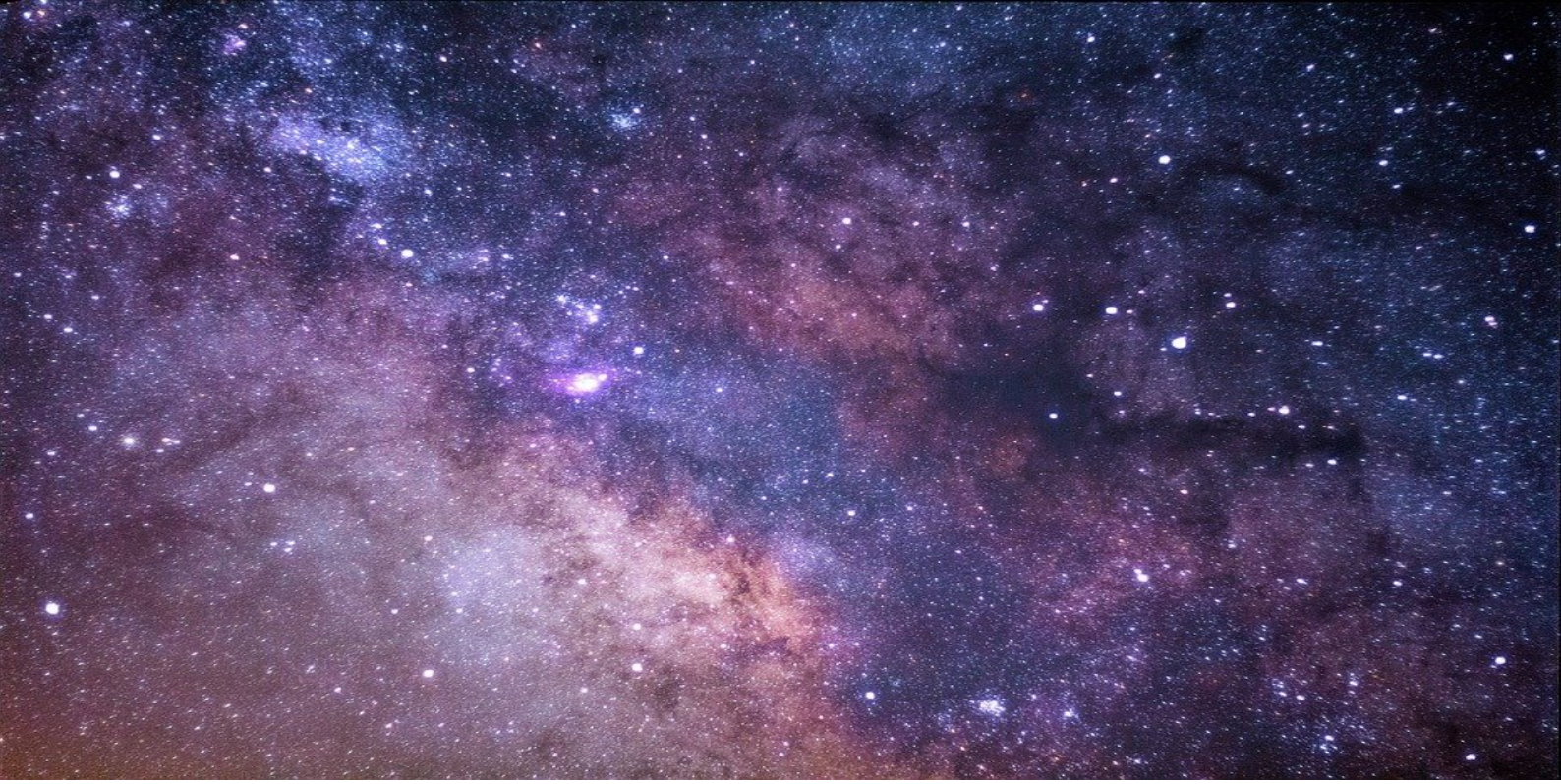


Using Large Language Models for ABSA

Aspect Based Sentiment Analysis

Sadiksha Baral

Business Data Science, AAU Business School

Master's Thesis

Here you can write something about which tools and software you have used for typesetting the document, running simulations and creating figures. If you do not know what to write, either leave this page blank or have a look at the colophon in some of your books.

AAU Business School
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Using Large Language Models for Aspect Based Sentiment Analysis

Theme:
Natural Language Processing

Project Period:
Spring semester 2024

Project Group:
XXX

Participant(s):
Sadiksha Baral

Supervisor(s):
Hamid Bekamiri

Copies: 1

Page Numbers: 65

Date of Completion:
June 2, 2024

Abstract:

This thesis investigates the use of Large Language Models (LLMs) for Aspect-Based Sentiment Analysis (ABSA). We start with traditional machine learning techniques and the move on to advanced deep learning models. The models were assessed based on their effectiveness in category classification, sentiment polarity classification, and joint classification tasks.
Fine-tuning experiments with OpenAI's GPT-3.5 Turbo model showed significant improvements in performance, highlighting the benefits of model adaptation. Despite these advances, limitations such as data availability, computational resource requirements, and model hallucinations were identified. The study recommends future work on expanding datasets, optimizing model architectures, and exploring advanced fine-tuning techniques to enhance the robustness and accuracy of LLMs for ABSA in real-world applications.

AAU Handelshøjskole
Aalborg Universitet
http://www.aau.dk

AALBORG UNIVERSITET
STUDENTERRAPPORT

Titel:
Brug af store sprogmodeller til aspektbaseret sentimentanalyse

Tema:
Naturlig sprogbehandling

Projektperiode:
Forårssemester 2024

Projektgruppe:
XXX

Deltager(e):
Sadiksha Baral

Vejleder(e):
Hamid Bekamiri

Oplagstal: 1

Sidetal: 65

Afleveringsdato:
Juni 2, 2024

Abstract:

Dette speciale undersøger brugen af store sprogmodeller (LLM'er) til Aspect-Based Sentiment Analysis (ABSA). Vi starter med traditionelle maskinlæringsteknikker og går videre til avancerede deep learning-modeller. Modellerne blev vurderet ud fra deres effektivitet i kategoriklassificering, sentimentpolaritetsklassificering og fælles klassifikationsopgaver. Finjusteringseksperimenter med OpenAI's GPT-3.5 Turbo-model viste betydelige forbedringer i ydeevnen, hvilket fremhævede fordelene ved modeltilpasning. På trods af disse fremskridt blev begrænsninger som datatilgængelighed, krav til beregningsressourcer og modelhallucinationer identificeret. Undersøgelsen anbefaler fremtidigt arbejde med at udvide datasæt, optimere modelarkitekturer og udforske avancerede finjusteringsteknikker for at forbedre robustheden og nøjagtigheden af LLM'er til ABSA i applikationer i den virkelige verden.

Contents

Preface

This master's thesis, titled "Using Large Language Models for Aspect-Based Sentiment Analysis," is submitted to the AAU Business School, Aalborg University, in fulfillment of the requirements for the Masters degree in Business Data Science. The research presented in this thesis was conducted during the Spring semester of 2024 under the supervision of Professor Hamid Bekamari.

The field of Natural Language Processing (NLP) has witnessed significant advancements in recent years, driven by the development of Large Language Models (LLMs). This thesis aims to explore the application of these advanced models in Aspect-Based Sentiment Analysis (ABSA), a sub-field of sentiment analysis that focuses on identifying and extracting sentiments related to specific aspects or components within a text. By evaluating various model architectures, this research seeks to contribute to the ongoing efforts to enhance the accuracy and efficiency of sentiment analysis in business applications.

The motivation for this study stems from the growing importance of understanding customer sentiments and opinions expressed in online reviews, social media, and other textual data sources. Effective sentiment analysis can provide valuable insights for businesses, enabling them to improve their products, services, and customer satisfaction.

I would like to express my deepest gratitude to my supervisor, Professor Hamid Bekamari, for his invaluable guidance, support, and encouragement throughout this research. I am also thankful to my family and friends for their continuous support and understanding during the course of my studies.

Finally, I would like to acknowledge the contributions of the researchers and developers whose work on NLP and LLMs has laid the foundation for this thesis. Their innovations have been instrumental in advancing the field and enabling new possibilities for sentiment analysis.

Sadiksha Baral
<sa73bi@student.aau.dk>
Aalborg, June 2024

Chapter 1

Introduction

In recent years sentiment analysis has gained significant attention due to its potential applications in various domains such as emotion detection, customer feedback analysis, political opinions,and social media monitoring. According to Zhang et al., 2023 Aspect Based Sentiment Analysis (ABSA) is a significant area of research within the field of fine-grained sentiment analysis that aims to identify and extract the sentiment of specific aspects or components of a product or service. For example, in the sentence "The restaurant was expensive, but the menu was great", there are two aspects that "price" and "food" that are associated with negative and positive sentiment respectively. The aspects can be defined as exactly the words present in the document or as a broader category as well and we will be using the later definition.

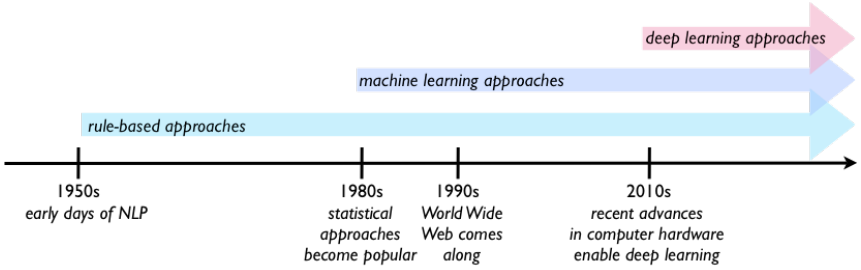


Figure 1.1: History of NLP (Source: Getting started with NLP book[1])

The field of NLP started with rule-based systems and has evolved to semantically aware models today. NLP was mainly fueled by shift from rule-based taxonomies in the 1960s to machine learning algorithms in the late 1980s. The early models were limited in terms of task specificity and language dependence. Since the introduction of Transformers followed by models like BERT and GPT, they have

[1]https://livebook.manning.com/book/getting-started-with-natural-language-processing/chapter-1/

had big impact on the whole NLP community with their amazing understanding of various NLP tasks. They can directly perform tasks in zero-shot or few-shot in-context learning manner and achieve strong performance without the need for any supervised training. This has sparked a new revolution and growth of interest in the field of NLP. Through this work we want to understand how we have reached here by using ABSA as the task of choice. The history is briefly summarized in Figure 1.1.

