UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL DE CALCULATOARE

PROIECT DE DIPLOMĂ

Analiza de sentimente bazată pe aspecte pentru recenzii online

Mihail Oprea

Coordonator științific:

Șl. Dr. Ing. Ciprian Octavian Truică
Conf. Dr. Ing. Elena Simona Apostol

BUCUREȘTI
2024

UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

DIPLOMA PROJECT

Aspect-Based Sentiment Analysis for Online Reviews

Mihail Oprea

Thesis advisor:

Șl. Dr. Ing. Ciprian Octavian Truică
Conf. Dr. Ing. Elena Simona Apostol

BUCHAREST
2024

CONTENTS

SINOPSIS

Analiza sentimentelor bazată pe aspecte (ABSA) este un task important în procesarea limbajului natural (NLP) care își propune să identifice polaritatea sentimentelor asociate cu aspecte specifice ale unei entități țintă într-un text. Acest studiu investighează integrarea modelului transformer BART cu arhitecturi mai simple de rețele neuronale, cum ar fi straturile BiLSTM și Conv1D, pentru a îmbunătăți performanța sarcinilor ABSA. Abordarea noastră se concentrează pe două subtask-uri principale: Extracția Categoriilor de bazată pe Aspect (ACE) și Extracția Sentimentului bazată pe Aspect (ASE). Pentru a aborda provocările date de mediile cu resurse reduse și absența extracției termenilor de aspect, modelul nostru utilizează înțelegerea contextuală a BART combinată cu eficiența structurală a straturilor BiLSTM și Conv1D. Analizăm impactul reducerii complexității sarcinii ACE prin reducerea numărului de clase de etichete la cele principale (în loc de "FOOD#QUALITY" avem "FOOD"), ceea ce îmbunătățește semnificativ metrici de performanță precum acuratețea, precizia, recall-ul și scorul F1. Experimentele realizate pe setul de date SemEval2016 Task 5 Restaurants demonstrează că modelul nostru integrat poate atinge performanțe competitive cu o sarcină computațională redusă. Studiul evidențiază, de asemenea, dificultățile în compararea directă a abordării noastre impartite a ABSA cu alte soluții de ultimă generație din cauza diviziunii noastre pentru sarcina ABSA.

Cuvinte cheie: transformeri, extracția categoriilor bazată pe aspect, extracția sentimentului bazată pe aspect, rețele neuronale

ABSTRACT

Aspect-Based Sentiment Analysis (ABSA) is an important task in Natural Language Processing (NLP) that aims to identify the sentiment polarity associated with specific aspects of a target entity within a text. This study investigates the integration of the BART transformer model with simpler neural network architectures, such as BiLSTM and Conv1D layers, to enhance the performance of ABSA tasks. Our approach focuses on two main subtasks: Aspect Category Extraction (ACE) and Aspect Sentiment Extraction (ASE). To address the challenges of low-resource environments and the absence of aspect term extraction, our model uses the contextual understanding of BART combined with the structural efficiency of BiLSTM and Conv1D layers. We analyze the impact of reducing the complexity of the ACE task by reducing the number of label classes to its main ones (instead of "FOOD#QUALITY" we have "FOOD"), which significantly improves performance metrics such as accuracy, precision, recall, and F1 score. The experiments conducted on the SemEval 2016 Task 5 Restaurants dataset demonstrate that our integrated model can achieve competitive performance with reduced computational overhead. The study also highlights the difficulties in directly comparing our segmented ABSA approach with other state-of-the-art solutions due to our division for the ABSA task.

Keywords: transformers, aspect category extraction, aspect sentiment extraction, neural networks

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation and gratitude towards the support received from my coordinators who guided me in the process of realizing this paper.

1 INTRODUCTION

Aspect-based sentiment analysis (ABSA) is a specialized field within sentiment analysis that focuses on identifying the polarity of opinions expressed towards specific aspects of a given target. The primary objective of ABSA is to determine pairs of entities (E) and attributes (A) mentioned in each sentence and to ascertain the sentiment (positive, negative, or neutral) associated with these pairs. This detailed level of analysis allows for a more contextual understanding of sentiment in the text compared to general sentiment analysis, which typically provides an overall sentiment score for entire documents or reviews.

ABSA is a complex task, so to ease its complexity we propose to divide it into two main subtasks: Aspect Category Extraction (ACE) and Aspect Sentiment Extraction (ASE).

Aspect Category Extraction (ACE): This subtask involves identifying the relevant category or aspect of the text input. For example, in a restaurant review, categories might include food, service, ambiance, and price. The goal of ACE is to classify the input text into one or more of these predefined categories.

Aspect Sentiment Extraction (ASE): Once the categories are identified, ASE focuses on determining the sentiment or polarity of the text related to each category. Using the same restaurant review example, ASE would analyze the sentiment towards the food, service, ambiance, and price, identifying whether the sentiments are positive, negative, or neutral.

Given the complexity of the ABSA task to achieve the best results possible in this kind of scenario, state-of-the-art solutions are required. All BART, BERT (Bidirectional Encoder Representations from Transformers), and GPT (Generative Pre-trained Transformer) are based on transformer architectures, the difference being that BERT uses only an Encoder, GPT uses only a Decoder, while BART uses both an Encoder and a Decoder. More BART and BERT are bidirectional while GPT is unidirectional. All are state-of-the-art solutions for Natural Language Processing (NLP) tasks. These transformer-based models excel at capturing contextual information and handling complex language structures, making them highly effective for both ACE and ASE tasks.

This thesis explores the implementation of a model architecture that integrates a BiLSTM (Bidirectional Long Short-Term Memory) network and a Conv1D (one-dimensional convolutional) layer with a BART transformer model for text processing. This approach aims to leverage the strengths of both sequential and convolutional neural networks, along with the powerful contextual understanding provided by BART, to enhance the performance of ABSA tasks.

1.1 Motivation

The motivation for this research stays in the desire to gain firsthand experience with state-of-the-art solutions for natural language processing (NLP) tasks, specifically using models like BART. So, we were really interested in exploring how cutting-edge transformer models can be effectively combined with simpler neural network architectures for such a complex task as the ABSA task.

One of the interesting aspects of this research is the decision to exclude aspect term extraction from the ABSA task. Typically, extracting terms associated with specific aspects helps in making more accurate predictions as it provides additional contextual information. Choosing to eliminate this part for the category and polarity prediction, would translate into a higher complexity for the handling of sentiment analysis which is a challenge that we want to face. We also acknowledge that such a decision will make a harder comparison with other state-of-the-art solution for the ABSA task.

Through this research, I aim to show how combining advanced models like BART with a simpler custom machine-learning architecture can lead to surprising results for the ABSA task.

1.2 Objectives

This research topic aims to design new Natural Language Processing, Machine Learning, and Deep Learning techniques to determine accurately the fine-grained opinion polarity from online reviews. This involves the following objectives:

1. **Integration of Advanced and Simple Models**:
    - *Combining BART with BiLSTM and Conv1D*: To explore the efficacy of integrating the BART transformer model, known for its powerful contextual understanding, with simpler neural network architectures like BiLSTM (Bidirectional Long Short-Term Memory) and Conv1D (one-dimensional convolutional) layers. This combination aims to leverage the strengths of both advanced and simple models, potentially leading to a more efficient and accurate ABSA system.
2. **Aspect-Based Sentiment Analysis (ABSA) Task Division**:
    - *Separating ACE and ASE Tasks*: To investigate the impact of dividing the ABSA task into Aspect Category Extraction (ACE) and Aspect Sentiment Extraction (ASE). This division implies a reduced complexity of the ABSA task..
3. **Evaluation of Model Performance Without Aspect Term Extraction**:
    - *Testing Model Robustness*: To evaluate how well the integrated model performs without the aid of aspect term extraction. This setup will test the model's ability to

accurately predict category and polarity based purely on the contextual information provided by the BART model and the structural features captured by BiLSTM and Conv1D layers.

4. **Performance Comparison with Reduced Complexity for ACE**:
    - *Reducing Number of Classes*: To see the performance differences when the complexity of the category label is reduced to its more general ones. This objective aims to demonstrate the benefits of reduced complexity in improving model accuracy, precision, recall, and F1 score for category classification task to a more general classification.

By achieving these objectives, we aim to contribute by providing a more in-depth understanding of how advanced and simple models can be combined in an effective way. The research in this paper is expected to offer valuable information about the findings and the reached conclusions, particularly in environments with limited resources like the SemEval2016 Task Restaurants dataset.

1.3 Structure of the thesis

The thesis is structured into 6 main chapters. Chapter **1**, is the introduction of this paper and explains the main task of this paper. In chapter **2** are presented other state-of-the-art solutions for the ABSA tasks from different publishers. Chapter **3** presents the pipeline architecture for my project step by step from start to finish, while chapter **4** presents how they were implemented. In the chapter **5** we have the results of the implemented solution and some information about the setup and the dataset used for testing. Finally, we have the conclusion in chapter **6**.

2 STATE OF THE ART

In this chapter, we present some state-of-the-art solutions for the Aspect-Based Sentiment Analysis (ABSA) task.

The main scope of this document Apostol et al. [3] is to address the complexities of the ABSA task by proposing a new model named ATESA-BÆRT. This model improves the granularity of sentiment analysis by dividing the task into two sub-tasks: Aspect Term Extraction (ATE) and Aspect Term Sentiment Analysis (ATSA). Utilizing six models on each branch (2 in total), ATESA-BÆRT can effectively handle multiple aspects within the same review. The methodology includes the use of BERT and BART transformers, fine-tuning, and an ensemble learning approach to enhance accuracy and efficiency. Experimental results demonstrate that ATESA-BAERT significantly outperforms existing models, providing more accurate aspect identification and sentiment classification. High computational times are expected.

In this article Chenhua et al. [4] is presented a method to enhance the ABSA task by proposing a model that uses discrete opinion trees instead of traditional dependency trees. This model addresses the limitations of dependency parsers, especially for low-resource languages and domains. The proposed method induces opinion trees from attention scores derived from BERT representations, which are then used to model the structural relations via Graph Convolutional Networks (GCNs). This approach not only improves interpretability but also demonstrates competitive performance across six English datasets and additional ones in Chinese and Korean. The results show that the model achieves superior accuracy and efficiency.

Article Hang et al. [5] proposes a solution for ABSA by unifying the diverse ABSA subtasks into a single generative framework. The authors redefine each subtask as a sequence generation problem using pointer and sentiment class indexes, allowing the use of a pre-trained sequence-to-sequence model, BART, to address all subtasks in an end-to-end manner. This unified framework can handle Aspect Term Extraction (ATE), Opinion Term Extraction (OTE), Aspect-Level Sentiment Classification (ALSC), Aspect-Oriented Opinion Extraction (AOE), Aspect Term Extraction and Sentiment Classification (AESC), Pair Extraction, and Triplet Extraction. Experiments across four datasets demonstrate that the proposed model outperforms state-of-the-art approaches, offering significant performance gains.

