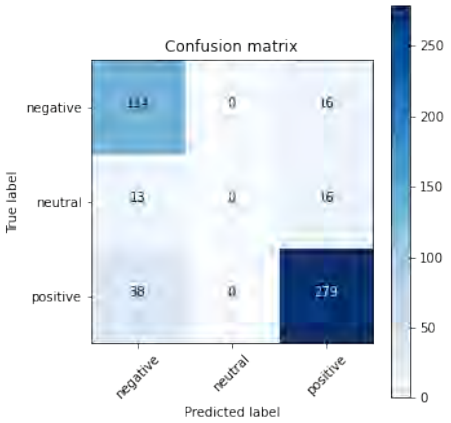


Figure A.22: Confusion matrix for Laptops with LSTM and RoBERTa, GloVe and domain embeddings

B. Comparison Models Confusion Matrices

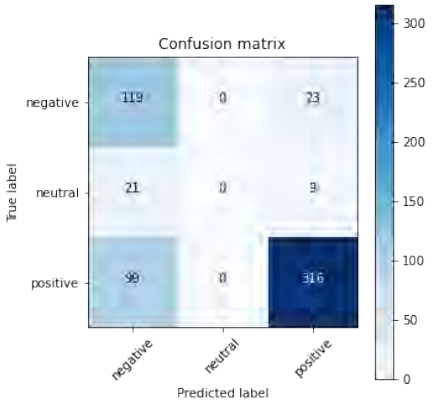


Figure B.1: Restaurant Domain AE-LSTM

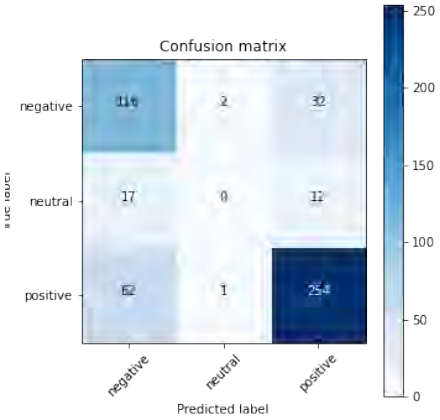


Figure B.2: Laptop Domain AE-LSTM

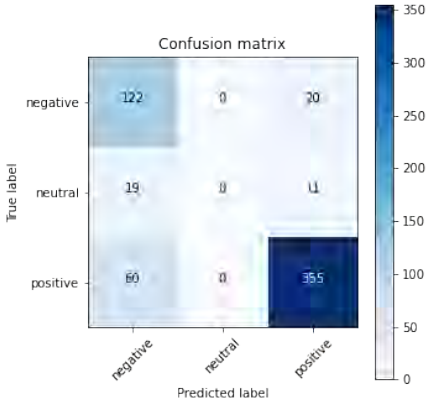


Figure B.3: Restaurant Domain AT-LSTM

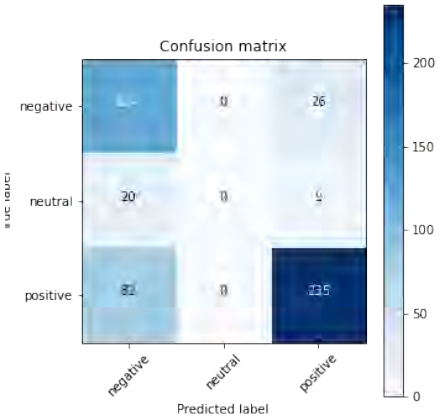


Figure B.4: Laptop Domain AT-LSTM

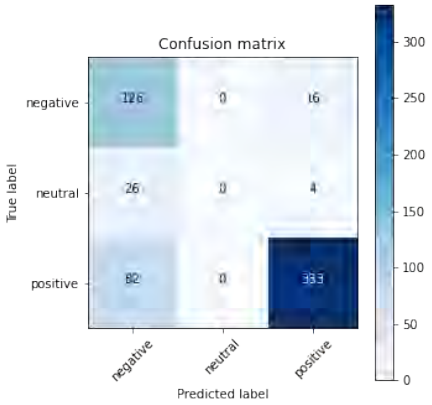


Figure B.5: Restaurant Domain ATAE-LSTM

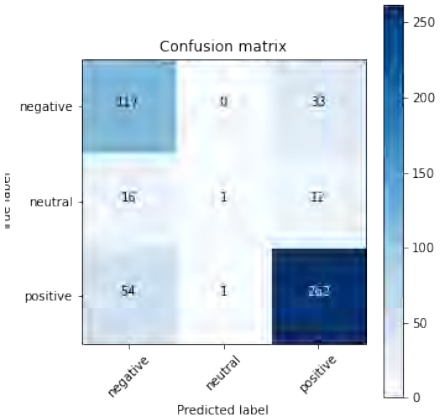


Figure B.6: Laptop Domain ATAE-LSTM

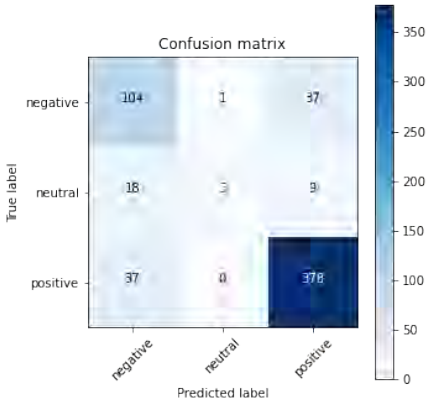


Figure B.7: Restaurant Domain HEAT

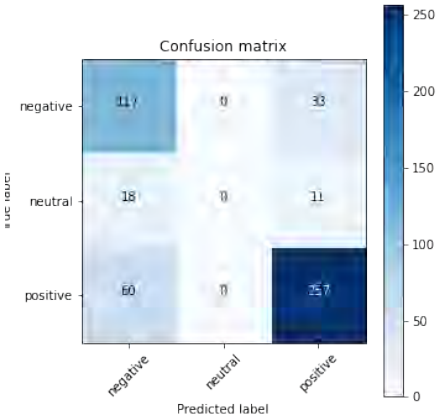


Figure B.8: Laptop Domain HEAT

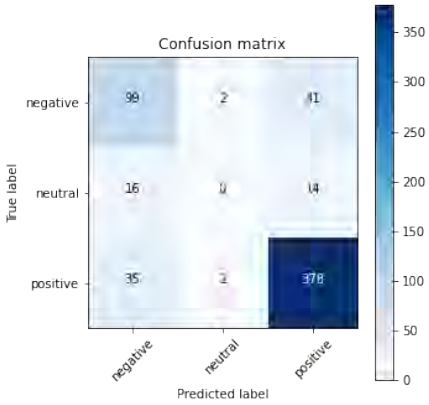


Figure B.9: Restaurant Domain Capsule Network

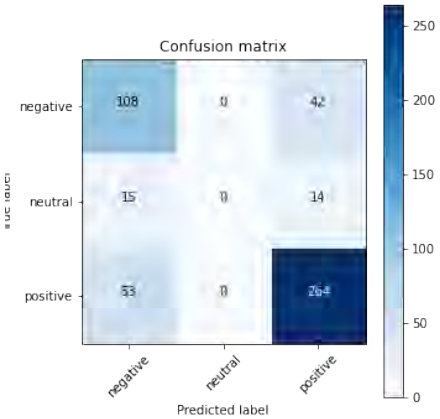


Figure B.10: Laptop Domain Capsule Network