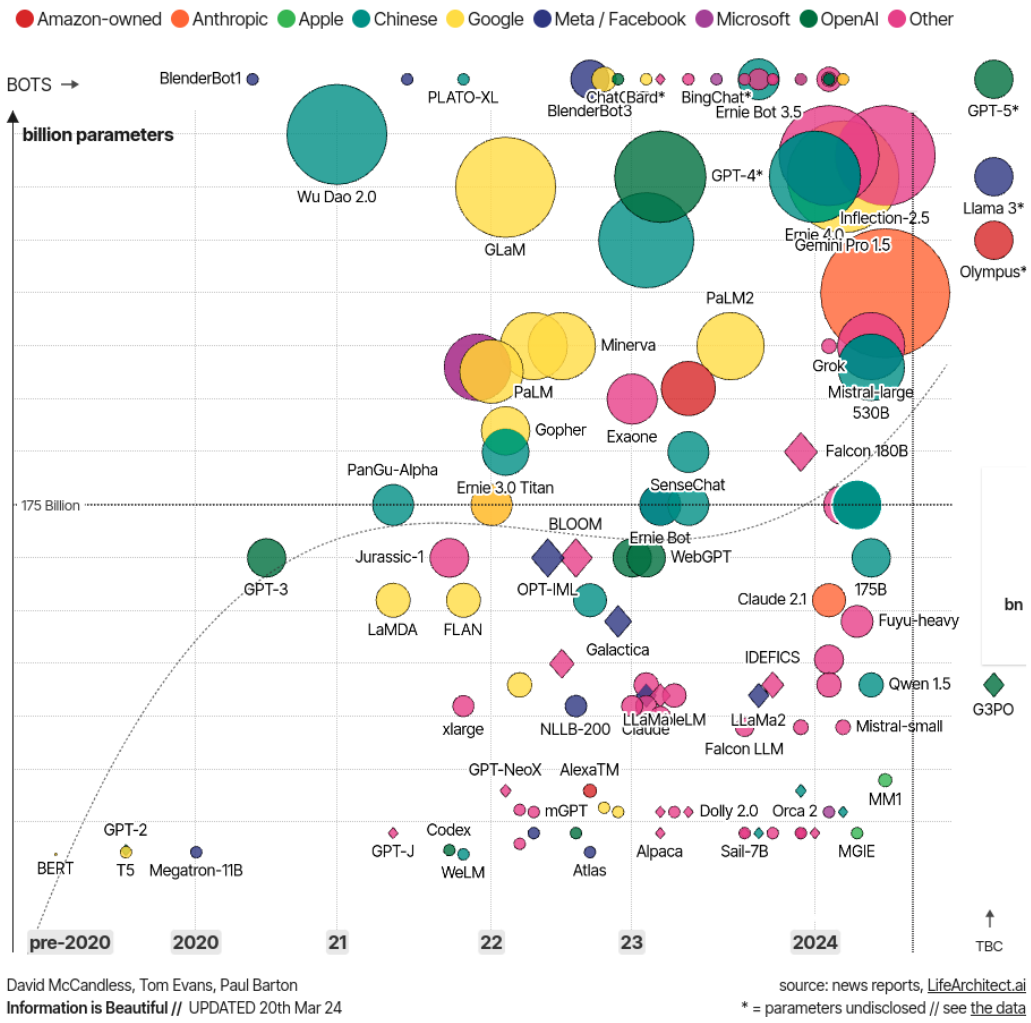


Figure 1.2: Comparision of recents LLms interms of size (Source: news reports[2])

Then in recent years, Large Language Models (LLMs) have risen to popularity and Generative AI has become the buzz word. These models are groing in size rapidly with GPT models having parameter sizes in Trillion. Recently, however

[2]https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgp

there also has been a push towards researching capabilities of smaller models as well. In Figure 1.2, we can see evolution of LLMs along with their parameter sizes.

Sentiment analysis helps businesses to understand their customers and by performing the analysis at aspect level analysis businesses gain even deeper insights. It allows you to analyze the sentiment associated with specific aspects or features of your product or service, rather than just looking at overall sentiment. Businesses can understand their strengths and weaknesses. It can help guide marketing and communication and make those efforts more data driven. Then by having the analysis over an extended period of time, a business can even monitor how their policies are impacting customer satisfaction. With these new models available as services, any business can get access to their power with ease. With proper pipeline and data cleaning, reviews from several sources can be fed into a single model. Then having access to a dashboard that shows the results on a single page means anyone can understand the results easily.

1.1 Objectives

With this work we mainly want to explore how Large Language models can be utilized for the task of ABSA. Particularly we have set following objectives for the work :

1. Understand the architecture and concept of Large Language Models
2. Explore the effective of Zero shot learning and few shot learning techniques for ABSA
3. Prompt engineering and fine tuning to understand the effect a prompt has on the outputs of language models
4. Experiments with different LLMs (both closed and open source)

1.2 Document Structure

This document uses AAU report template available on Overleaf. The document is divided into 5 chapters. The chapters are organized as follows :

- Chapter 1 gives a quick summary of the work done. It also provides the motivation, objectives and contributions of the research.
- Chapter 2 provides background knowledge and contextual information related with the work. It also provides information about several possible approaches that are taken for ABSA. We explore historical approaches and models along with concepts related to Large Language Models.

- Chapter 3 describes the datasets used, preprocessing done on the data and finally implementation of the models. It also describes the training process and evaluation methods used. It also has inference and finetuning techniques related with LLMs.
- Chapter 4 compares the results of our implemented models with each others and also with other literature that have been talked about throughout the research.
- Chapter 5 concludes the work and provides the findings in brief. This chapter also provides the limitations of the work and some suggestions for future work.

Chapter 2

Background

In recent decades there has been vast amount of text data in forms of reviews and comments available on the internet. As a result of this ever growing data, Natural Language Processing has taken a big leap in recent years. Organizations and businesses are keen to extract valuable insights from customer reviews, social media comments, and other textual sources. Recent breakthroughs and data availability has made it easier to understand and analyse public sentiments and opinions. Aspect-Based Sentiment Analysis (ABSA) and Transformers have emerged as powerful tools for extracting fine-grained sentiment information from text data. In this section, we will explore the development of ABSA and use of Machine Learning for such analysis.

2.1 Natural Language Processing

Natural language refers to text, speech or other ways which we use to communicate with each other and Natural Language Processing (NLP) is an area of artificial intelligence (AI) that focuses on the research and application exploring how computer systems can be used to understand and process the natural language. Natural Language Processing helps AI models understand and interpret human language. It has grown to be one of the most powerful field of AI. With the vast availability of data NLP has grown a lot in recent years.

One of its primary uses aspects of NLP is sentiment analysis, where businesses can gauge public opinion on products and services by analyzing customer reviews and social media posts. This insight helps in understanding customer satisfaction, brand reputation, and identifying areas for improvement. Furthermore, NLP is integral in customer support chatbots, which can provide real-time assistance to customers, leading to improved customer service and cost reduction. Additionally, NLP can be employed in a wide range of applications including language translation, text summarizing, text generation, sentiment analysis and a lot more.

In early days, NLP tasks were with dictionary matching, grammatical rules or simple statistical approaches. In recent decades those early approaches were replaced by early machine learning approaches involving static word embeddings (Word2Vec and GloVe) which represent word as vectors in a high dimensional space. These methods captured the semantic and syntactic relationships between words based on co-occurrence statistics. Then in recent years, NLP has boomed a lot due to advancements in Deep Learning and some recent models have outperformed humans in some NLP tasks (Lauriola et al., 2022). Most of these deep learning models rely on LSTM or Transformers. Transformer based language models like BERT and GPT use embeddings that are contextual and are generated based on input sentence or sequence of words. With contextual embeddings, same word can have a different embedding vector based on the input sequence. The contextual information captured by these embeddings can help improve the performance of NLP tasks, such as sentiment analysis.

2.2 Sentiment Analysis

Sentiment Analysis (SA), sometimes also known as opinion mining (OM) is the computational study of peoples opinions, attitudes and emotions toward an entity which can represents individuals, events, or topics. While the terms Sentiment Analysis and Opinion Mining are often used interchangeably, Tsytsarau and Palpanas (2012) suggest slight differences and stated Opinion Mining extracts and evaluates people's opinions, while Sentiment Analysis identifies and analyzes the sentiment expressed in a text.

As defined by Techtarget (2023), sentiment analysis has emerged as a critical tool across various industries due to its multifaceted applications. These tools are indispensable for monitoring and understanding customer sentiments expressed on social media platforms such as Facebook, Instagram, and Twitter, thereby bolstering the effectiveness of social media marketing strategies. Furthermore, they play a central role in evaluating brand awareness, reputation, and popularity, offering insights that are invaluable in the moment and over time. It is equally essential for organizations in gauging consumer reactions to new products and features, facilitating iterative product improvement and refinement. It is also instrumental in assessing the success of marketing campaigns, identifying specific target audiences, and conducting comprehensive market research to uncover emerging trends and competitive insights.

Medhat et al. (2014) describe three primary levels of Sentiment Analysis (SA): document-level, sentence-level, and aspect-level SA. Document-level SA focuses on categorizing an entire opinion document as conveying a positive or negative sentiment, treating the document as a single information unit discussing one topic. Sentence-level SA, on the other hand, seeks to classify the sentiment

expressed in individual sentences. It starts by identifying whether a sentence is subjective or objective, and if it's subjective, it then determines whether the sentence conveys a positive or negative opinion. Aspect-level Sentiment Analysis focuses on categorizing sentiment based on specific aspects of entities. It begins by identifying these entities and their associated aspects.

Document Level		
Text	**Sentiment**	
Even though its good seafood, the prices are too high	Neutral	
Sentence Level		
Text	**Sentence**	**Sentiment**
Even though its good seafood, the prices are too high	First	Positive
	Second	Negative
Aspect Level		
Text	**Aspect**	**Sentiment**
Even though its good seafood, the prices are too high	food	Positive
	price	Negative

Figure 2.1: An example review at different levels of Sentiment Analysis

As seen in Figure 2.1, each level gives a more deeper analysis into the sentiments present in the text and depending on the use-case one might be more useful than other.

2.3 Aspect Based Sentiment Analysis

Hoang et al. (2019) stated on his paper that traditionally sentiment analysis focuses on determining the overall sentiment expressed in a text without specifying the subjects or topics under discussion. However, this approach may prove inadequate when a text simultaneously discusses various topics or entities, each potentially conveying different sentiments. To address this challenge and provide a more comprehensive analysis, aspect-based sentiment analysis (ABSA) comes into play.

ABSA is designed to identify and analyze sentiments associated with specific aspects or entities mentioned in the text, offering a more detailed understanding of sentiment in context. Hence, ABSA provides a great opportunity to analyse sentiments (public) over time across different contents present on media. It was first introduced in SemEval-2014 (Pontiki et al., 2014), which provided a dataset with annotated reviews about restaurants and laptops. The ABSA task in SemEval-2014 did not contain full reviews until SemEval-2015 (Pontiki et al.,

2015), and the dataset for SemEval-2016 (Pontiki et al., 2016) did not change from 2015 except for additional test data.