Article Akbar et al. [1] enhances the performance of BERT in ABSA tasks through the introduction of two simple modules: Parallel Aggregation (P-SUM) and Hierarchical Aggregation (H-SUM). These modules leverage the intermediate layers of BERT to improve

Aspect Extraction (AE) and Aspect Sentiment Classification (ASC). P-SUM aggregates information from the last four layers of BERT in parallel, while H-SUM uses a hierarchical approach inspired by Feature Pyramid Networks (FPNs). Both models employ Conditional Random Fields (CRFs) for AE to capture the dependencies in sequence labeling. Extensive experiments on SemEval 2014 and 2016 datasets demonstrate that these modules outperform baseline models, including BERT-PT, in terms of F1 and Macro-F1 scores.

The main scope of document Shaowei et al. [10] is to enhance the extraction of aspect-opinion pairs in fine-grained opinion mining by proposing a novel model called the Synchronous Double-channel Recurrent Network (SDRN). This model integrates an opinion entity extraction unit and a relation detection unit into a double-channel network, enabling the simultaneous extraction of opinion entities and their relations. BERT is adopted as an encoding layer. The synchronization unit, comprising the Entity Synchronization Mechanism (ESM) and Relation Synchronization Mechanism (RSM), is designed to enhance the mutual benefit of the two channels. Tests on manually built datasets from SemEval 2014 and 2015 demonstrate that SDRN achieves state-of-the-art performance, significantly outperforming existing methods in terms of accuracy and efficiency.

The goal of article Kai et al. [7] is to utilize syntax details with a new model known as the Relational Graph Attention Network (R-GAT). The authors address the limitations of attention-based models, which often fail to accurately connect aspects with their corresponding opinion words due to language complexity and multiple aspects within a single sentence. They propose constructing an aspect-oriented dependency tree by reshaping and pruning an ordinary dependency parse tree to root it at the target aspect, facilitating a more precise connection between aspects and opinion words. R-GAT encodes these trees by generalizing graph attention networks to include labeled edges, thereby enhancing the model's ability to capture syntactic dependencies. Extensive experiments on the SemEval 2014 and Twitter datasets show that R-GAT significantly improves the performance of graph attention networks, setting a new state-of-the-art solution.

This paper Hu et al. [6] introduces DomBERT, a domain-oriented extension of BERT. DomBERT is designed to address the limitations of general-purpose language models, which often fail to capture domain-specific nuances, especially in low-resource settings. The model achieves this by simultaneously learning from both in-domain and relevant domain corpora, leveraging domain embeddings to identify and incorporate knowledge from related domains. Experiments conducted on various ABSA tasks, including aspect extraction (AE), aspect sentiment classification (ASC), and end-to-end ABSA (E2E-ABSA), demonstrate that DomBERT outperforms baseline models, including BERT and BERT-Review, in terms of F1 and accuracy scores. applications.

The scope of article Zhen et al. [11] is to propose a new tagging scheme, Grid Tagging Scheme (GTS), for end-to-end Aspect-oriented Fine-grained Opinion Extraction (AFOE). The proposed GTS addresses the limitations of traditional pipeline approaches, which often suffer from error propagation, by transforming the extraction of opinion pairs (aspect term, opinion

term) and opinion triplets (aspect term, opinion term, sentiment) into a unified tagging task. The authors implement GTS using CNN, BiLSTM, and BERT models, demonstrating that GTS achieves state-of-the-art performance on various datasets, significantly outperforming existing methods. Additionally, GTS includes an inference strategy to exploit mutual indications between different opinion factors for more accurate extractions.

In this study Qingnan et al. [9] is presented a new large-scale dataset called Multi-Aspect Multi-Sentiment (MAMS) designed to advance research in ABSA. Unlike existing datasets, MAMS includes sentences with at least two aspects having different sentiment polarities, making it more challenging and preventing the task from turning into sentence-level sentiment analysis. The authors propose effective models, CapsNet and CapsNet-BERT, by combining it with BERT, to handle the complexities of this dataset. Experimental results show that these models outperform state-of-the-art baselines on the MAMS and SemEval-2014 Restaurant datasets. The complexity of the dataset and CapsNet-BERT can be translated into more computational resources

The role of this research Maria et al. [8] is to present the third edition of the ABSA task, which is part of the SemEval competitions. The 2016 task builds on previous years, providing 19 training and 20 testing datasets across 8 languages and 7 domains, and introducing text-level ABSA. This task aims to extract opinions on specific entities and their aspects from the text. The task is divided into three subtasks: sentence-level ABSA, text-level ABSA, and out-of-domain ABSA, with evaluations conducted on aspect category detection, opinion target extraction, and sentiment polarity classification. The task attracted significant participation, with 245 submissions from 29 teams. A negative aspect of this paper is the complexity of the annotation process, which requires significant manual effort.

This paper Amir et al. [2] proposes a model that integrates gated graph convolutional networks (GCNs) with syntax-based regulation. This model addresses two main issues in prior graph-based ABSA models: the lack of customization of hidden representation vectors for aspect terms and the failure to utilize the overall importance scores of words from dependency trees. The authors introduce gate vectors derived from aspect term representation vectors to refine GCN hidden vectors, and they propose a mechanism to obtain and integrate syntax-based importance scores for each word to improve representation vectors for ABSA. Three key datasets were used for testing, where the proposed model achieves state-of-the-art performance. Here, gating and regulation mechanisms can lead to increased complexity and computation.

3 METHODOLOGY

This chapter aims to create an overall picture of the architecture used in developing the sentiment prediction model for online reviews. It will discuss the model architecture, the data preparation process including sentence splitting into clauses using the BART model, and the web application developed for model interaction.

In this chapter, we highlight the methodology used in the development and implementation of our Aspect-Based Sentiment Analysis (ABSA) solution. The chapter is structured into two main sections: Algorithms/Models and Architecture. Each section provides an overview of the processes and structures used to achieve the project's objectives.

The Algorithms/Models section presents an overview look at the various algorithms and models utilized throughout the project, focusing on their roles in data processing and model training, while the Architecture section provides an overview of the overall architecture of our model (the pipeline architecture). For more detailed and in-depth presentations for our solution is presented in 4 chapter.

3.1 Algorithms/Models

In this section, we discuss the algorithms and models used in different moments of the implementation of the project. This includes the use of the ChatGPT API for sentence splitting, the BART model for vectorizing clauses, and the final sentiment prediction model.

3.1.1 Alghoritms for data processing

We use the ChatGPT API for splitting sentences into clauses because of the complexity of the sentences that some can have so that would be reduced, but this isn't the only reason. The other reason is that in a more complex sentence, multiple pairs of category and sentiment can be found, so to cover each category and sentiment they must represent different parts of the sentence because it would make no sense to train the model with the same sentences or the same parts of it represented by clauses for different categories and polarities.

After splitting the sentences into clauses, each clause needs to be vectorized to be eligible to be used as input data to feed the classification solution. For this task, we use BART from

Hugging Face which is known to be a good choice for sentiment prediction problems. So after we use BART for a clause, it returns a three-dimensional tensor, where the first dimension is unimportant, representing the batch size which is always one, while the second two dimensions represent the tensor as a matrix.

3.1.2 Algorithms for Model Training

For the sentiment prediction task, we choose a bidirectional LSTM (BiLSTM) combined with a convolutional layer (Conv1D) as the core of the model pipeline. These layers are chosen based on their performance in handling sentiment analysis tasks and their ability to learn complex patterns in textual data. Following these layers, a max pooling layer (GlobalMaxPooling1D) is used to reduce the output, capturing the most significant features. Between the BiLSTM and Conv1D layers, dropout layers are introduced to prevent overfitting by randomly dropping units during training. Finally, fully connected (Dense) layers are used for the output, applying softmax activation to generate the final predictions. The pooling layer used is max pooling, chosen over average pooling based on its slight performance advantage that it has in our architecture based on multiple tests.

3.2 Architecture

This section provides a view of the overall architecture for the aspect and sentiment prediction model. The architecture is designed to handle the complexity of extracting and analyzing sentiments from online reviews. We can break down the architecture into three main modules: Data Processing, Model Architecture, and Model Deployment. In Figure **1** is displayed the pipeline from start to finish for the ACE and ASE tasks.

The pipeline of our solution starts with the Data Processing module, responsible for transforming raw review data into a structured format suitable for training our prediction model. This module includes several key steps: data extraction and parsing, text preprocessing, sentence and clause splitting and vectorization.

The Model architecture module integrates various neural network components to perform category and sentiment prediction. The architecture highlights both advanced transformer models and common neural network layers to achieve robust performance.

The final module in the Architecture section, Web Application, focuses on integrating the trained model into a web application to provide almost real-time sentiment analysis capabilities.

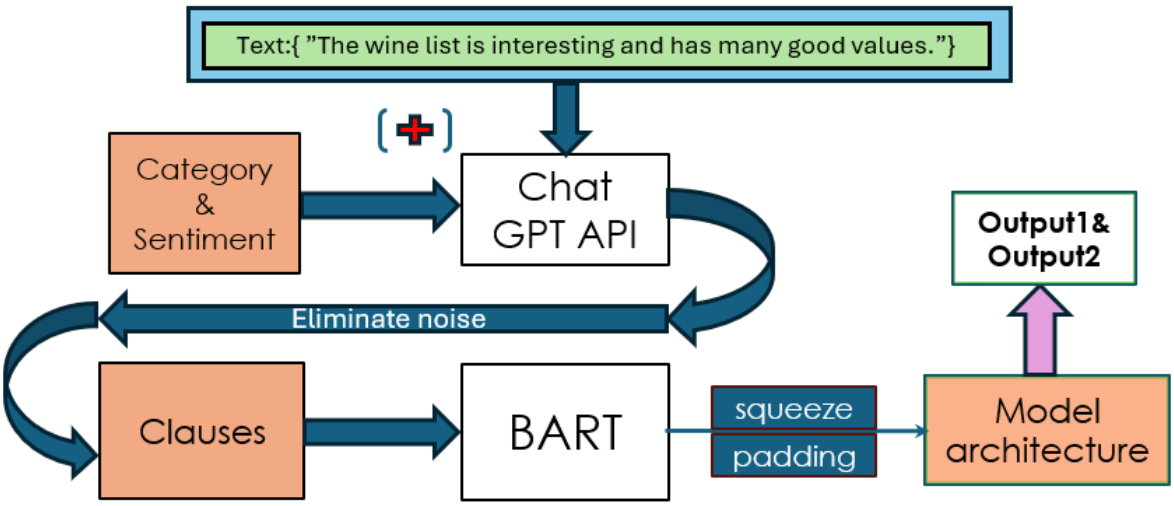

Figure 1: **Model pipeline architecture**

3.2.1 Data processing

To prepare the data for training the model, first I had to collect it from a relevant dataset. For this, I have used the SemEval 2016 Task Restaurants dataset which can be found at this **link**. After we download the dataset, we get three XML file formats. The data is stored in the XML files in a way that is easy to take for each sentence found in a review the opinions. For one sentence, multiple opinions can be found, which makes this task so complex, and each opinion is defined by a target, an aspect, and a polarity that must emerge from the sentence. It is enough to extract only the aspect and the polarity, but the target has its role too, because it indicates at what words (one or two usually) in a sentence (mostly nouns), the pair of aspect-polarity (opinion pair) refer to. For example, if we have the sentence "The duck confit is always amazing and the foie gras terrine with figs was out of this world.", we have first the target "foie gras terrine with figs" with the category "FOOD#QUALITY" and the polarity "positive" and then the target "duck confit" with the category "FOOD#QUALITY" again in this case and the polarity, same as before, "positive", for the same sentence. There is also another case where the target element can be "NULL", in this case, the opinion pair comes from the context of the text. This can be seen in the following example where the target is associated with "NULL", while the category is "SERVICE#GENERAL" and the polarity is "negative": "After all that, they complained to me about the small tip.". This target element isn't always reliable for the ABSA task as the **SemEval 2016 Task 5 Laptops** dataset and **MAMS** dataset do not have this target element.