Often times different authors use the same term aspect and ABSA to describe different types of analysis. Some authors use aspect to refer a broader category. As part of this work the terms are defined as :

> **Aspect** refers to category the words present in the input sentence (For example food, price, ambience, service, etc)
>
> **Sentiment** is sentiment classification belonging to a particular aspect
>
> **Sentiment terms** are the words that hold the sentimental value (also called opinion terms)

ABSA is structured into two primary processing phases: Aspect Extraction (AE) and Aspect Sentiment Analysis (ASA). Some authors also explore Sentiment Evolution (SE) as part of their analysis. Nazir et al. (2022) stated in the first phase, aspects are identified, including explicit and implicit aspects, aspect terms, entities, and Opinion Target Expressions (OTE). The second phase involves sentiment polarity classification for predefined aspects, entities, or targets, while also considering interactions and semantic relationships for enhanced sentiment accuracy. The third phase is dedicated to understanding the dynamic nature of people's sentiment towards various aspects or events over time, influenced by social characteristics and personal experiences. In this paper we will focus on aspect extraction and aspect sentiment analysis.

2.3.1 Issues and Challenges

Aspect extraction and classification both come with their own set of issues and challenges. Nazir et al. (2022) did a comprehensive review of the issues and challenges. Some of the challenges are :

1. **Multi aspect and ambiguous language:** In cases with ambiguous language or multiple aspects in a single sentence, identifying and extracting all the aspects is very challenging. Models need to understand the context and sentiment associated with each aspect, which can be complex in the presence of negations, modifiers, or subtle expressions.

2. **Data availability :** LLMs require vast amount of data for training and obtaining labelled data for ABSA is quite difficult. Manually annotating data with aspect and sentiment labels requires domain expertise and can be subjective, leading to potential biases.

3. **Domain knowledge :** ML models trained on one domain may not generalize well to other domains. This can be solved with large amount of data and

requires additional labeled data and careful consideration of domain-specific language and context.

4. **Multilingual Challenges :** ABSA in multilingual settings adds complexity, as sentiment expressions and aspects can vary across languages. In our case, Danish is mixed in the review texts which could be a challenging factor for a model.

5. **Black box models :** Using black box models often gives good results but understanding how and why a model makes specific predictions is challenging.

With careful approach combined with proper data pre-processing, model selection and feature engineering, the impact of these issues and challenges can be minimized.

2.4 Approaches to ABSA

Over the years several methods have been proposed for ABSA like Traditional methods, Machine learning models, Deep learning models, and Language Models.

2.4.1 Traditional approaches

As noted by Ma et al. (2018), ABSA's biggest problem lies in effectively capturing aspect-specific sentiment information from the comment. Although some traditional methods for target sentiment analysis have shown potential success, they tend to be resource-intensive due to their heavy reliance on feature engineering and extensive linguistic resources, as observed in the works of Kiritchenko et al. (2014) and Wagner et al. (2014). The need for such comprehensive linguistic resources can pose a challenge, especially when dealing with diverse domains or industries.

Due to their substantial reliance on feature engineering and large language resources, traditional ABSA techniques have resource intensity issues. They often fail to produce similar level of results as more recent approaches while using more resources. These difficulties may restrict the scalability and flexibility of conventional techniques, leading to the investigation of different strategies for more effective and efficient ABSA.

Model	**Task**	**Dataset**	**f1 score**	**Cite**
Rule-based	Sentiment	semeval14 restaurant	77.8%(acc)	Wagner et al. (2014)

Table 2.1: Traditional Approaches to ABSA

2.4.2 Machine Learning based approaches

To perform natural language processing with machine learning algorithms, language must be mathematically represented, and one way to achieve this is through word vectors. Different authors used different vectorization techniques such as Bag of Words, Term Frequency Inverse Document Frequency (TF-IDF) (Bhoi & Joshi, 2018), word2vec (Wei et al., 2020). TF-IDF is a sparse, count-based representation, while Word2Vec provides dense, context-based word vectors that capture semantic meaning. TF-IDF is primarily used for text classification and information retrieval, while Word2Vec is used for capturing semantic relationships and word similarities.

Previously Support Vector Machines(SVM) was also used for NLP tasks. SVMs are a machine learning classification technique which use a function called a kernel to transform data points. Wagner et al. (2014) trained a four way SVM classifier to perform Aspect based polarity classification. Through their experiments they discovered that traditional rule-based approaches outperformed their early ML approaches in some cases. Those cases were more common when there was a sentence with rare word with strong polarity. Except that they discovered the SVM system outperforms the traditional rule based approach. Onwuegbuche et al. (2019) used SVM classifier for sentiment analysis of Nigerian banks twitter data and achieved f1 score of 0.7180.

Model	**Task**	**Dataset**	**score**	**Cite**
SVM	Joint	semeval16 restaurant	0.7303(f1)	Jihan et al. (2017)
SVM	Sentiment	Five Nigerian banks twitter data	0.7180(f1)	Onwuegbuche et al. (2019)
SVM	Sentiment	semeval14 restaurant	81%(acc)	Wagner et al. (2014)
SVM	Sentiment	Macedonian restaurant reviews	0.84(f1)	Najkov (2023)
RandomForest	Sentiment	Macedonian restaurant reviews	0.78(f1)	Najkov (2023)
XGBoost	Sentiment	Macedonian restaurant reviews	0.81(f1)	Najkov (2023)

Table 2.2: Machine learning based approaches

2.4.3 Deep learning based approaches

In recent years the early ML approaches for NLP tasks are being replaced and outperformed by recurrent architectures and deep learning models. Deep learning techniques have been increasingly applied to Aspect-Based Sentiment Analysis (ABSA) and have shown significant improvements in the accuracy and effectiveness of aspect based sentiment analysis. Deep learning approaches use neural networks (NN) to learn the semantic and syntactic features automatically and perform well in the related extensive experiment, in contrast to machine learning approaches that mostly depend on the quality of the created features. Najkov (2023) saw different models from Decision tress to transformers for sentiment analysis task. They did sentiment analysis on Macedonian Restaurant

Reviews and concluded that deep learning models and transformers, outperform traditional machine learning models like Random Forests and Support Vector Machines. Zhu et al. (2022) also used deep learning approach for ABSA and were able to obtain similar results.

Deep neural network (DNN) is more useful for NLP due to their use of (i) dense word embedding, (ii) the inclusion of multiple hidden layers between the input and output, and (iii) the presence of output units (Do et al., 2019).

The difference between NN and DNN is that DNN has more hidden layers (at least two layers theoretically). The main idea of DNN is to regard the output of the previously hidden layer as the input of the current hidden layer to obtain more abstract high-level features. In order to avoid gradient disappearance or gradient explosion, weight initialization in neural network is also an important task.

Convolution Neural Network (CNN) is one of the most successful deep learning system mainly used for image categorization. But a lot of works have used CNN for tasks such as question answering, SA, and machine translation. Some researchers have used CNN for ABSA tasks, and the results have been promising. It mainly consists of an input layer, a convolution layer, a pooling layer and a full connection layer. Shu et al. (2019) used a Double Embedding CNN that uses a domain specific CNN to perform ABSA without adding any manual features. Noh et al. (2019) used two CNNs to extract positional information and classify targets. Wang et al. (2021) proposed a Unified Position aware Convolutional Neural Network (UP-CNN) for ABSA which used the concept of aspect mask to more efficiently obtain the context information.

The RNN model has two important features compared to the feed-forward neural network. First, unlike the CNN has different parameters at each layer, the parameters in RNN are the same in each steps, which then reduces the number of parameters needed to learn (Zhang et al., 2018).

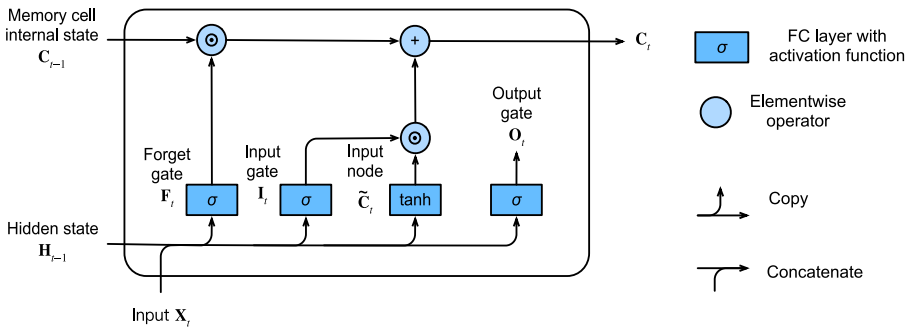


Figure 2.2: An LSTM cell (Source: dl2.ai[1])

[1]https://d2l.ai/chapter_recurrent-modern/lstm.html

Long Short Term Memory (LSTM) is a type of Recurrent Neural Network meaning a neural network with chain of several units one after another and have the ability to process a sequence of inputs like text sentences. To overcome the vanish gradient problem present in RNN models LSTM was developed and achieved superior performance (Hochreiter & Schmidhuber, 1997). It introduces a memory cell that allows the network to selectively forget or remember information over time. As seen on Figure 2.2, LSTM has three special gates, Input gate, Forget gate, and Output gate. Input gate controls what information must be stored in the cell state and also takes current input data and previous hidden state as input. Forget gate decides which information from the previous cell state should be discarded, and Output gate determines the amount of information to be output from the current cell state. Since LSTM can capture sequence model, most researchers use LSTM based method in text classification and sentiment classification tasks, especially in ABSA. Tang et al. (2016) proposed Target-dependent sentiment classification (TD-LSTM), which use two LSTM neutral networks and the target is placed in the middle of sentence. Wang et al. (2016) also proposed new idea with adding aspect embedding in LSTM (ATAE_LSTM) also includes the attention mechanism. The combination of the aspect attention and sentiment attention was proposed by Cai and Li (2018) and called Joint attention LSTM network (JAT-LSTM) and has the highest accuracy among these models.

Model	**Task**	**Dataset**	**Score**	**Cite**
CNN	Category	semeval16 Restaurant	0.7564(f1)	Shu et al. (2019)
LSTM	Category	semeval14 Restaurant	82%(acc)	Wang et al. (2016)
TD-LSTM	Category	semeval14 Restaurant	84%(acc)	Wang et al. (2016)

Table 2.3: Deep learning based approaches

2.4.4 Transformer based approaches

Large Language Models (LLMs) are the most recent advancement made in the field of NLP. These models such as BERT (Devlin et al., 2018), GPT (Brown et al., 2020), PaLM (Chowdhery et al., 2022), Flan-UL2 (Tay et al., 2023) and LLaMA (Touvron et al., 2023) are massive in size with hundreds of millions to billions of parameters. Trained on vast amount of text data, LLMs have the ability to understand and generate human language. They can perform a wide range of NLP applications from text classification and sentiment analysis to language translation and text generation. Their capacity for transfer learning where they are pre-trained on general language understanding and then fine-tuned for specific tasks makes them versatile tools for a wide array of language-related tasks.

LLMs are built upon the Transformer architecture which uses the concept of

attention, specifically self attention mechanism. Attention mechanism allows the model to weigh the importance of different parts of an input sequence when processing information. This attention mechanism enables LLMs to capture complex linguistic dependencies, context, and long-range relationships, making them highly effective at understanding and generating human language.

We can use different language models for several NLP tasks easily with the help of packages like transformers[2] and huggingface[3]. Transformers is an open-source library that was developed to help advance the opensurce machine learning research community (Wolf et al., 2020). Along with huggingface portal, the library enables any users to use state-of-the art Transformer architectures under a unified API.

Transformer

Transformer is a type of neural network which pays attention to different parts of input data simultaneously and allows it to understand relationships in a sequence. It was introduced on the paper "Attention Is All You Need" (Vaswani et al., 2017). The paper presents a deep learning architecture that leverages the power of self mechanism. Transformers use an encoder-decoder structure for tasks with varying input and output lengths. The overall architecture can be seen on Figure 2.3. It has become the dominant architecture for natural language processing, surpassing alternative neural models such as convolutional and recurrent neural networks in performance for tasks in both natural language understanding and natural language generation.

The key component of Transformer model is the concept of "attention," which allows the model to concentrate on different parts of the input sequence when producing the output sequence. Transformers uses Self-Attention Mechanism, Multi-Head Attention, Positional Encoding and Encoder-Decoder Architecture.

Self-Attention Mechanism allows the model to weigh the importance of different words in a sequence when processing a particular word.

Multi-Head Attention capture different aspects of relationships in the input data

Positional Encoding are added to the input embeddings to provide information about the positions of words in a sequence as transformer don't understand the order of input.

Encoder-Decoder Architecture encode to process the input sequence and a decode to generate the output sequence

[2]https://huggingface.co/docs/transformers/index
[3]https://huggingface.co/

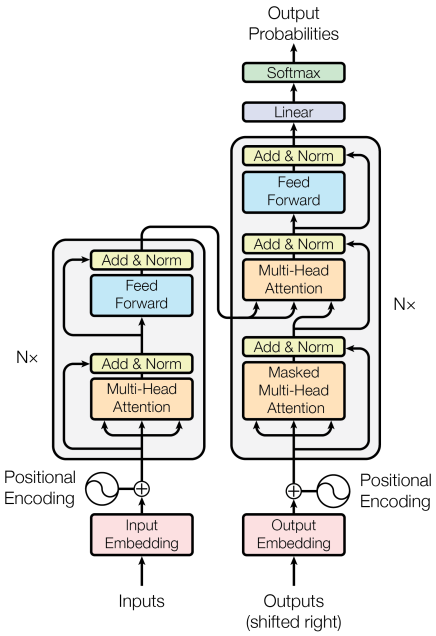


Figure 2.3: Transformer model architecture. (Source: Vaswani et al., 2017)

The transformer has proven highly effective in natural language processing tasks, such as machine translation, text summarization, and language modeling. It has also been adapted and extended for various other domains beyond NLP. Language models like BERT and GPT has also used transformers.

Binary Encoder Representation from Transformer (BERT)

The pre-training language model attracted much interest from academia and industry because it performs well in several NLP applications. BERT is the first deeply bidirectional and unsupervised language representation model developed by Google. It was introduced in a research paper titled "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," by Devlin et al. (2018). It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. BERT uses a transformer-based architecture, which allows it to take into account both the context of a given word and the words that come before and after it.

BERT is pre-trained on a large corpus by using the Masked Language Modeling (MLM) and next-sentence prediction objectives and due to its architecture it develops an understanding of the language. As seen on Figure 2.4, other than the output layers, same architecture are used in both pre-training and fine tuning procedures. Same pre-trained model parameters are used to initialize models

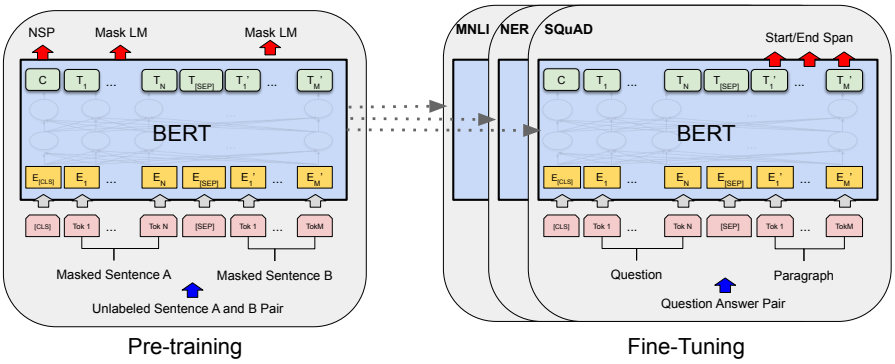


Figure 2.4: BERT training and fine-tuning procedure (Source: Figure 1 in Devlin et al., 2018)

for different down-stream tasks. Once trained, the resulting model can then be fine-tuned on a small labelled dataset for a specific NLP task. Hao et al. (2019) visualized the loss landscapes and optimization trajectories of the BERT fine-tuning procedure. Their results indicate fine-tuning BERT tends to generalize better because of easier optimization compared with training from scratch. They also demonstrate that fine-tuning procedure is robust to overfitting, even though BERT is highly over-parameterized for downstream tasks.

BERT has shown to produce good results on NLP tasks (Wang et al., 2018) due to the large amounts of text it has been trained on. Some of the researchers have explored using BERT in combination with other ML methods. Mewada and Dewang (2023) proposed synthetic attention in bidirectional encoder representations from transformers (SA-BERT) with an extreme gradient boosting (XGBoost) classifier to classify sentiment polarity in the review dataset and the experimental outcomes show SA-BERT-XGBoost model has better accuracy and less time overhead in sentiment analysis of reviews than base models.

Generative Pre-trained Transformer (GPT)

GPT is a Transformer based architecture and training process for NLP tasks, introduced by Radford and Narasimhan (2018). It is trained on vast amounts of diverse data, enabling it to capture complex patterns and relationships in language. GPT models follow a two step process involving pre-training and fine-tuning. Pre-training means it learns from a broad range of text before being fine-tuned for specific tasks, making it versatile for various applications, including Aspect-Based Sentiment Analysis (ABSA). This process allows the model use general understanding of language gained during pre training to excel at tasks during fine tuning.

In context of ABSA, GPT can be used to get an innovative and promising approach. Due to the amount of training data and general language

understanding, GPT can easily learn detailed features and representations from data, removing the need for manual feature engineering. GPT's contextual awareness allows it to obtain the sentiment associated with specific aspects mentioned in a comment or review.

There has been some work in experimenting with models that integrate GPT with other simpler ML approaches. Mingzheng et al. (2023) proposed an aspect-level sentiment analysis model that interates GPT with multi-layer attention (GPT and Multi-Layer Attention Network, GPT-MAN). Initially, the model integrates GPT and aspect coding to create a new word vector representation approach. Then inputs word vector with rich semantic information into the network featuring a multi-layer attention mechanism, leading to better accuracy in sentiment analysis. They experiment and optimize the hyperparameters using of Restaurant, Laptop and Twitter datasets.

Simmering and Huoviala (2023) evaluated the effectiveness of GPT-4 and GPT-3.5 in zero-shot, few-shot, and fine-tuned scenarios for aspect-based sentiment analysis (ABSA). They were able to F1 score of 83.8 on the SemEval-2014 Task 4 joint aspect term extraction and polarity classification, outperforming InstructABSA (Scaria et al., 2023) by 5.7%. InstructABSA was an approach to ABSA subtask where the model was trained with instruction based prompts. It used tk-instruct as its base and were able to surpass much larger models.

Recently ChatGPT is being used for a lot of NLP tasks and has drawn attention of research community as well. Wang et al. (2023) evaluated performance of ChatGPT on sentiment analysis. They were trying to evaluate ChatGPT's understanding of opinions, sentiments, and emotions contained in the text. They concluded ChatGPT performed well in zero-shot sentiment analysis and was able to comparable results with fine-tuned BERT and SOTA models trained with labeled data in respective domains. Their study suggests ChatGPT could be used in other sentiment analysis sub-tasks to obtain comparable results without any further fine-tuning as well.

Model	**Task**	**Dataset**	**Score**	**Cite**
BERT based model	Sentiment	Macedonian restaurant reviews	0.89(f1)	Najkov (2023)

Table 2.4: Transformer Approaches comparision

Sentence transformer (SBERT)

Sentence transformers are a type of machine learning model specifically designed for natural language processing (NLP) tasks. These models are based on transformer architectures and are particularly optimized to produce high-quality, dense vector representations (embeddings) of sentences and text.

In the paper "Sentence-BERT: Sentence Embeddings using Siamese

BERT-Networks" (Reimers & Gurevych, 2019), present Sentence-BERT (SBERT), which modifies the pretrained BERT network by employing siamese and triplet network structures to generate semantically meaningful sentence embeddings that can be compared using cosine similarity. This approach significantly reduces the time required to find the most similar sentence pairs from 65 hours with BERT/RoBERTa to approximately 5 seconds with SBERT, while maintaining BERT's level of accuracy.

Mistral models

The Mistral model represents a significant advancement in natural language processing (NLP), building on enhanced transformer architectures to deliver superior performance and efficiency. By incorporating a sparse attention mechanism and optimizing computational pathways, Mistral reduces complexity and improves processing speed. This allows the model to handle longer sequences and larger datasets effectively. Its advanced contextual understanding capabilities lead to more accurate and coherent outputs, making it highly effective for tasks such as machine translation, text summarization, sentiment analysis, and question answering. The model's scalability and efficiency improvements position it as a powerful tool for a wide range of NLP applications, underscoring its contribution to advancing state-of-the-art language technologies (Jiang et al., 2023).

Mistral also has a variant called Mixtral which is a Mixture of Experts (MoE) model with 8 experts per MLP, with a total of 45 billion parameters. Mixture of Experts enable models to be pretrained with far less compute, which means you can dramatically scale up the model or dataset size with the same compute budget as a dense model. In particular, a MoE model should achieve the same quality as its dense counterpart much faster during pretraining (Jiang et al., 2024).