Now, that we have the sentences and opinions for each one of them, we need to split the sentences into clauses for each opinion pair. So for sentences that contain multiple opinions, each category and sentiment must point to different groups of words so that two or more

opinions won't interfere as also presented in the **3.1.1** section. These groups of words aren't just random words, but they have meaning too, meaning that can be associated with the correct aspect and polarity, so now we can call one of these groups a clause.

It is possible that one or more words from a clause to be found in another clause. The best to put this into perspective is to use some examples. "The sushi seemed pretty fresh and was adequately proportioned.", here we have two opinions, both having the same target "sushi", in this case having the same category and polarity (food and positive) which isn't always the case. The two clauses that we can spot would be "The sushi seemed pretty fresh" and "The sushi was adequately proportioned.". In the next example we highlight that in the XML files of the dataset, not all opinions appear in the order that they appear in a sentence. For this text, "The decor is night tho...but they REALLY need to clean that vent in the ceiling...its quite un-appetizing, and kills your effort to make this place look sleek and modern." the first opinion that appears contains the target element "place", the second one, "decor", and the third one which is also the last is "vent"; here can be easily seen that the order of the words from the target elements in the sentence is different, "decor" word is first, "vent" is second and "place" is last in the sentence.

In conclusion, based on all we discussed about splitting sentences into clauses in this section and in **3.1.1** section, splitting sentences into clauses is a very complex and difficult task to achieve, which makes it really hard to find a suitable tool or solution. The only good and reliable solution that we come up with is using the ChatGPT API from OpenAI. We also tried to do it with the spaCy library, which is a free open-source library for NLP in Python, but the results wouldn't come close to the solution that Openai can offer. Even with this solution, it still isn't perfect but faster than splitting them manually.

This to work, we give the API the sentence alongside the list of opinions and the desired action written in a text about what we expect it to return, and how the output will look. The user input is then structured in a JSON file and is called the prompt. We write in the prompt that we expect an output like this: "Output format:[(1, "clause", "category", "polarity")]". So we expect that the output is inside some round brackets, first, the number of the clause, then the clause, followed by the category, and then followed by the polarity, all split by commas. The reason behind this output format is that we want to extract the data as easily as possible because as we all experienced ourselves with the AI chatbot we know that it does not always give the wanted output format, even if specified, and using it through its API for thousands of questions, that "not always" becomes we will certainly get some unexpected behavior which is manifested here in unwanted data besides the expected one. For example, we get some text outside the brackets or new lines somewhere in the output string, or sometimes, the squared brackets are missing or the number inside the round ones. To outcome this we parse the whole string and extract only the data between round brackets in a list and then evaluate it as a real expression, in this case a tuple and we also get rid of the numbers for clause counting. We only use the ChatGPT API only on sentences with more than one opinion, as for ones with only one opinion, is simple, the clause is the entire sentence.

Now that we have the clauses the next step is vectorizing the clauses with BART from Hugging Face to use them as input for the model architecture. As presented in **3.1.1** section, for a given clause, BART gives as output the vectorized clause as a tensor of 3 dimensions. The first dimension represents the batch dimension and is always one, and the other two dimensions represent a matrix of tensors. Between multiple vectorized clauses, the second dimension can vary while the last one is always the same, for example for a vectorized clause we can have the shape "torch.Size([1, 5, 1024])" while for others I can have the shape "torch.Size([1, 18, 1024])". If we use BERT, the last dimension is 768. After we use BART for both the test and training data, because of the variable second dimension found in the shape we need to make all tensors have the same shape. To do this the easiest and most common solution is to pad the sequences of all the vectorized clauses with zeros to the biggest first dimension of the matrix of tensors. By doing this all outputs from BART will have the same shape. The tensors for each clause are stored in a list for both test and training data. To do this, first we combine the lists for both data, to all be padded to the biggest value, but before the padding, we also squeeze the batch dimension. After that, we split back the list of processed BART data into the test and training data (X_train and X_test).

The categories and polarities are used as the two model outputs. For each processed clause we have a category and a polarity stored in different lists, The categories are stored in y1_train and y1_test and the polarities are stored in y2_train and y2_test. The polarity can be either "positive", "negative" or "neutral", while the category can be one of the main classes combined with one of these values "GENERAL", "PRICES", "QUALITY", "STYLE_OPTIONS", "MISCELLANEOUS", in this way creating some more classes and adding some more specific meaning to a class. The main classes are "AMBIENCE", "DRINKS", "FOOD", "LOCATION", "RESTAURANT", "SERVICE".

3.2.2 Model architecture

In this subsection, we provide a detailed explanation of each layer in the proposed model architecture used for aspect and sentiment prediction. The model architecture combines an Input layer, a Bidirectional LSTM (BiLSTM) layer, a Convolutional layer (Conv1D), two dropout layers, a pooling layer, and two fully connected layers (Dense). A better look at the proposed architecture can be seen in Figure **2** First, we have the Input layer. The input layer accepts sequences of feature vectors derived from the vectorized clauses. Each input has a shape of `(sequence_length, feature_dimension)`.

A Bidirectional LSTM (BiLSTM) layer is the next layer after the Input one, which is used to capture dependencies in the sequence data in both forward and backward directions. This enhances the model's ability to understand context from both past and future tokens in the sequence. The sequence data is represented by the processed vectorized clauses as shown in **3.1.1** subsection about data processing.

After the BiLSTM we use a first Dropout layer to reduce overfit. Using the Dropout layer is

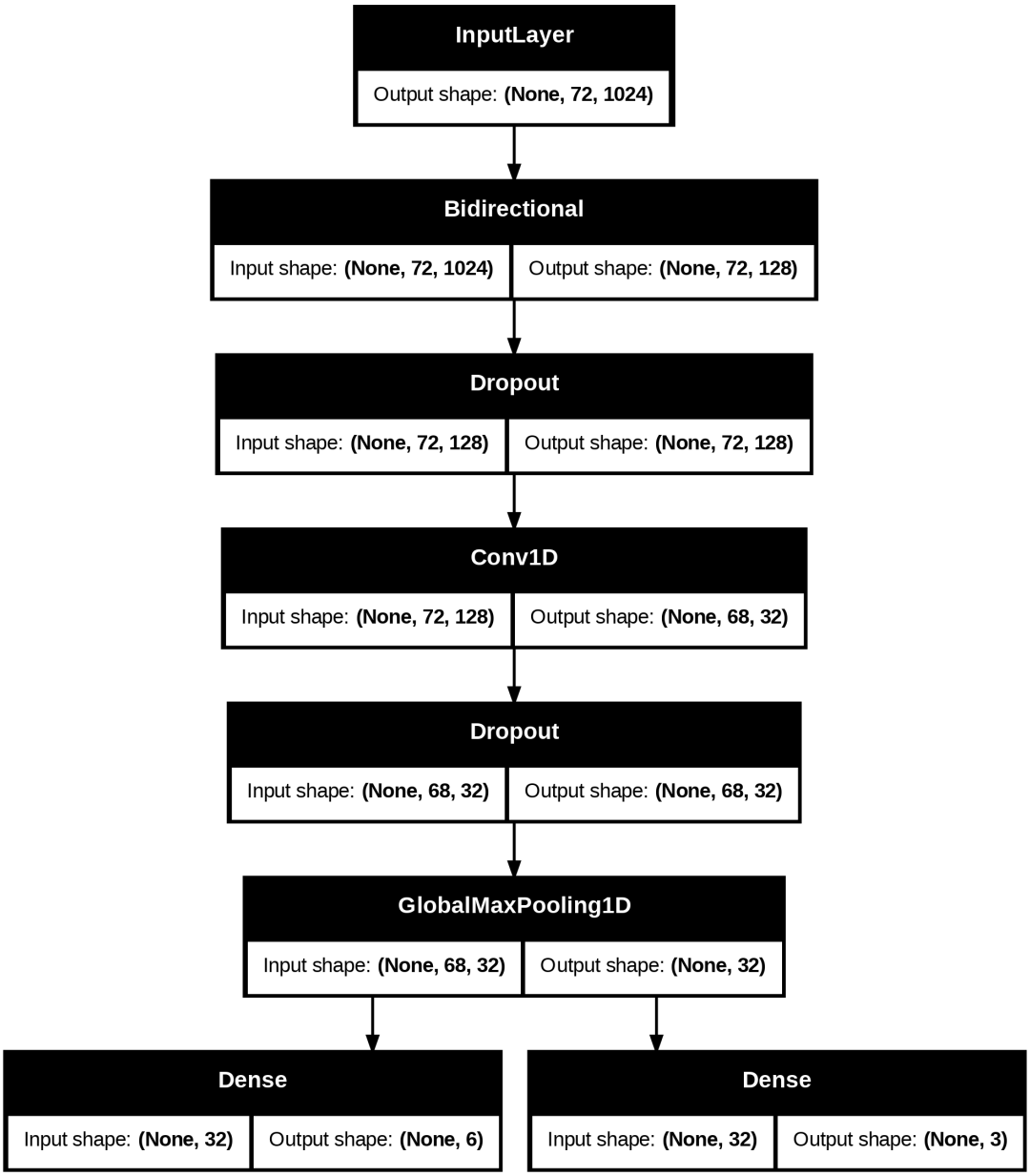

Figure 2: **Model architecture**

a well-known regularization technique that helps prevent overfitting by randomly deactivating neurons. This forces the model to learn redundant representations.

The first Dropout layer is then followed by a one-dimensional convolutional layer (Conv1D) for extracting the most important features by applying convolutional filters, allowing the model to detect local patterns and features in the text data.

A second and final dropout layer is added after the convolutional layer to further regularize the model and prevent overfitting.

After that, we have a global max pooling layer (GlobalMaxPooling1D), for reducing the output dimensions while keeping the most significant features detected by the Conv1D layer.

The final layers are two Dense layers that are used to produce the final predictions for the two outputs. Each dense layer corresponds to one output, the first is for the category output, while the second one is for the polarity output.

The model is compiled with the Adam optimizer, categorical cross-entropy loss for each output, and metrics including accuracy, precision, and recall. Early stopping is used to prevent overfitting by monitoring the validation loss, which means that the model stops training after several bad validation loss values.

The model is trained using the `fit` method, with the training data (`X_train`) and the two sets of labels (`y1_train` and `y2_train`) to satisfy both ACE and ASE tasks. The test data is used for validation.

3.2.3 Web application

We also created a web application for the project, to facilitate an easier interaction with my proposed architecture for category and sentiment predictions, by offering a simple and easy-to-use interface. The application is developed with Flask for the server and HTML for the front end.