Phi

The Phi-3 model is a 3.8 billion parameter language model introduced by Microsoft(Abdin et al., 2024). It was trained on 3.3 trillion tokens. By introducing novel mechanisms to refine attention mechanisms and handle long-range dependencies more effectively, the Phi-3 model enhances contextual understanding and accuracy. It is designed to be computationally efficient, making it faster and more scalable without sacrificing performance. This efficiency allows the Phi-3 model to handle larger datasets and more complex tasks, making it suitable for a wide range of applications such as machine translation, text summarization, sentiment analysis, and question answering. Microsoft and the researchers use SLM (Small Language Model) to descibe phi family of models as they have smaller number of parameters than the current state of the art language models.

2.5 Large Language Models (LLM)

Language models (LMs) are computational systems with the ability to comprehend and generate human language. LLMs can be broadly categorized into three groups based on their architectural design. The first group comprises encoder-only models, like BERT (Devlin et al., 2018) and its variants. These models employ a pre-training followed by fine-tuning approach for natural language processing (NLP) tasks, utilizing masked language models as the primary training objective during pre-training and fine-tuning on annotated downstream datasets. The second group consists of decoder-only models, such as GPT, (Radford et al., 2018) which leverage the decoder of an auto-regressive transformer model to predict the next token in a sequence. These model also follow the pre-training then fine-tuning paradigm. The third category encompasses encoder-decoder models like T5 (Raffel et al., 2023) and its variants.

While classical language models like BERT are efficient on various NLP tasks and trained on large amounts of un-annotated textual data, they still require a substantial amount of annotated data to perform well on targeted tasks such as NER, NLI, and RE. These models also have difficulty generalizing their knowledge to other languages or domains once adapted to a particular task and context. Collecting such data for any scenario is then expensive, as it requires highly qualified annotators and raises privacy concerns.

Recently, LLMs have brought additional performance improvements, especially in generative tasks. These models are composed of billion of parameters and trained on gigantic amounts of data, from various natures, domains and languages.

New approaches using these generative LLMs capabilities have aimed to align them with instructions (Ouyang et al., 2022) giving them greater abilities to handle multiple NLP tasks in multiple languages in zero- or few-shot learning (Bang et al., 2023).

2.5.1 Tokenization and Data processing

Tokenization in Large Language Models (LLMs) is a crucial step that involves breaking down a sequence of text into smaller units called tokens. Tokens can represent words, subwords, or even characters, depending on the tokenization method used. Tokenization provides a structured way to break down text into manageable pieces for the model to process Ghashami (2023).

LLMs are mainly generative models and generate output using the understanding developed from training data. A lot of these models use a filtered dataset to reduce the risk of unwanted or unsafe utterances. This prevents filter out certain personal information and other sensitive data. After cleaning and pre-processing, the cleaned text data tokenize into smaller units such as words or subword pieces (e.g., Byte-Pair Encoding or WordPiece).

2.5.2 Pre-training LLMs

LLMs are trained using huge datasets. LLMs can be taught on datasets larger than a petabyte. A one-gigabyte document can hold around 180 million words. A petabyte can store one million gigabytes of data. This dataset includes text from a variety of sources to guarantee that the model learns a diverse set of linguistic patterns. For example, Gemma 2B and 7B are trained on 2T and 6T tokens comprising predominantly English data from web texts, mathematics, and coding, respectively (google deepmind, 2024). Considering the huge quantity and scale of LLM models and data, model training requires a significant amount of processing power. To reduce training time, it is usual to use a model parallelism feature that divides sections of the model over numerous GPUs.

Additionally, the models also have billions of parameters and require infrastructure equipped with multiple GPUs to facilitate the training of these large models. Training GPT-3, a previous-generation model with 175 billion parameters, would take 288 years to train on one NVIDIA V100 GPU (run:ai, 2023) which itself is a very powerful server GPU. Typically, LLMs are trained on thousands of GPUs in parallel. For example, Google used 6,144 TPUv4 chips to distribute training and develop its PaLM model with 540 billion parameters. For the Gemma 7B model, Google used 16 pods totaling 4096 TPUv5e (google deepmind, 2024).

2.5.3 Zero-shot and Few-shot Learning

Few-shot learning (FSL) is the challenge of learning novel classes with a tiny training dataset of one or a few images per category. FSL is closely related to knowledge transfer where a model, previously trained on large data, is used for a similar task with fewer training data. The more the transferred knowledge is accurate, the better FSL will generalise. Zero-shot learning is a machine learning paradigm where a model is trained to recognize and generalize to classes that it has never seen during the training phase. Zero-shot learning, like all n-shot learning, refers not to any specific algorithm or neural network architecture, but to the nature of the learning problem itself: in ZSL, the model is not trained on any labeled examples of the unseen classes it is asked to make predictions on post-training.

2.5.4 Finetuning

Large language models (LLMs) have revolutionized the field of natural language processing (NLP) but these pre-trained models often lack domain-specific knowledge and can struggle with tasks requiring specialized understanding. Fine-tuning allows us to adapt the pre-trained LLMs to specific tasks and domains. Fine tuning involves adjusting the internal parameters of the LLM based on labeled

data, allowing it to excel in specific applications like ABSA.

Compared to training an LLM from scratch, fine-tuning significantly reduces the computational cost and time required, making it a more efficient approach for quick adaptation to specific tasks. It allows the model to leverage its pre-trained knowledge base while acquiring task-specific knowledge, leading to improved performance on the target task. ALso, some customization could be done to the LLM's output to align with the specific language nuances and requirements of the target domain.

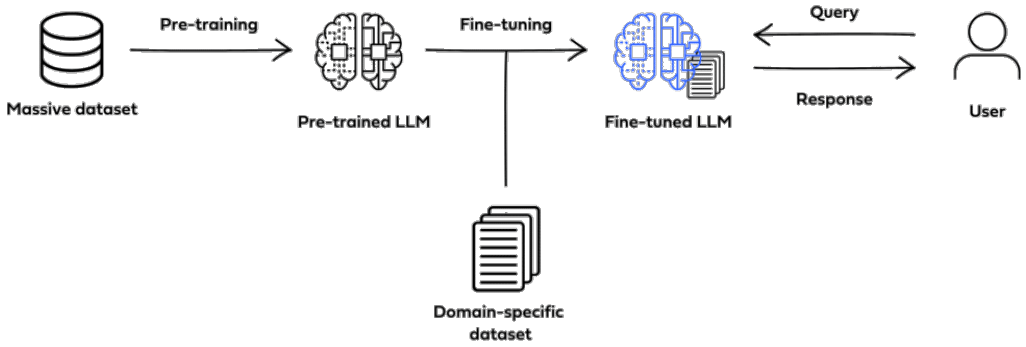


Figure 2.5: An example LLM fine tuning pipeline (Source: deci.ai blog[4])

Figure 2.5 shows a general fine tuning pipeline. If enough resources are available, fine tuning also can be performed locally with the help of libraries like Hugging face. It provides greater control and flexibility but has huge resource requirements. Supervised Fine-Tuning (SFT) and Parameter-Efficient Fine-Tuning (PEFT) have emerged as most popular approaches to performing fine tuning locally. If resources are sparse and for quick experiments, using API based finetuning might be more useful.

In-context learning

An interesting way of utilizing LLMs post-training is with In-context learning (ICL) approach. Without any gradient update, model learns to address a new task during inference by receiving a prompt, including task examples. In-context learning was popularized in the original GPT-3 paper as a way to use language models to learn tasks given only a few examples (Xie et al., 2022). In-context learning is very useful if we dont have direct access to the model, for instance, if we are using the model through an API.

Prompt Engineering

A prompt is a set of instructions provided for a Large Language Model (LLM), contributing to the customization and refinement of its capabilities. This

[4]https://deci.ai/blog/fine-tuning-peft-prompt-engineering-and-rag-which-one-is-right-for-you/

set of instructions can shape the LLM's responses and outputs in subsequent interactions. By establishing specific rules and guidelines for the initial phase of an LLM conversation, a prompt plays a crucial role in defining the context, signaling the importance of certain information, and specifying the desired format and content for the generated output. In particular, a prompt sets the context for the conversation and tells the LLM what information is important and what the desired output form and content should be. White et al. (2023)

Prompt engineering is the means by which LLMs are programmed via prompts. It refers to the process of designing and refining prompts or instructions given to a language model. Weng, 2023 describes prompt Engineering, also known as In-Context Prompting, refers to methods for how to communicate with LLM to steer its behavior for desired outcomes without updating the model weights.

Prompt patterns are essential to effective prompt engineering and the impact of its methods can exhibit significant variation across different models. This requires extensive experimentation and reliance on heuristics. Additionally, it falls within the domain of the prompt engineer to comprehend how to optimize outcomes for various generative AI models available in the market. In some cases (like OpenAI's GPT-4) no more training or fine tuning is allowed. In in these cases prompt engineering can prove to be very useful technique to extract more accurate and precise information. Formulating prompt for different LLMs requires different format and just because one prompt resulted better result in one model doesn't guarantee better result on another LLM.

Retrieval augmented generation (RAG)

Large language models (LLMs) have made remarkable success in natural language processing. Despite these advancements, LLMs face limitations, particularly in handling specialized queries and generating accurate information, termed as "hallucinations". Meta AI researchers have introduced a solution called Retrieval Augmented Generation (RAG) [5] to address these challenges. RAG integrates an information retrieval component with a text generator model, allowing for efficient fine-tuning and modification of internal knowledge without retraining the entire model. This approach enhances the model's capability to provide precise and relevant responses by incorporating external data retrieval into the generative process. RAG emerges as a promising method to overcome LLM limitations, making it more practical for real-world applications. (Gao et al., 2024)

Prompt tuning

Prompt-tuning is an emerging strategy to adapt large language models (LLM) to downstream tasks by optimizing prompt parameters from data (Rawat et al., 2023).

[5] https://ai.meta.com/blog/retrieval-augmented-generation-streamlining-the-creation-of-intelligent-natural-language-proces

Prompt-tuning allows for easy adaption of large language models (LLMs) to new tasks by training a small number of prompt parameters. It involves fine-tuning the prompt parameters to maximize the performance of the LLM on the task. Despite its effectiveness, one of the major drawback of prompt tuning lies in its lack of interpret ability as it is very hard to understand the reasoning behind optimal prompt. Notably, research suggests that prompt tuning can outperform fine-tuning across various scenarios, including full-data settings, data scarcity settings, and cross-domain settings.

Prompt tuning starts by feeding the model with a baseline prompt that captures the essence of the task. Then prompt parameters (Example: Context Window, Max Tokens, Temperature, etc) are optimized with the help of Bayesian optimization or gradient-based optimization optimization or other optimization algorithms. Using this technique in an iterative manner, we can maximize the performance of a LLM. Prompt-tuning allows a company with limited data to tailor a massive model to a narrow task. It also eliminates the need to update the models billions (or trillions) of weights, or parameters. Some researchers had also conclude that that prompt tuning can outperform finetuning under full-data settings, data scarcity settings, and cross domain settings. (Wang et al., 2022)

Conventional Finetuning

In-context learning is a valuable and user-friendly method for situations where direct access to the large language model (LLM) is limited, such as when interacting with the LLM through an API or user interface.

However, if we have access to the LLM, adapting and finetuning it on a target task using data from a target domain usually leads to superior results. It provides greater control and flexibility but has huge resource requirements. There are three conventional approaches to finetuning outlined in the figure below.

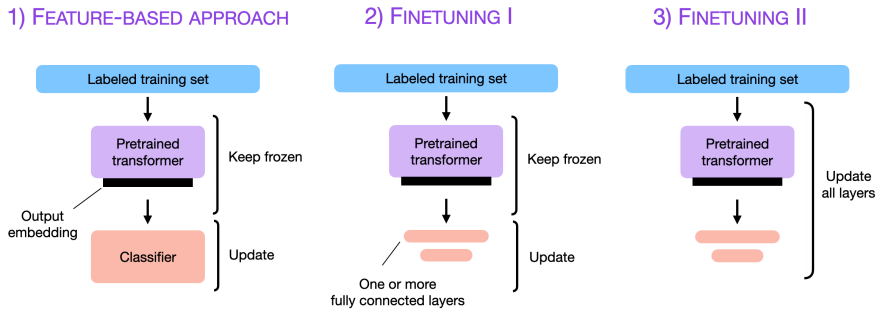


Figure 2.6: The 3 conventional feature-based and finetuning approaches. (Source: Raschka, 2023)

1. **Feature based approach:** If we have a separate classifier on top of our transformer model (For example, BERT+CRF architecture), we can use

feature based fine tuning. It involves freezing the body of transformer and only task-specific layers or additional classification layers attached to the pre-trained model.

2. **Fine-tuning I:** In this approach, just the output layers are updated. Similar to the feature-based approach, the parameters of the pretrained LLM are kept frozen. Freezing layers in a neural network model is often done from bottom to top because the lower layers of the model tend to learn more general and abstract features that are applicable across different tasks, while the higher layers learn more task-specific features. By freezing the lower layers and fine-tuning the higher layers, the pre-trained model aims to downstream NLP task while preserving the more general features learned during pre-training.

3. **Fine-tuning II:** In this approach, none of the layers are of pretrained model are frozen. Updating all the parameters optimizes performance but has much higher computational cost.

.

Supervised Fine-Tuning (SFT)

Supervised fine-tuning (SFT) in the context of large language models (LLMs) involves taking a pre-trained model and further training it on a specific task with labeled data. This process helps the model specialize in a particular domain or task by adjusting its parameters based on the provided labeled examples. In SFT, all of the model parameters are updated to produce outputs that are adapted to the task.

During the fine-tuning process, the model adjusts its parameters to better capture the patterns and features relevant to given task based on the provided labeled examples. This fine-tuning enables the model to specialize in a given task while retaining the general language understanding capabilities learned during pre-training.

Supervised fine-tuning (SFT) is often preferred over conventional fine-tuning as it preserves pre-trained knowledge and allows for faster task adaptation. This means reduced computational cost and data requirements which are a big factor when finetuning a LLM.

Parameter-Efficient Fine-Tuning (PEFT)

Parameter-Efficient Fine-Tuning (PEFT) is a technique used to adapt pre-trained models to specific tasks or domains while minimizing the number of parameters that need to be updated or fine-tuned. PEFT involves only fune tuning a small number of model parameters which decreases the computational and storage

costs. The number of trainable parameters is significantly reduced in PEFT (Reduced to less than 1%). (NVIDIA, 2024). This approach is particularly valuable when computational resources are limited or when fine-tuning a large number of parameters is impractical.

SFT most often leads best possible results but PEFT also can also achieve nearly the same degree of accuracy with significantly lower computational cost. As LLMSs continue to grow in size, PEFT is gaining popularity due to its lightweight requirements on training hardware.

If resources are sparse and for quick experiments, using API based finetuning or directly using a commercial API might be more useful.

Reinforcement Learning from Human Feedback (RLHF)

Reinforcement learning from human feedback (RLHF) is a machine learning approach that combines reinforcement learning techniques, such as rewards and comparisons, with human guidance to train an artificial intelligence (AI) agent.

There are times when human feedback is vital to fine-tune an interactive or generative AI, such as a chatbot. Using human feedback for generated text can better optimize the model and make it more efficient, logical and helpful. In RLHF, human testers and users provide direct feedback to optimize the language model more accurately than self-training alone.

RLHF is a challenging concept because it can be involves a multiple-model training process and different stages of deployment. (Nathan Lambert, 2022) The process unfolds through multiple stages, initiating with the utilization of a pretrained LM, such as GPT-3 or GPT-4. This initial model can also be fine-tuned on additional text or conditions, but does not necessarily need to be. Following this, a reward model (RM) is crafted to capture human preferences, employing human annotators to rank generated text outputs and create a standardized dataset. In the fine-tuning process, the task is framed as a reinforcement learning (RL) problem. The policy, represented by a language model, takes a prompt and generates text, with the action space covering the model's vocabulary. The reward function combines the preference model with a constraint addressing policy shifts in this RL-based approach. Optionally, the RLHF process permits iterative updates, with users ranking outputs against earlier LM versions.

Chapter 3

Implementation

In this chapter, we will discuss how specific models are developed and trained. We will also talk briefly about the datasets used, pre-processing and training process. Evaluation and metrics used for comparision are also discussed in this chapter. We will be using libraries xmltodict, pandas and numpy for reading and processing the XML data. Then scikit learn, keras, pytorch and transformers were used to develop and train models. This chapter provides a comprehensive overview of our methodology and serves as a foundation comparisons and discussions in chapter 4.

3.1 Datasets

The evaluation of the proposed model is conducted on the datasets of SemEval-2014 Tak 4. SemEval-2014 Task 4 is a widely used benchmark dataset for ABSA. Its consists four different subtasks: Aspect term extraction, Aspect term polarity, Aspect category detection and Aspect category polarity.

Aspect Term Extraction is the identification and extraction of specific aspects or features within a given text that are important for sentiment analysis. The next subtask, Aspect Term Polarity determines the sentiment polarity associated with each extracted aspect term, classifying sentiments as positive, negative, or neutral. Then going to a much broader level, Aspect Category Detection involves identifying broader categories to which aspect terms belong. It helps us identify topics of interest present within the text. Finally, Aspect Category Polarity tries to identify sentiment polarity associated with identified aspect categories, providing a overall sentiment towards specific topic in the text.

All the subtasks share two domain datasets, which are a restaurant dataset and a laptop dataset. We used Restaurant dataset for both training and testing. In the restaurant dataset, there are 5 different aspects: food, service, price, ambience, anecdotes/miscellaneous. Most of the reviews have single aspect or no aspects at

Set	Total	Aspects			Polarity			
	Sentence	None	Single	Multi	Positive	Negative	Neutral	Conflict
Train	3041	1020	1666	355	1303 (2164)	549 (805)	464 (633)	84 (91)
Test	800	194	521	85	411 (728)	142 (196)	128 (196)	14 (14)

Note: In columns related to polarity, the values in brackets represent the number of aspects. Other values represent the number of sentences.

Table 3.1: Number of sentences and aspects in each class in SemEval-14 restaurant Dataset

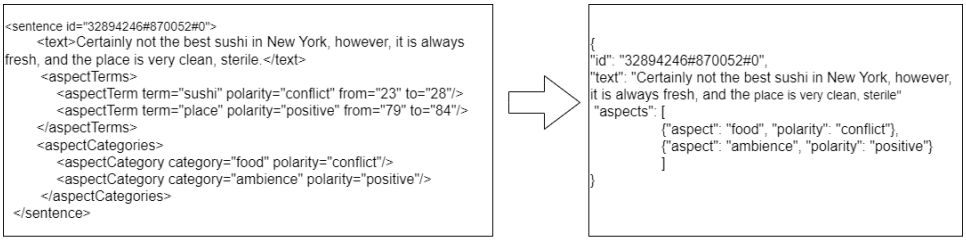
only and only few contain multi aspect. There are total of 3041 sentences in the training set and 800 sentences in the test set. There were a lot of reviews without any sentiment aspects in semEval dataset Restaurant set (1020 sentences in train set and 194 in test set). The remaining 2021 (3041 - 1020) sentences with 3693 in train set and 606 (800 - 194) sentences with 1168 in test set. There are more aspects than sentences as some reviews also have multiple aspect terms. SemEval annotation guide also defines a "conflict" class which is supposed to be used when an aspect has both positive and negative sentiment. There were 91 different aspects in 84 sentences which were marked as conflict in train set and 14 aspects in 14 sentences in test set. A brief summary is presented on Table 3.1.

```
<sentence id="270">
  <text>From the incredible food, to the warm atmosphere, to the friendly
  service, this downtown neighborhood spot doesn't miss a beat.</text>
  <aspectTerms>
    <aspectTerm term="food" polarity="positive" from="20" to="24"/>
    <aspectTerm term="atmosphere" polarity="positive" from="38" to="48"/>
    <aspectTerm term="service" polarity="positive" from="66" to="73"/>
  </aspectTerms>
  <aspectCategories>
    <aspectCategory category="food" polarity="positive"/>
    <aspectCategory category="service" polarity="positive"/>
    <aspectCategory category="ambience" polarity="positive"/>
  </aspectCategories>
</sentence>
```

An example review from the restaurant training set can be seen above. This example contains one single aspect which is the case for majority of reviews in this dataset. Restaurant version of the dataset has aspect term and aspect category but only category is used in this work. Only the term aspect also refers to aspect category in this literature.

3.2 Data Preprocessing

The original data was available in XML format. It is hard to use XML format. The data was first converted into a dictionary with the help of xmltodict[1] library. Then finally converted to pandas dataframe as it makes data processing more easier and streamlined. Figure 3.1 shows sample amount of data at several stages of conversion before any further preprocessing is done.



	text	category	polarity
0	The bread is top notch as well.	[food]	[positive]
1	I have to say they have one of the fastest del...	[service]	[positive]
2	Food is always fresh and hot- ready to eat!	[food]	[positive]
3	Did I mention that the coffee is OUTSTANDING?	[food]	[positive]
4	Certainly not the best sushi in New York, howe...	[food, ambience]	[conflict, positive]

Figure 3.1: Converting XML data to dataframe

After that, the text is converted to lowercase in order to provide consistency in character representation and newline characters are replaced with spaces. The text is then tokenized into individual words using nltk tokenizer[2] and only alphabetic tokens are retained, thereby discarding non-alphabetic characters such as punctuation and numbers. Moreover, common English stopwords that contribute minimal semantic value, are removed from the tokenized list.