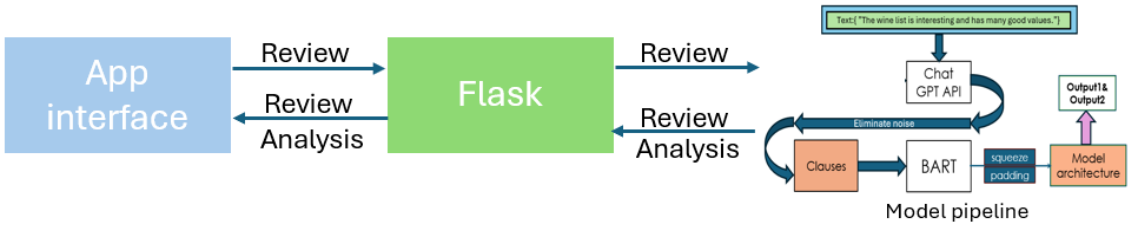

Figure 3: **Web application architecture**

The flow of the overall web application interaction is as follows. First, there is a textbox where you can insert text. Then, on submit, the text is processed and then sent to the trained implemented model. After all is processed, it will appear right below the textbox the results: for each clause found in the text it will be displayed the clause together with its predicted category and polarity.

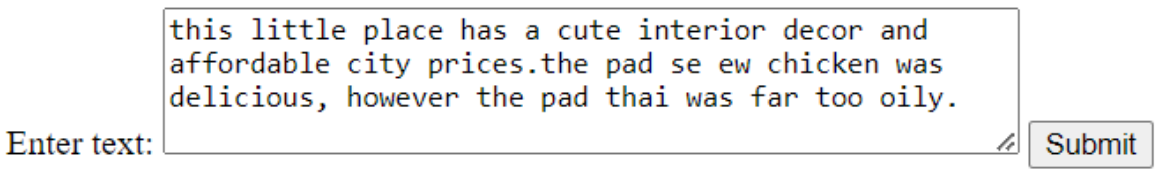

Figure 4: **Input review**

The clauses with their predicted opinions pair will be shown in the application interface one below the other. The text that we introduce can contain one or multiple sentences that are recommended to represent a review for a restaurant, as the model is trained on a dataset that contains online reviews for restaurants with clear signs of overfitting as can be seen in chapter **5**. Figure **4** shows the textbox where we can write our review and the submit button. In Figure **6** we can see our app as a whole, having the results displayed for our input review.

4 IMPLEMENTATION

In this chapter we discuss the tools and technologies used in the process of realizing this project from start to end; to be more specific, from way back to the data processing part to the model interaction with the web application. This chapter covers the entire workflow, presenting how each component and step contributes to the final solution. By detailing the implementation process, we provide in-depth looks into the practical challenges encountered and the solutions adopted to achieve a functional solution.

4.1 Modules implementation

The implementation of each module presented in the subsections of the **3.2** section of the methodology chapter is presented in the following subsections.

The modules covered include data processing, model architecture, and the development of the web application. The data processing module involves extracting and preparing the dataset for analysis, which includes parsing, cleaning, and vectorizing the data. This step is fundamental as it transforms text data into a structured format that can be effectively utilized by our machine learning model.

The model architecture module focuses on designing and implementing a simple model that integrates the BART transformer model with BiLSTM and Conv1D layers in the pipeline process. This combination aims to highlight the strengths of advanced transformer models and simpler neural network components to achieve robust performance in ABSA task. The architecture is designed to create a balance between complexity and efficiency, ensuring that our implemented model can handle detailed sentiment analysis while maintaining low training times and computational requirements to facilitate a less capable hardware like our testing setup.

The web application subsection presents an interactive interface that allows users to input text and receive sentiment analysis results. This module is implemented using the Flask framework, which facilitates the development of a responsive and user-friendly application. The integration of the model with the web application ensures that users can easily interact with the system and obtain almost real-time results for the sentiment analysis of a review.

4.1.1 Data processing

For the data processing implementation, we start with how we get the data from the dataset. Because the dataset is stored in XML files we parse it for extracting for each sentence the categories and sentiments. This is achieved by using the "xml.etree.ElementTree" library, which is a Python library that is a very powerful tool in this kind of parsing or creating XML data tasks.

To set up the ChatGPT API for splitting sentences into clauses I followed the instructions found on the official OpenAI website, which can be found at this **address**. The downside of using the OpenAI's API is that it costs real money and the pricing is based on usage. For example for the restaurants dataset, to get the clauses for all the sentences it costed 3.62 dollars from an initial credit fill of 5 dollars. More information about the SemEval 2016 Task 5 restaurant dataset and how big it is can be found in 5.1 section, to make an idea of the usage that implies on the API. It is also important to mention that the sum given was reached on multiple tests and runs of the API for the dataset. The chosen model from OpenAI was "GPT-4o" to get the best possible results. GPT-4o is the most capable and advanced AI chat bot from OpenAI.

To know how to use BART for vectorizing the clauses we used the code from the examples that can be found on the Hugging Face website at this **link**. Then the outputs from the last hidden state of the transformer model are squeezed and padded. The code for the processing part of the transformer outputs is shown in Figure **5**.

```
def proccess_bart_data(clause_vectors_res):
    tf.squeeze(clause_vectors_res[0], axis=0)
    clause_vectors_res_2d = [tensor.squeeze(0) for tensor in clause_vectors_res]
    clause_vectors_padded = pad_sequence(clause_vectors_res_2d, batch_first=True)
    return clause_vectors_padded
```

Figure 5: **Squeeze and pad transformer outputs**

The processed data, now vectorized and properly formatted, is fed into the BiLSTM-CNN model. This model leverages both the temporal dependencies captured by BiLSTM and the spatial hierarchies identified by CNN, making it highly effective for our classification tasks. The combination of these techniques ensures that the model can handle complex language structures and deliver precise predictions.

In addition to the initial data parsing and clause splitting, the processing pipeline involves several stages to ensure the data is in the correct format for machine learning applications. After splitting the reviews into sentences, each sentence is further divided into clauses. This detailed segmentation allows for a more granular analysis of the text, improving the accuracy of sentiment and category predictions.

This detailed and multi-stage data processing approach is essential for preparing the dataset

for advanced machine learning tasks. By leveraging tools such as "xml.etree.ElementTree," the OpenAI GPT-4o API, and the BART model from Hugging Face, we ensure that the data is processed efficiently and accurately, ready for training robust models for sentiment analysis and category classification. Future improvements could involve optimizing the API usage to reduce costs and exploring more advanced NLP solutions to replace the usage of the ChatGPT API completely.

For the splitting of sentences into clauses, we also took a more common approach, by using the SpaCy library, which is a Python library for NLP. The solution that we found can be explored at this address. Next I present how this solution fails to perform for this task. We best highlight this with an example. For the first sentence provided in the textbox of our Web Application in Figure **6**, the resulted clauses are displayed in Figure **4** of my application. So for the sentence "this little place has a cute interior decor and affordable city prices." we get the following clauses: "this little place has a cute interior decor" and "this place has affordable city prices". This split correctly highlights that the "place" noun can be associated with two sentiments, one for the fact that the place is cute and another for the fact that the place has affordable prices. Now if we try to split our sentence with the solution found at this link, we get as result "place has a cute interior decor and affordable city prices", but for a more simple sentence like "He eats cheese, but he won't eat ice cream" it splits it in "He eats cheese," and "he won't eat ice". We also tried to implement a solution using the SpaCy library but no result was good, so in order to best evaluate our BiLSTM-CNN-based model we use the ChatGPT API. A better choice than using the ChatGPT API is to create a better dataset that has for a sentence also the clauses with each opinion pair. Of course, this would translate into more excessive manual work.

4.1.2 Model architecture

We implement the proposed model architecture using TensorFlow, which is a free and open-source software library for machine learning developed by Google. TensorFlow supports both GPU and CPU, making it a versatile tool for deep learning tasks. Each layer in our architecture is implemented using TensorFlow components, ensuring consistency and compatibility throughout the model.

The core of our architecture is a Bidirectional Long Short-Term Memory (BiLSTM) and a Conv1D layer. BiLSTM is particularly effective for sequence learning tasks as it processes the data in both forward and backward directions, capturing dependencies from both ends of the sequence. In our implementation, the BiLSTM uses an LSTM layer with 64 units. This LSTM layer takes as input the sequence data shaped by the outputs from the last hidden state of the BART model after padding.

The 1-dimensional convolutional layer (Conv1D) is used to capture local patterns within the sequence data. The Conv1D layer is defined with a filter size of 32 and a kernel size of 5. The filter size determines the number of output channels, while the kernel size specifies the

window over which the convolution is applied. This configuration helps to detect features that are important for the classification tasks.

The outputs from the Conv1D layer are then processed through a global max-pooling layer, which reduces the dimensionality of the data by taking the maximum value from each feature map. This step ensures that the most prominent features are retained while reducing the computational load for subsequent layers.

Following the convolutional and pooling layers, the data is reduced and passed through fully connected (dense) layers. These dense layers are responsible for combining the features extracted by the previous layers and making the final predictions. The final dense layer uses a softmax activation function to output probabilities for each class in the category and polarity classification tasks.

To prevent overfitting, we include dropout layers in our architecture. Dropout is a regularization technique where randomly selected neurons are ignored during training, which helps to generalize the model better. In our model, we use a dropout rate of 0.5, meaning that half of the neurons are dropped during each training iteration, significantly reducing the chances of overfitting. We use 2 Dropout layers, one before and one after to better as we train the model in a low resources environment that represents the SemEval2016 Task Restaurants dataset.

By combining the strengths of BiLSTM for sequence learning and Conv1D for feature extraction, this architecture is well-suited for complex text classification tasks as seen in the state-of-the-art solutions presented in chapter 2.

4.1.3 Web application

In this project, I made the backend of my application using Flask. It is a lightweight web framework for Python, and it is made in such a way as to ease the development of web applications. The backend renders the template of HTML. Below, in Figure **6**, we see our web application interface.

The Flask application is based on HTTP requests, using a REST API architecture. I have only one root route that supports both GET and POST requests, and both render the application interface with the HTML template. The difference between the two requests is that for the POST request on submit for the text in the textbox if it is not NULL, the text as a restaurant review will be processed and the results will be displayed.