Since conflict refers to cases where even human reviewer found it hard to classify, and also looking into the figure 3.2, there is just 5% data belonging to conflict class. So if the review has conflict polarity, the entire review is removed. Furthermore anecdotes/miscellaneous class is renamed to other for easier readability. Additionally, a joint label column is introduced, combining category and polarity information. This allows us to create a classifier for joint

[1]https://pypi.org/project/xmltodict/
[2]https://www.nltk.org/api/nltk.tokenize.html

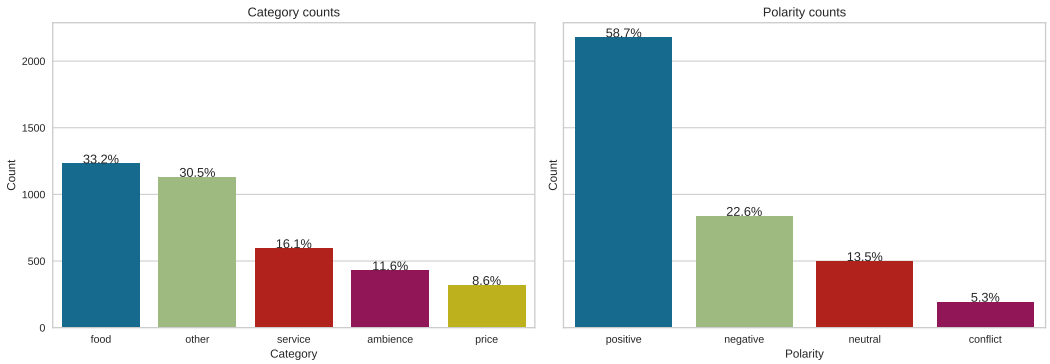


Figure 3.2: Aspect category and Sentiments polarity counts

classification as well. To use the multilabels with developed models, we need to convert the text label into numerical encodings. The conversion was done with the scikit-learn MultiLabelBinarizer[3]. An example of data after all the processing can be seen on Figure 3.3.

	text	category	polarity	joint	category_labels	polarity_labels	joint_labels
0	staff horrible us	[service]	[negative]	[service#negative]	[0, 0, 0, 0, 1]	[1, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
1	completely fair redeeming factor food average ...	[food, other]	[positive, negative]	[food#positive, other#negative]	[0, 1, 1, 0, 0]	[1, 0, 1]	[0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
2	food uniformly exceptional capable kitchen pro...	[food]	[positive]	[food#positive]	[0, 1, 0, 0, 0]	[0, 0, 1]	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
3	gabriela personaly greets recommends eat	[service]	[positive]	[service#positive]	[0, 0, 0, 0, 1]	[0, 0, 1]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
4	go enjoy say get	[other]	[positive]	[other#positive]	[0, 0, 1, 0, 0]	[0, 0, 1]	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]

Figure 3.3: Sample data with after MultiLabelBinarizer

Then, A new column with labels formatted as dictionary was also added. This was done to make it easier to perform evaluation on Large Language Model. Ultimately we converted our data to Huggingface datasets[4] format which is more optimized for speed and efficiency while using transformer models. One example of the format is shown below.

```
{
  'text': 'bread top notch well',
  'category': ['food'],
  'polarity': ['positive'],
  'joint': ['food#positive'],
  'category_labels': [0, 1, 0, 0, 0],
  'polarity_labels': [0, 0, 1],
  'joint_labels': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
```

[3]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html

[4]https://huggingface.co/docs/datasets/index

```
  'true_labels': {
       'ambience': None,
       'food': 'positive',
       'other': None,
       'price': None,
       'service': None
       }
}
```

3.3 Task Description

The task of aspect detection and aspect polarity classification can be treated as a multi-class multi-label classification problem. ABSA can be performed as a separate classification tasks (Aspect classification + Sentiment classification) or as joint classification task. We experiment with both approaches for our simpler models and experimented with only joint classification for complex models.

3.3.1 Separate tasks

The simplest approach is to train two models, one for aspect classification and other for sentiment classification.

The first model is dedicated to aspect classification, where the goal is to identify and categorize aspects mentioned in the text. For example, given a product review, this model aims to identify aspects such as "food", "service", "price", "ambiance" and "others". The second model is then trained for sentiment classification. The model simply performs a multilabel classification with three classes, positive, negative, or neutral.

This separate tasks approach allows for a modular and specialized treatment of aspect and sentiment prediction. Each model can be trained independently, potentially using different architectures or hyperparameters that performs best for the specific task.

3.3.2 Joint task

In the joint task approach, Aspect-Based Sentiment Analysis (ABSA) is treated as a unified problem and a single model is employed to predict both aspects and their associated sentiments simultaneously. The model uses the joint label that has the information of both aspect category and sentiment. The model is also trained for multi label multi class classification. This approach might lead to poorer performance as there are a lot more classes and there are less reviews per class.

But it might be beneficial in cases where the interactions between aspects and sentiments is important.

Another possible approach for performing the task jointly is to treat is like a NER problem. This is more useful in cases of aspect term sentiment analysis and requires some further data processing as each token in the input sequence needs a label indicating whether it is part of an entity and if so the type of entity (An example would be BIO labeling scheme where 'B' is used to represent beginning of aspect, 'I' is used to represent inside and end on a multi-word aspect and 'O' represents everything else that is not an aspect term). Transformer models usually perform quite well for these sort of token classification task but as we are more focused on aspect category we will not be exploring it in this work.

3.3.3 Generative predictions

With recent LLMs, predictions can be made with generative responses. One of the major problems with using generative responses for prediction is hallucinations. As LLMs tend to hallucinate and add things that are not present in the original text. This would mean we might end up with cases where the predicted aspects are different from the aspects we want. It also requires some post processing to extract predictions from the generated text. Prompt engineering and prompt tuning are gaining more popularity to further improve and structure generative texts.

As part of this work, we have also experimented with LLMs. We tried to predict both aspects and polarity at once with LLMs. Some post processing was essential to convert generated text to proper format (dict) for evaluation.

3.4 Model Selection

We wanted to observe the effectiveness of using simple ML models for the ABSA but also experiment with more complex and more resource intensive models. We start with simple SVM based models and eventually experiment with latest large language models-

3.4.1 Support Vector Machine (SVM)

Support Vector Machine is generally used for both classification and regression tasks and it works by finding a hyperplane in a high-dimensional space that best separates data points into different classes. Linear Support Vector Classification uses a linear kernel function, making it suitable for linearly separable data. Linear SVMs are computationally efficient and seeks to find a hyperplane that separates classes in a linear way. The implementation was done with scikit-learn library.

As SVC is a binary classification algorithm it doesn't natively support multi classification.

One approach for using binary classification algorithms for multi class or multi label problems is to split task into multiple binary classification and fit a model for each. Two different approaches are One vs Rest and One vs One strategies. One vs Rest strategy splits the classification task into one binary classification problem per class and One-vs-One strategy splits the classification task into one binary classification problem per each pair of classes. As part of this work, One vs Rest approach is used.

TF-IDF vectorizer

To convert the raw text into vectors TF-TDF vectorizer was used for SVM model. Term Frequency-Inverse Document Frequency (TF-IDF) Vectorizer takes into account the importance of words in a document relative to their frequency across multiple documents in a corpus. It assigns weights to words based on two main factors: term frequency (TF), which measures how often a word appears in a document, and inverse document frequency (IDF), which assesses the rarity of a word across the entire corpus. By combining these two metrics, the TF-IDF Vectorizer is able to highlight words that are both frequently occurring in a document and relatively unique to that document. This means if a word appears frequently in the entire dataset, it is given low importance but if it appears frequently in a particular document but is rare in dataset, it is given high importance. After passing the text to vectorizer, we end up with a sparse vector document that represents the text.

3.4.2 Gradient Boosting

Gradient Boosting is an ensemble learning method that combines the predictions of multiple weak learners (typically decision trees) to create a strong learner. Gradient boosting was also explored as it often performs well in practice and can capture complex relationships in the data. It is less prone to overfitting and can handle missing data. We had to use One vs Rest strategy for Gradient Boosting as well. With Gradient boosting and One vs Rest, each weak learner is dedicated to distinguishing one specific class from the rest of the classes. During training, the gradient boosting algorithm optimizes the ensemble of weak learners to collectively address the classification problem. This simplifies the extension of gradient boosting to handle multiple classes by decomposing the problem into a set of binary classification sub-problems. We used the same TF-IDF vectorizer as SVM for gradient boosting as well.

3.4.3 Long Short-Term Memory(LSTM)

The LSTM architecture is a recurrent neural network (RNN) architecture that captures and remembers long-term dependencies in sequential input. Unlike traditional RNNs, LSTMs are equipped with memory cells and a sophisticated gating mechanism, allowing them to selectively store, read, and erase information over extended sequences. This allows the models to better understand the context and this in theory would improve accuracy scores in text based tasks as contextual information has a great significance.

Tokenizer provided as part of keras library was used for tokenization. Keras tokenizer by default removes all punctuation. Then the text is converted to space-separated sequences of words which are then split into lists of tokens. There were only 6 reviews which had longer than 180 characters, so we decided to set the max length of tokenizer to 180 and max words to 5000. The tokenizer pads or truncated the review where necessary.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 180, 20)           100000

 dropout_1 (Dropout)         (None, 180, 20)           0

 lstm_1 (LSTM)               (None, 180, 300)          385200

 global_max_pooling1d_1 (Gl  (None, 300)               0
 obalMaxPooling1D)

 dense_1 (Dense)             (None, 5)                 1505

 activation_1 (Activation)   (None, 5)                 0

=================================================================
Total params: 486705 (1.86 MB)
Trainable params: 486705 (1.86 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3.4: Summary of LSTM based model used

The model was implemented as a Sequential model in Keras. Sequential lets us group a linear stack of layers. The model has a max pooling layer followed by a dense layer. There is also a dropout layer in the model which makes the model less prone to overfitting. A summary of model structure can be seen in Figure 3.4. There are a total of 486705 trainable parameters in the model. We will be training the model for 20 epochs with Early stopping and Reduce learning rate on plateau callbacks. We used Adam as our optimizer with binary crossentropy as loss metric. The training pipeline also saves the best model which can be loaded

later on to make sure evaluations are done on the best performing instance.

3.4.4 Bidirectional Encoder Representations from Transformers(BERT)

BERT is an advanced natural language processing (NLP) model developed by Google and rooted in the transformer-based architecture. Historically, language models could only read text input sequentially either left-to-right or right-to-left but couldn't do both at the same time. BERT is different because it is designed to read in both directions at once.