The text processing done by the server is similar to the one shown in Figure **1** of the model pipeline, or explained in the **3.2.1** subsection, with only two differences. The first difference is that because in the textbox can be inserted a whole review that can be expected to be represented by multiple sentences, it is mandatory that the review is split into sentences first before anything else is done. This is done with a function we implemented. The other

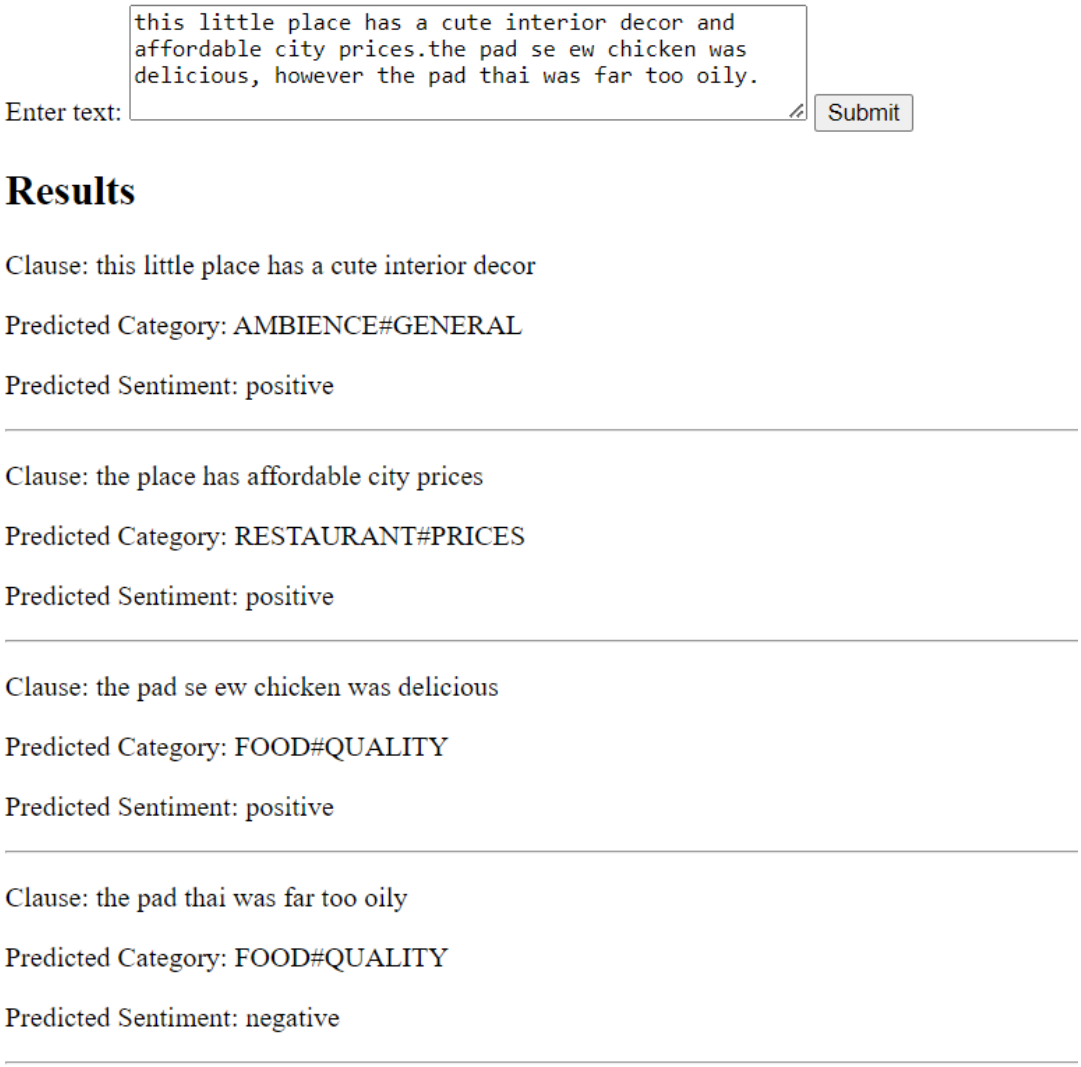

Figure 6: **Web application interface**

difference is because we can only give a review to the server, the sentence splitting into clauses can not be assisted by the expected category and polarity, as they have to be predicted, not given.

The machine learning model, which was trained on a dataset specifically for this task, then analyzes the processed text. So after the sentence splitting into clauses, part is done with the ChatGPT API, the clauses are extracted separately and then sent to the BART transformer model for vectorization. The model's predictions, which include the category and polarity of each sentence, are generated based on the sequence of features extracted from the last hidden state of the transformer-based architecture with further preprocessing afterward. These predictions are then sent back to the frontend, where they are displayed to the user, providing a clear and concise analysis of their review. This whole process from pressing submit to the review to getting the results displayed will last for a couple of seconds at best.

To elaborate further, the Flask framework was chosen for its simplicity and flexibility, making it ideal for creating a prototype for this application. The use of a single root route simplifies

the architecture, allowing for a streamlined process for both GET and POST requests. This design decision ensures that the application remains lightweight and responsive. The Flask server will be started at the localhost on port 7020.

5 EXPERIMENTAL RESULTS

In this chapter, we present a detailed description of the experimental results obtained through the category and polarity prediction model. This involves detailed descriptions of the dataset used, the experimental setup, and the results from various tests used to estimate the model's performance, so we can validate the effectiveness of our methodologies.

We begin by detailing the datasets used in our experiments in section **5.1**. This section provides a thorough overview of the SemEval2016 Task 5 Restaurants dataset, which serves as the primary dataset for training and evaluation. The discussion includes the structure of the dataset, the nature of the reviews, sentences, and opinion pairs, and the challenges posed by the dataset's characteristics. Understanding the dataset is a crucial task in how to prepare the data and what performances to expect based on its characteristics.

Following the dataset details, section **5.2** describes the experimental setup. This section outlines the hardware and software environments, specifying the computational resources utilized, including the specifications of the systems on which the experiments were conducted. Here, knowing the capabilities of your hardware is crucial because a less capable hardware can make you take different approaches in choosing the right architecture for your model in order to reach your goals.

Finally, section **5.3** presents the results of our experiments. Here we provide detailed performance metrics of our model. We report on various evaluation metrics such as accuracy, precision, recall, and F1 score for both category classification and polarity prediction. The results are compared across different configurations, including a comparison between BART and BERT pipelines. We also discuss the impact of reducing the complexity of category labels.

5.1 Datasets details

We use the SemEval2016 Task 5 Restaurants (SE2016TASK5R) dataset to evaluate our solution. The dataset is organized into three files: one for training data, one for test data, and one for trial data. This organization means there is no need to manually split the data into training and testing sets, as each set is already provided separately. The dataset can be downloaded from this **website**.

The training data contains 350 reviews with a total of 1708 sentences, where each sentence

can have from a minimum of zero opinion pairs to a maximum of 8. As a reminder, an opinion pair is a pair of category and sentiment that can be found in a sentence. The number of each polarity for each category is shown in Figure **7**. It results in a number of 2508 opinion pairs and clauses.

Category	Positive	Neutral	Negative
RESTAURANT#GENERAL	313	8	101
SERVICE#GENERAL	211	12	226
FOOD#QUALITY	617	28	204
FOOD#STYLE_OPTIONS	83	9	45
DRINKS#STYLE_OPTIONS	29	0	3
DRINKS#PRICES	13	0	7
RESTAURANT#PRICES	34	6	40
RESTAURANT#MISCELLANEOUS	58	13	27
AMBIENCE#GENERAL	197	16	42
FOOD#PRICES	41	1	48
LOCATION#GENERAL	21	6	1
DRINKS#QUALITY	40	2	5

Figure 7: **Distribution of polarities for each category for the train data**

Data	Reviews	Sentences	Opinions
Train	350	1708	2508
Test	90	676	861
Trial	10	41	65

Figure 8: **Dataset data**

The test data contains 90 reviews with a total of 676 sentences, where each sentence starts from a minimum of zero opinion pairs to a maximum of 26, which is an exception to have such a high number of opinion pairs, the second maximum number being 6. Figure **9** shows the distribution for the test data. Here we have a total of 861 clauses and opinion pairs. Lastly, the trail data contains 10 reviews which can be translated into 41 sentences and 66 clauses and opinion pairs. The maximum number of opinions in a sentence is 4.

It is important to mention that the sentences with no opinions were left out, as they couldn't be used for training the model.

In this paragraph, we present the challenges that we encounter with the online reviews restaurants dataset. The SE2016TASK5R dataset presents certain challenges due to its relatively small size and limited resources. The dataset's small number of samples can lead

Category	Positive	Neutral	Negative
FOOD#QUALITY	269	13	31
FOOD#STYLE_OPTIONS	31	9	15
RESTAURANT#GENERAL	107	1	34
SERVICE#GENERAL	72	7	76
AMBIENCE#GENERAL	61	3	2
DRINKS#STYLE_OPTIONS	11	0	1
FOOD#PRICES	6	3	14
RESTAURANT#PRICES	6	2	13
LOCATION#GENERAL	11	2	0
DRINKS#QUALITY	21	0	1
RESTAURANT#MISCELLANEOUS	16	4	13
DRINKS#PRICES	0	0	4

Figure 9: **Distribution of polarities for each category for the test data**

to issues such as overfitting, where the model performs well on the training data but fails to generalize to new, unseen data. Additionally, the limited diversity of the reviews in the dataset may not cover the full range of possible reviews, this potentially impacting the model's ability to handle more varied data.

Despite these challenges, the dataset provides a structured and well-annotated source of data for developing and evaluating aspect-based sentiment analysis models. While this dataset is a crucial resource for training and testing our model, it is essential to acknowledge its limitations. More information about the dataset, including its structure and annotations, can be found in the referenced article Maria et al. [8] in the bibliography part of this paper, which can be found at the bottom.

We also choose for our research to evaluate the model on a more general case. So for the category prediction, the classes "RESTAURANT#GENERAL" and "FOOD#PRICES" are reduced to "RESTAURANT" and "FOOD", getting rid of the entity after "#" char. Doing so the classes are reduced to a more general representation of them. The new resulting distributions of polarities for each category for the reduced complexity ACE task (ACERC) are shown in Figures **10** and **10**. **11**.

5.2 Experimental setup

I trained the model on a laptop, which has the following specifications: an Intel i5-9300H CPU with 4 cores, a GTX 1050 mobile with 3GB of VRAM, and 8GB RAM. For training, I

Category	Positive	Neutral	Negative
RESTAURANT	405	27	168
SERVICE	211	12	226
FOOD	741	38	297
DRINKS	82	2	15
AMBIENCE	197	16	42
LOCATION	21	6	1

Figure 10: **Distribution of polarities for each category for the train data (ACERC)**

Category	Positive	Neutral	Negative
FOOD	306	25	60
RESTAURANT	129	7	60
SERVICE	72	7	76
AMBIENCE	61	3	2
DRINKS	32	0	6
LOCATION	11	2	0

Figure 11: **Distribution of polarities for each category for the test data (ACERC)**

used only the CPU. The CPU has a base clock of 2.4GHz and a turbo clock of 4.1GHz and the RAM is a DDR4 with a frequency of 2400MHz.

For model evaluation, I also vectorized the clauses with BERT in order to see how it compares with the transformer model BART for this task. In rest, the pipeline remains the same.

Because we train the model with an early stopping of 2 for the validation loss, the model for each transformer lasts for 11 epochs. We use Adam optimizer with a default learning rate of 0.001 and a batch size of 32.

With this hardware, the vectorization of the clauses with the BART-based transformer architecture for the training data lasted 10 minutes and 56 seconds while for the same inputs, the vectorization with BERT Encoder-only architecture lasted for 2 minutes and 65 seconds as displayed in Figure **12**. This is to be expected because of the more complex architecture of BART with both an Encoder and Decoder. In rest both transformer models are bidirectional. For the training times, we have a similar number of seconds as we use the same BiLSTM-CNN base model architecture. For example the training for the pipeline with Bart with the training data and test data as validation data, the training lasts for 2 minutes and 46 seconds. Even if the hardware and software setup are sufficient for our experiment, it presents several limitations, like the low numbers of cores and frequency of the CPU, and the limited size of the RAM and frequency too. Using a more complex architecture with more layers or increasing the

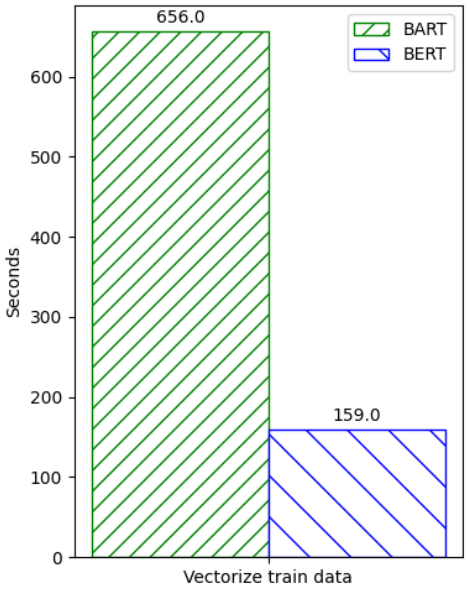

Figure 12: **Vectorization time**

number of units for the BiLSTM and Conv1D of our current implementation, would clearly be a more challenging run for our testing hardware. It will be interesting to see in future testing how the running times will stack for the GPU configuration as the Tensorflow framework has support both for CPU and GPU.

5.3 Results

Here we present the results for our both ABSA tasks based on multiple tests on the SE2016TASK5R dataset. To evaluate the model we compute the accuracy, precision, recall, and F1 scores from scikit-learn. We also create graphics to display the loss over the epochs and to display the accuracy for both training and validation data over the epochs for a more in-depth understanding of how our classification model performs.

This section is divided into 2 subsections because we wanted to see how the model performs if we reduce the complexity of the ACE task by reducing the category label to its more general classes. The first section exclusively presents the results of the main pipeline and how it compares with the one with BERT for both ACE and ASE tasks, while the second section does the same but with the reduced complexity approach for the ACE task (ACERC).

5.3.1 Aspect Category Extraction

Here we evaluate the results for our BiLSTM-CNN-based model with both BART and BERT for clause vectorization. For the main pipeline using BART, we get the following scores,

displayed in Figure **13** and **14** for category prediction and respective for polarity prediction.

Model	Transformer	Accuracy	Precision	Recall	F1-Score
BiLSTM-CNN	BART	0.77	0.76	0.77	0.75
BiLSTM-CNN	BERT	0.74	0.73	0.74	0.72

Figure 13: **accuracy, precision, recall, f1 score for the ACE task**

Model	Transformer	Accuracy	Precision	Recall	F1-Score
BiLSTM-CNN	BART	0.89	0.88	0.89	0.88
BiLSTM-CNN	BERT	0.83	0.82	0.83	0.82

Figure 14: **accuracy, precision, recall, f1 score for the ASE task**

The loss over epochs can be seen in Figure **15**. Here, the training loss is decreasing over the epochs. The validation loss decreases initially over the first 3 epochs, but then it starts to decrease slowly with some small spikes on the graph and ends in a small increase. This means that the model starts to overfit after some point even if initially it seems to improve. In Figure **16** the training accuracy and validation accuracy for the category prediction start from lower bounds. The training accuracy for the ACE task manages to come close to the one for the ASE task, while for the validation accuracy after a slight increase for the first 2 epochs, it then maintains a constant noticeable distance from it with fewer spikes. Both graph lines for validation accuracy show signs of overfit. The accuracy for the category prediction is 0.77 and the accuracy for the polarity prediction is 0.88, with a clear advantage for the ASE task.

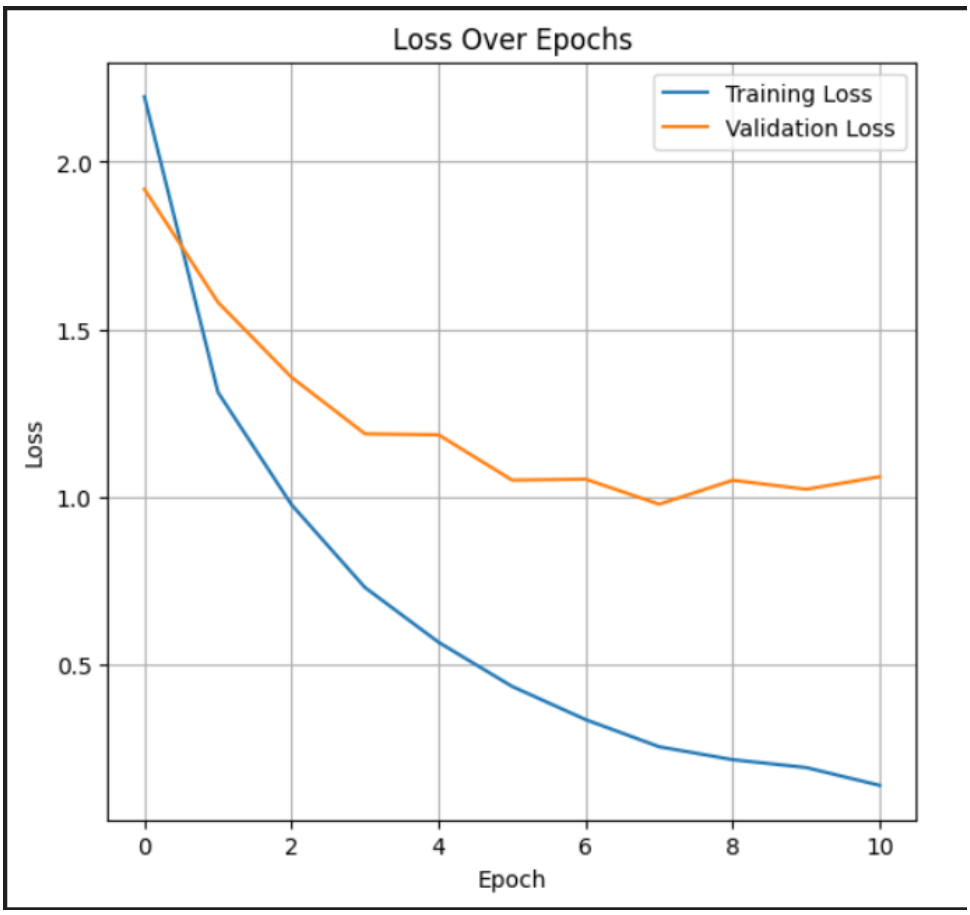

Figure 15: **loss over epochs for the main pipeline**

Next, we discuss about the pipeline architecture with BART and how it performs compared to BERT. The computed scores for the pipeline with BERT are presented in Figure **13** and **14** for both tasks. The scores aren't as good as the ones with BART.

For the graph with the loss over the epochs in Figure **17**, the validation loss decreases for the first 2 epochs, then it decreases with one spike and stabilizes. Lastly, it ends with a spike and a slight increase. Compared with our main pipeline architecture, the validation loss has a lower decrease overall. In Figure **18** the training accuracy follows the same path as in the one in Figure **16**. The same applies to the validation accuracy with a difference, the graphic lines for both outputs look like they are mirrored, when one has an increase the other one has a decrease. The accuracy score for the first output is 0.74 while the one for the second output is 0.83. The vectorization times for both pipeline architectures are presented in Figure **12**. Splitting the sentences with BART takes 4 times more seconds than with BERT, resulting in a more complex architecture for BART.

5.3.2 Aspect Category Extraction Reduced Complexity

Here, in this subsection, we train the model with only the main classes for the ACE task, for a more generalized classification using the main classes only. Instead of having the category

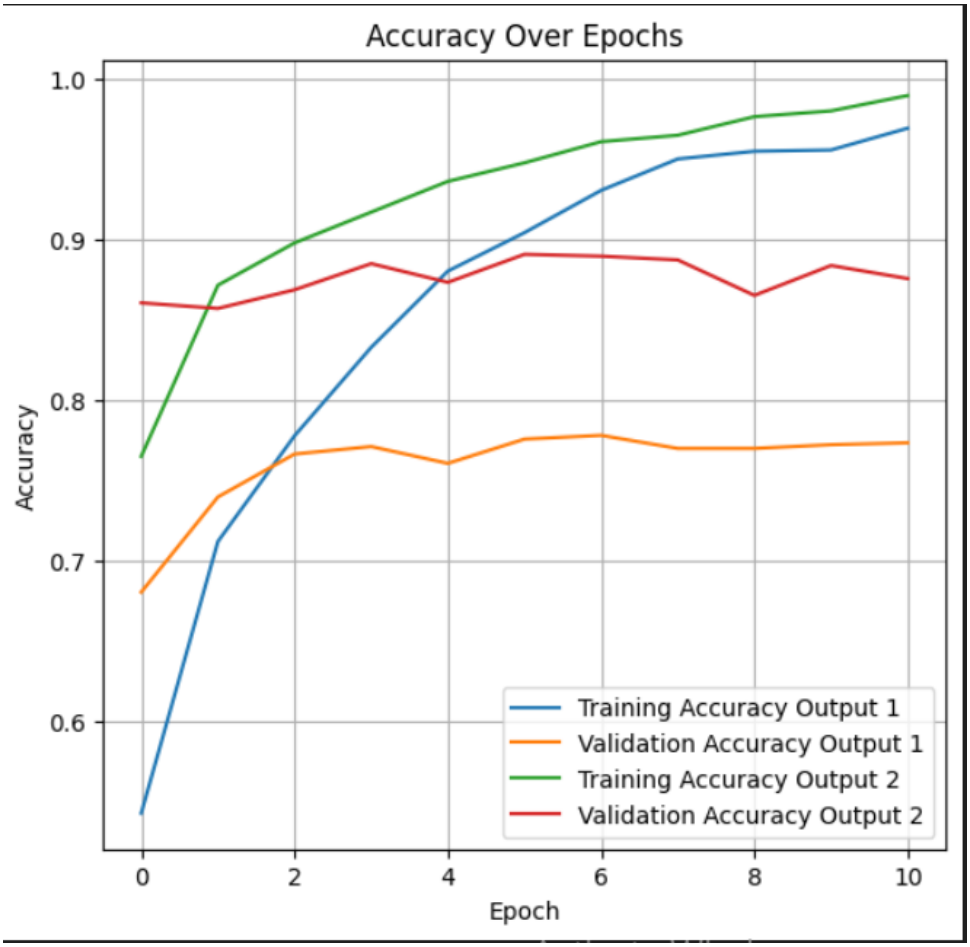

Figure 16: **accuracy for training and validation data over the epochs for the main pipeline**

"FOOD#QUALITY" we have just "FOOD", so instead of 12 classes, we have 6. The computed scores in Figures **19** and **20** show significant performance gains for the model, especially for the category classification. Here, for the pipeline with BART, the training will last for 11 epochs while for the pipeline with BERT will last for 13 epochs.

For the main pipeline using BART, I get the following scores, displayed in Figures **19** and **20**. The loss over epochs can be seen in Figure **21**. Here, the training loss is decreasing over the epochs. The validation loss decreases initially over the first 4 epochs, but then it starts to have some spikes on the graph and then stabilizes. This means that the model starts to overfit after some point even if initially it seems to improve.

In Figure **22** for the training accuracy the graphic line for both outputs seems to have the same form. The same can be said about the validation accuracy for both outputs, the difference being that for the training accuracy, the lines are closer while for the validation accuracy, the graphic lines have some more distance between them. The accuracy computed with scikit-learn for the category prediction is 0.86 and the accuracy for the polarity prediction is 0.89.

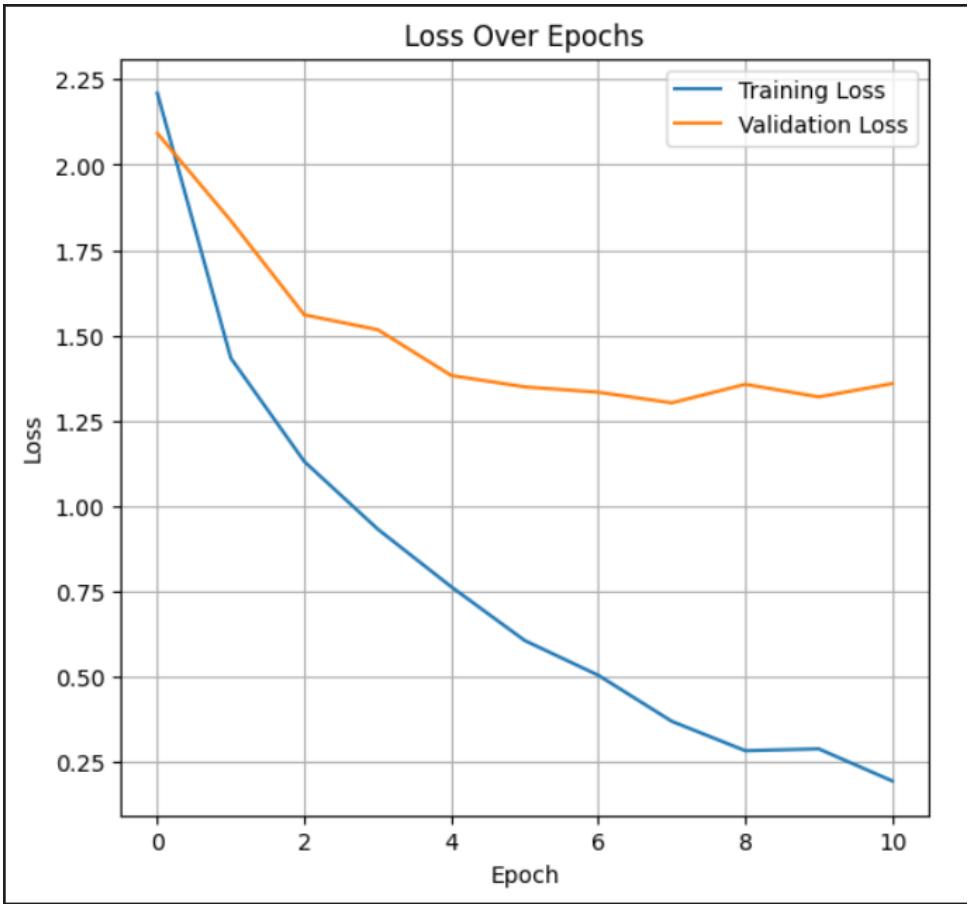

Figure 17: **loss over epochs for the BERT pipeline architecture**

Because I used BART for my pipeline architecture for the text processing step, it was interesting to see how BART would stack against the use of BERT instead for this step. The implemented model architecture would remain the same.

The computed scores for the pipeline with BERT are presented in Figure **19** and **20**. For the graph with the loss over the epochs in Figure **23**, the validation loss decreases for the first 3 epochs, then it has some spikes and in the end it increases again. Compared with my main pipeline architecture, it has more spikes and it doesn't stabilizes at the end.

In Figure **24**, the training accuracy for ATE task and for ATSA task are nearly the same as in Figure **16**, while the validation accuracies compared with the ones in Figure **24** are much closer, but they have different shapes with more pronounced spikes.

The machine learning model training times for both pipeline architectures are presented in Figure **25**. Splitting the sentences with BART takes 4 times more seconds than with BERT. The training times are comparable as the same model is being used.

Figure 18: **accuracy for training and validation data over the epochs for the BERT pipeline architecture**

Model	Transformer	Accuracy	Precision	Recall	F1-Score
BiLSTM-CNN	BART	0.86	0.86	0.86	0.85
BiLSTM-CNN	BERT	0.83	0.84	0.83	0.83

Figure 19: **accuracy, precision, recall, f1 score for the ACE task using the main classes for category (ACERC)**

Model	Transformer	Accuracy	Precision	Recall	F1-Score
BiLSTM-CNN	BART	0.90	0.89	0.90	0.89
BiLSTM-CNN	BERT	0.83	0.82	0.83	0.82

Figure 20: **accuracy, precision, recall, f1 score for the ASE task using the main classes for category (ACERC)**

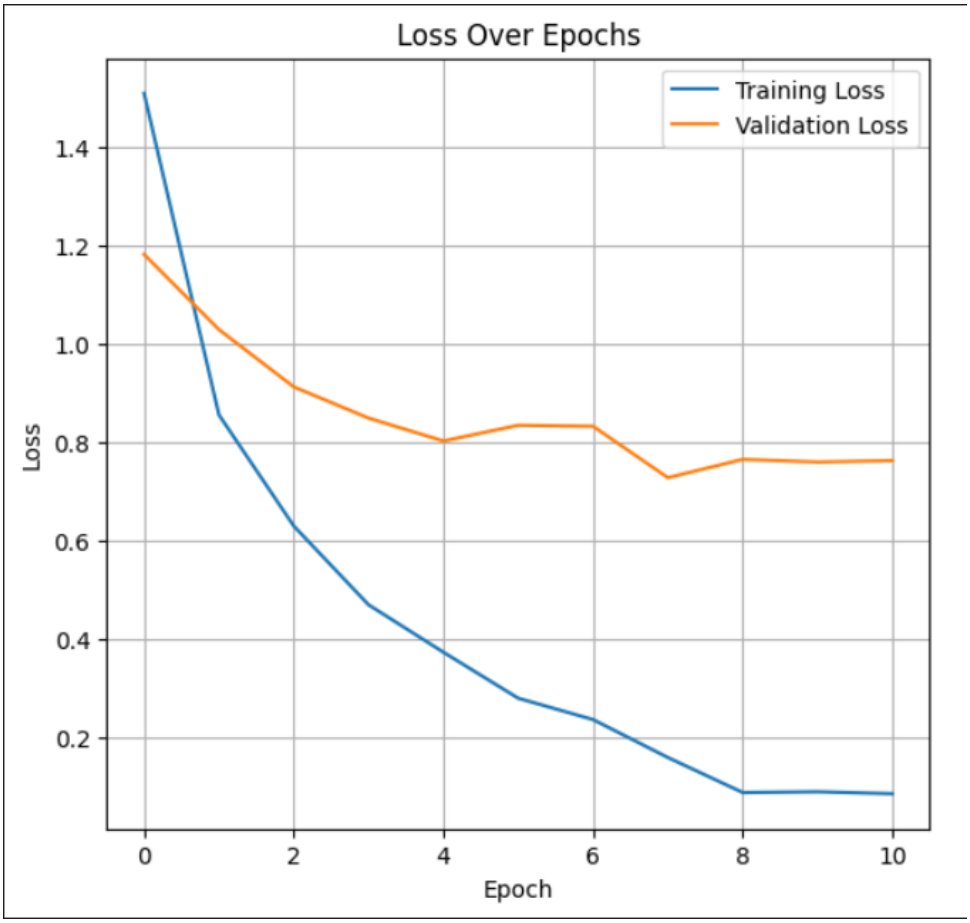

Figure 21: **loss over epochs for the main pipeline (ACERC)**

5.4 Discussions

In this section, we compare the performance of our model using two different transformer models: BART and BERT. The evaluation metrics used for this comparison are accuracy, precision, recall, and F1 score presented in tables in the **5.3** section. We will look at the scores for both category classification and polarity prediction.

For the ABSA task, we resumed only to the category prediction and polarity prediction. The training data had a relatively small number of samples to work with (2508), even after splitting the sentences into clauses for each opinion, in which case the model would be from start prone for overfit. Given these aspects, and the simple architecture of the model, the resulting scores presented in section **5.3** are promising. More we don't extract the aspect terms with the category and polarity which can result in a less accurate prediction for the model.

In the category classification task, the pipeline with BART transformer model shows superior performance compared to the pipeline with BERT across all evaluation metrics. The accuracy of BART is 0.77, which is slightly higher than BERT's accuracy of 0.74. Similarly, BART outperforms BERT in precision (0.76 vs. 0.73), recall (0.77 vs. 0.74), and F1 score (0.75 vs. 0.72). These results suggest that BART is more effective in correctly identifying and

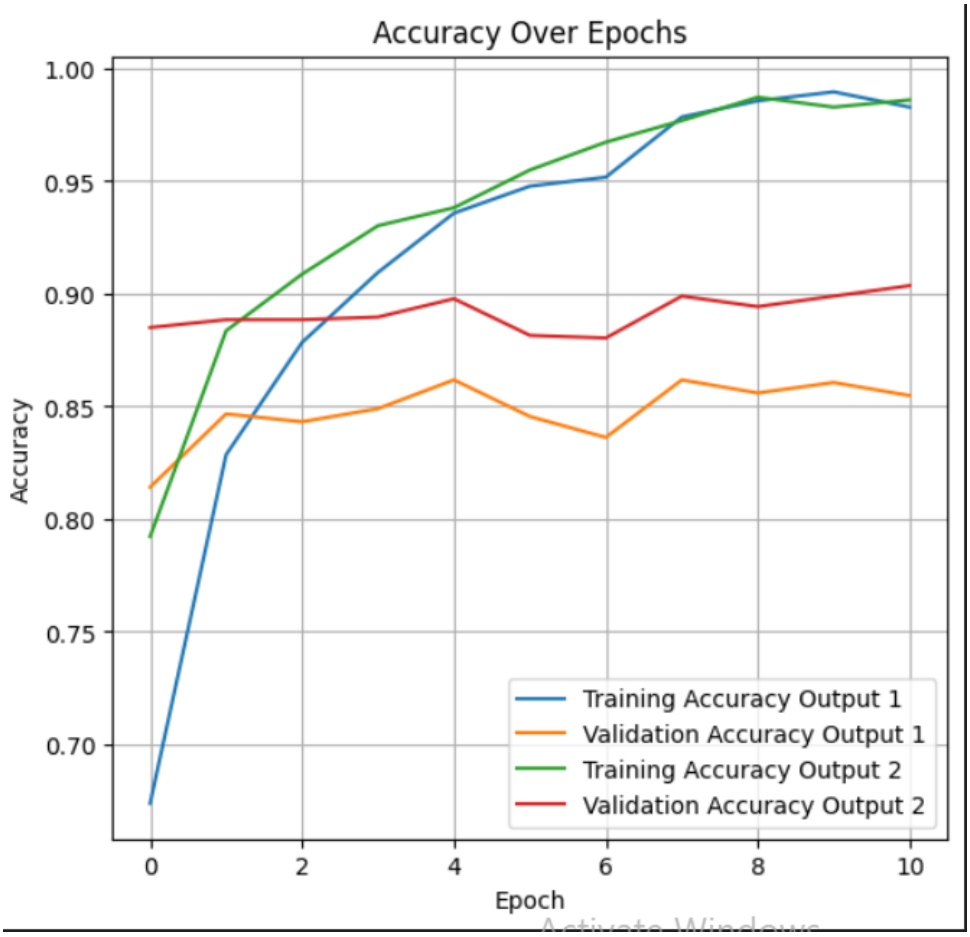

Figure 22: **accuracy for training and validation data over the epochs for the main pipeline (ACERC)**

classifying categories within the text, likely due to its ability to better capture and utilize contextual information.

For polarity prediction classification, the BART pipeline again outperforms the BERT pipeline. The use of BART with our BiLSTM-CNN model achieves an accuracy of 0.89 compared to BERT's 0.83. Precision, recall, and F1 scores are also higher for BART (0.88) compared to BERT (0.82). These results indicate that BART is better at distinguishing the sentiment of the text, whether positive, negative, or neutral.

A clear performance advantage can be seen for the ASE task over the ACE. This is mainly expected because of the difference in the number of classes for each label with 4 times more classes for the category output.

Next we analyze the performance of the BART and BERT pipelines when the complexity of the category label is reduced by halving the number of classes from 12 to 6. This comparison aims to understand the impact of reducing the number of classes on model performance and why having fewer classes can be advantageous.

When the number of classes for category classification is reduced from 12 to 6, both BART

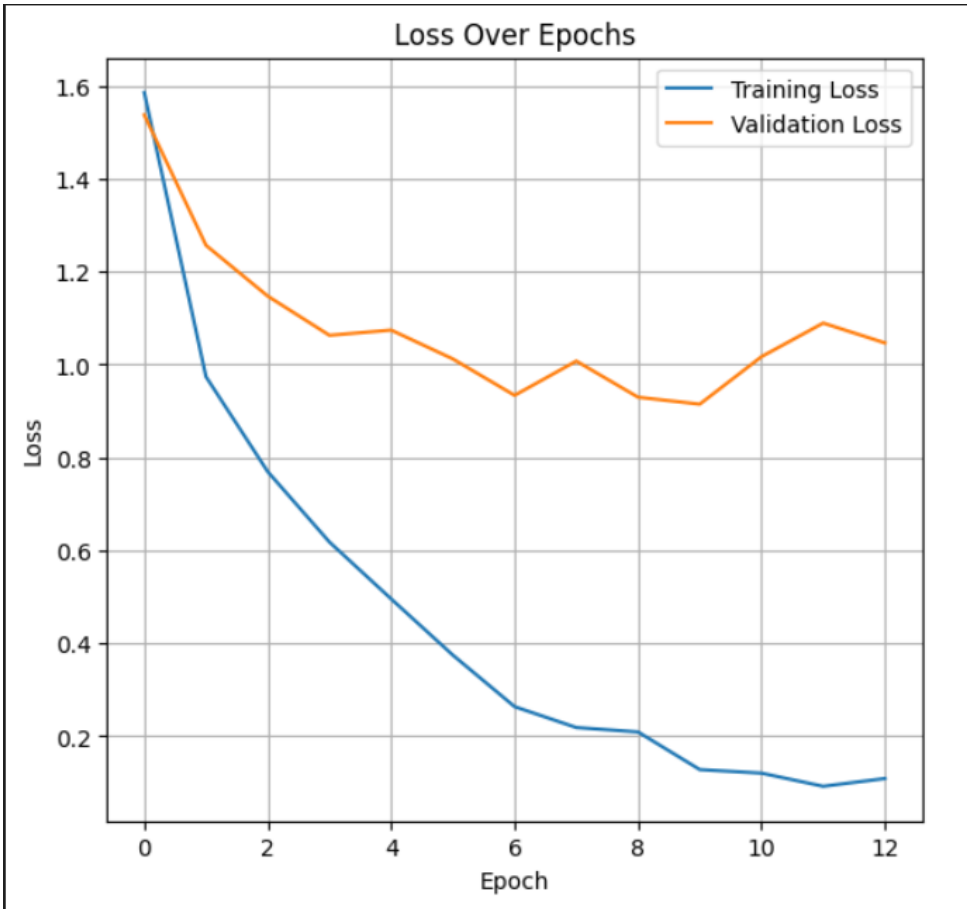

Figure 23: **loss over epochs for the BERT pipeline architecture (ACERC)**

and BERT show improved performance compared to their previous results with more classes. For BART, the accuracy increases to 0.86 from 0.77, precision to 0.86 from 0.76, recall to 0.86 from 0.77, and F1 score to 0.85 from 0.75. Similarly, BERT's scores improve, with accuracy at 0.83 from 0.74, precision at 0.84 from 0.73, recall at 0.83 from 0.74, and F1 score at 0.83 from 0.72. The reduction in the number of classes simplifies the classification task, making it easier for the models to distinguish between categories. This simplification reduces the risk of misclassification, as there are fewer categories to choose from, leading to higher overall performance metrics.

For polarity prediction, the performance metrics remain consistent with the previous classes for the category label, with less significant gains. BART continues to outperform BERT significantly. BART achieves an accuracy of 0.90, precision of 0.89, recall of 0.90, and F1 score of 0.89. In contrast, BERT's performance metrics are accuracy at 0.83, precision at 0.82, recall at 0.83, and F1 score at 0.82.

Because of how we divided the ABSA task, it is hard to get a one-to-one comparison with other state-of-the-art solutions.

Figure 24: **accuracy for training and validation data over the epochs for the BERT pipeline architecture (ACERC)**

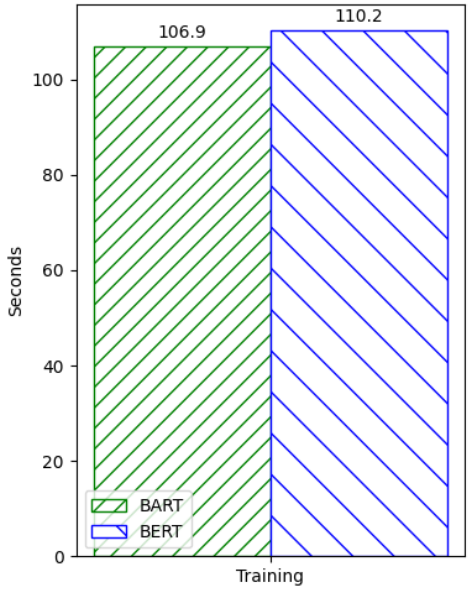

Figure 25: **Training time (ACERC)**

6 CONCLUSIONS

For the state-of-the-art solutions presented in chapter **2**, the majority of them are based on transformer-based architectures like BART and BERT combined with complex models that outperform other state-of-the-art solutions with simpler pipeline architectures, even in low-resource environments, with the expense of more computational overhead.

Our solution uses a simple model architecture with a BiLSTM and a Conv1d layer combined with a BART transformer model for text processing which results in low training times. Alongside these 2 layers, we also have a max pooling, two dropouts, and 2 fully connected layers, one for each prediction label. Adding to this the fact that we split the ABSA task in the ACE task and the ASE task, the ignoring of the term extraction alongside the category and polarity makes it harder for the model to achieve a higher accuracy or other scores in low-resource environments like the SemEval2016 Task 5 Restaurants dataset [8].

Based on the tests done, presented in section **5.3**, reducing the complexity of the ACE task by narrowing down the label classes to its core ones, cuts them in half, and manifests in substantial performance gains, especially for the category classification. We also observe improvements for the ASE task, but not that significant.

Future related work for this task would be to test it on multiple datasets and try to make the model more generalized not only specialized on a single dataset. Another thing that can be done for further improvements is to experiment with different dropout rates, filter sizes, and additional regularization techniques to further enhance model performance. Also, it will be also nice to see, how the training times can improve for our model by training it on the GPU instead of the CPU. Moreover, integrating aspect term extraction with category and polarity predictions could potentially provide additional contextual information, thereby improving the model's predictive accuracy.

Finally, based on the results and research done for this paper, a model architecture that combines these elements, concluded in this chapter, could facilitate in-depth comparisons with other state-of-the-art solutions as the one highlighted in this document, and for sure will lead to better overall performances.

BIBLIOGRAPHY

[1] Karimi Akbar, Rossi Leonardo, and Prati Andrea. Improving bert performance for aspect-based sentiment analysis. *Website. Available online at https://aclanthology.org/2021.icnlsp-1.5*, 2023.

[2] Pouran-Ben-Veyseh Amir, Nouri Nasim, Dernoncourt Franck, Hung-Tran Quan, Dou Dejing, and Huu-Nguyen Thien. Improving aspect-based sentiment analysis with gated graph convolutional networks and syntax-based. *Website. Available online at http://dx.doi.org/10.18653/v1/2020.findings-emnlp.407*, 2020.

[3] Elena-Simona Apostol, Alin-Georgian Pisică, and Ciprian-Octavian Truică. Atesa-bÆrt: A heterogeneous ensemble learning model for aspect-based sentiment analysis. *Website. Available online at https://arxiv.org/abs/2307.15920*, 2023.

[4] Chen Chenhua, Teng Zhiyang, Wang Zhongqing, and Zhang Yue. Discrete opinion tree induction for aspect-based sentiment analysis. *Website. Available online at http://dx.doi.org/10.18653/v1/2022.acl-long.145*, 2022.

[5] Yan Hang, Dai Junqi, Ji Tuo, Qiu Xipeng, and Zhang Zheng. A unified generative framework for aspect-based sentiment analysis. *Website. Available online at http://dx.doi.org/10.18653/v1/2021.acl-long.188*, 2021.

[6] Xu Hu, Liu Bing, Shu Lei, and Yu Philip. Dombert domain-oriented language model for aspect-based sentiment analysis. *Website. Available online at http://dx.doi.org/10.18653/v1/2020.findings-emnlp.156*, 2020.

[7] Wang Kai, Shen Weizhou, Yang Yunyi, Quan Xiaojun, and Wang Rui. Relational graph attention network for aspect-based sentiment analysis. *Website. Available online at http://dx.doi.org/10.18653/v1/2020.acl-main.295*, 2020.

[8] Pontiki Maria, Galanis Dimitris, Papageorgiou Haris, Androutsopoulos Ion, Manandhar Suresh, AL-Smadi Mohammad, Al-Ayyoub Mahmoud, Zhao Yanyan, Qin Bing, De-Clercq Orphée, Hoste Véronique, Apidianaki Marianna, Tannier Xavier, Loukachevitch Natalia, Kotelnikov Evgeniy, Bel Nuria, Jiménez-Zafra Salud, María, and Eryiğit Gülşen. Semeval-2016 task 5 aspect based sentiment analysis. *Website. Available online at http://dx.doi.org/10.18653/v1/S16-1002*, 2016.

[9] Jiang Qingnan, Chen Lei, Xu Ruifeng, Ao Xiang, and Yang Min. A challenge dataset and effective models for aspect-based sentiment analysis. *Website. Available online at http://dx.doi.org/10.18653/v1/D19-1654*, 2019.

[10] Chen Shaowei, Liu Jie, Wang Yu, Zhang Wenzheng, and Chi Ziming. Synchronous double-channel recurrent network for aspect-opinion pair extraction. *Website. Available online at http://dx.doi.org/10.18653/v1/2020.acl-main.582*, 2020.

[11] Wu Zhen, Ying Chengcan, Zhao Fei, Fan Zhifang, Dai Xinyu, and Xia Rui. Grid tagging scheme for aspect-oriented fine-grained opinion extraction. *Website. Available online at http://dx.doi.org/10.18653/v1/2020.findings-emnlp.234*, 2020.