For this work, "bert-base-uncased" was used as our base model with a Linear classifier on top. Transformer natively supports multi-label multi-class classification meaning it didn't require any additional processing for the task. With the help of huggingface trainer, training bert model was very easy and straightforward. We mostly used the default values of huggingface library for most of the hyperparameters. The model was trained for 20 epochs with learning rate set to 1e-5. The best epoch was evaluated with micro f1 score and best one was loaded at the end for evaluation. We used a batch size of 8 and AdamW as the optimizer. Bert was also used for tokenization and padding of sequences.

BERT Tokenizer

To tokenize and vectorize the text, we utilized the pre-trained BertTokenizer from the Transformers library. BERT tokenizer is based on the BERT model and generates tokens and maps them to corresponding IDs in the BERT vocabulary. Special tokens are added to the beginning ([CLS]) and end of each input sequence ([SEP]) to indicate the sequence's start and end. Special padding token ([PAD]) is also added is any padding is required. BERT tokenizer outputs input_ids, token_type_ids, attention_mask, and label_ids and can be used with any transformer models.

Figure 3.5: Traditional tokenization approach compared with wordpiece tokenization (Source: Google Research[5])

BERT employs a unique tokenization method called wordpiece tokenization which breaks words into subwords to handle words not in the tokenizer's vocabulary (Song, 2021). This enhances models generalization capabilities. Unlike traditional tokenizers that mark out-of-vocabulary words as "unknown" wordpiece tokenization's subword approach increases the likelihood of subword components being present in the vocabulary. We used "longest" padding strategy which pads the sequences in a batch to be same length as the longest sequence. Figure 3.5 shows how a really long word that is not split into tokens with traditional white space tokenization would be tokenized with wordpiece tokenization.

3.4.5 Sentence transformers (sbert)

Sentence transformer has several pre-trained models which can be further trained for any downstream tasks. For this work, "all-MiniLM-L6-v2"[6] and "all-mpnet- base-v2"[7] were used as aspect filering and polarity classification models respectively. They both are all-round models fine tuned for many use-cases and trained on a large and diverse dataset of over 1 billion training pairs. all-MiniLM-L6-v2 is based on nreimers/MiniLM-L6-H384-uncased[8] which is a 6 layer version of microsoft/MiniLM-L12 -H384-uncased created by keeping only every second layer. Then all-mpnet-base-v2 is based on microsoft/mpnet-base[9].

To use these sentence transformer models for our experiments, we used setfit library. SetFit is an efficient and prompt-free framework for few-shot fine-tuning of Sentence Transformers models (Tunstall et al., 2022). It also has a AbsaModel class as well which can be instantiated for ABSA training and prediction. We instantiated it with AbsaModel.from_pretrained() with following parameters:

- Sentence Transformer model to be used for the aspect filtering is passed as the first argument. **(sentence-transformers/all-MiniLM-L6-v2)**
- Sentence Transformer model to be used for the polarity classification is the second argument. If not provided, the same Sentence Transformer model as the aspect filtering model is also used for the polarity classification model. **(sentence-transformers/all-mpnet-base-v2)**
- Spacy model to be used is passed as third argument. Small spacy model **(en_core_web_sm)** was used to keep training times lower.

Similar to Transformers, setfit also has a built in trainer which makes training and evaluation very easy. Setfit specifically also has AbsaTrainer. We trained the

[5]https://blog.research.google/2021/12/a-fast-wordpiece-tokenization-system.html
[6]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[7]https://huggingface.co/sentence-transformers/all-mpnet-base-v2
[8]https://huggingface.co/nreimers/MiniLM-L6-H384-uncased
[9]https://huggingface.co/microsoft/mpnet-base

models for 5 epochs with batch size of 128. To train our sentence transformer models with setfit, we also had to prepare training and testing set according to their requirement. The dataset must have "text" (full sentence or text), "span" (aspect), "label" (sentiment polarity), and "ordinal" (index of occurrence in cases of multi occurrence).

3.4.6 Mistral and Phi-3

Mistral and Phi-3 both are recent Language Models from Mistral AI and Microsoft respectively. As part of this work we used “Mistral-7B-Instruct-v0.2” and “Phi-3-mini-128k-instruct”. They both are instruction fine tuning which means these models are more aligned to follow instruction provided in form of prompt.

As both these models ware hosted on huggingface, we used huggingface transformers to experiment with these models. Transformers has AutoModelForCausalLM which can be used with from_pretrained() method to instantiated with either phi3 or mistral model. These both models require large amount of data and resources for training, so we decided to not perform any further training with these model. Rather than training, we have experimented with using these models with zero shot and few shot approach. By utilizing huggingface pipeline, inference with these models was very easy.

Large language models often exceed the capacity of consumer GPUs. If more resources are available, they can be loaded into multiple GPUs or used through cloud providers. One effective strategy to manage this is quantization, which reduces the precision of the model’s weights and activation from higher-precision formats, such as 32-bit floating-point, to lower-precision formats, like 8-bit integers. This technique optimizes the model by significantly decreasing its memory footprint, improving inference speed, and reducing energy consumption, making it more efficient for deployment on devices with limited resources.

Mistral models have a quantized version which use AWQ quantization technique. AWQ (Activation-aware Weight Quantization) is a quantization method that aims to reduce the memory requirements of large language models while preserving or even improving their performance.

Model Name	Model Size	Released	By
Llama 2	7B	July, 2023	Meta
Mistral	7B	September, 2023	Mistral AI
Mixtral	8X7B	December, 2023	Mistral AI
Phi-2	2.7B	December, 2023	Microsoft
Phi-3 mini	3.8B	May, 2024	Microsoft
GPT3	175B	May, 2020	OpenAI
PaLM2	540B	May, 2023	Google

Table 3.2: Different Large Language Models

There are a lot more models being released everyday. Some of them are openly and freely available on Huggingface while some are closed behind a paywall. Some models like Gemini-1.5 and GPT-4 do not even have parameter size as public information. Table 3.2 shows some of the recent models along with parameter size and release Date.

3.5 Model Training

Due to the different in complexity of our models, we couldn't use same approach to training all the models. Different model training was also implemented using different libraries.

- For SVM and Gradient Boosting we simply used scikit-learn for training.
- For LSTM a training loop implemented in Pytorch was used.
- Huggingface trainer was used for BERT model.
- Sentence transformer has a built in ABSA trainer which we used to train our sbert models.

We couldn't train the large language models as they require much more data and computing power. But we used them without training which is described further in next section.

3.6 Inference with LLMs

Training large language models (LLMs) on consumer hardware is not feasible due to limitations related to computational resources and infrastructure. Modern

LLMs, with billions to trillions of parameters, demand immense computational resources. This means we have to rely on techniques such as finetuning, n-shot classification, prompt engineering and API based access to use LLMs.

```
messages = [
    {"role": "system", "content": "System prompt here"},
    {"role": "user", "content": "Prompt 1"},
    {"role": "assistant", "content": "Answer to Prompt 1"},
    {"role": "user", "content": "Prompt 2"},
    # Output to prompt 2 is generated by the model
    ]

pipe = pipeline("text-generation",
                model=model,
                tokenizer=tokenizer)
generation_args = {"max_new_tokens": 500,
                   "return_full_text": False,
                   "temperature": 0.0,
                   "do_sample": False}

output = pipe(messages, **generation_args)
generated_text = output[0]['generated_text']
```

Listing 3.1: Sample code for using huggingface pipeline

In this work, we have used Huggingface pipeline as our main library to utilzie LLMs. As our dataset was already in huggingface datasets format, it was simply a matter of defining model and tokenizer and setting huggingface pipeline to "text-generation". The message follows the template shown in the listing above.

3.6.1 N-shot classification

N shot classification involves providing only N samples of data to the model before performing the task. In context of an Large Language Model, N samples are passed as part of the prompt. We used *<User>* -> *<Assistant>* -> *<User>* -> *<Assistant>* order to create chain of messages with sample reviews and responses.

We used the same approach to test both zero shot and few shot approaches for both models. To experiment with zero shot approaches, the input message had to be formatted with only one prompt with no example outputs.

```
zero_shot = [{"role": "system", "content": "System prompt here"},
             {"role": "user", "content": "Prompt 1"}]

# Message for N shot classification for task with M classes
n_shot    = [{"role": "system", "content": "System prompt here"},
             {"role": "user", "content": "Example 1 of class 1"},
```

```
{"role": "assistant", "content": "class 1"}
...
{"role": "user", "content": "Example N of class M"},
{"role": "assistant", "content": "class M"},
{"role": "user", "content": "Unseen Text"}]
```

Then for N-shot we needed to add one example of each class to the message prompt. As we wanted output to be formatted as a dictionary, with aspect as key and polarity as value, one of the major improvement from N shot was reliable and consistent output format.

3.6.2 Prompt engineering

Proving instruction to a LLM is a very hit and miss process as same prompt doesn't work for multiple tasks. Prompt engineering is simply the process of trying to find best prompt for the given task. We started with a very simple system prompt *"Perform Aspect based sentiment analysis where aspects are ['food, service, ambience, price, other'] and sentiments are ['positive, neutral, negative']".* but this was not enough. We were getting outputs with long text explaining the answer and thought process.

After experimenting with few prompt styles, we found that asking the model to be a persona and then asking it to perform task at hand worked the best. We were only able to perform prompt engineering with phi-3 model as it was the only model that consistently fitted into our GPU. Some of the prompts we experimented can be seen below.

```
"You are a skilled data analyst. Your task is to extract and analyze
sentiments from user reviews and structure the information into a JSON
format. Analyze the provided user reviews and create a JSON summary.
Focus on these key aspects: {aspects} and sentiments: {sentiments}."

...

"Your task is to analyse and perform aspect based sentiment analysis
on the given restaurant reviews text. The aspects should only be
{aspects} and sentiments should only be {sentiments}. Output should
only contain mentioned aspects and their respective sentiments as a
dictionary. Each aspects should only have one sentiment and not every
aspect is necessarily present. Do not provide any further
explanation."
```

Listing 3.2: Prompts used as part of experiment

All the prompts used as part of our experiment can be seen in Appendix A.

Prompt formats

Different LLMs require different prompt formats. Most of them unified prompt templates for inference as we have described before in some cases it needed to be formatted slightly differently. In case of Mistral model, even though we were using huggingface pipeline for inference, prompt had to be formatted as *"<s>[INST] system_prompt [/INST]</s>[INST]review_text[/INST]"* and system prompt included a sample review and response text.

3.6.3 Finetuning

To use an LLM for custom task, we can use techniques such as Retrieval Augumented Generation (RAG) or Finetuning. These approaches can help ground the model better and ultimately reduce hallucinations. Depending on the task one approach migght be better than the other. Singh, 2024 highlights the generic differences between two approaches. As shown in Figure 3.6, RAG approach is much more suited if there is a knowledge base that needs to be queried for responses. For our usecase, finetuning is much better as we have a custom dataset that we can use for training. Finetuning has lower data and computational

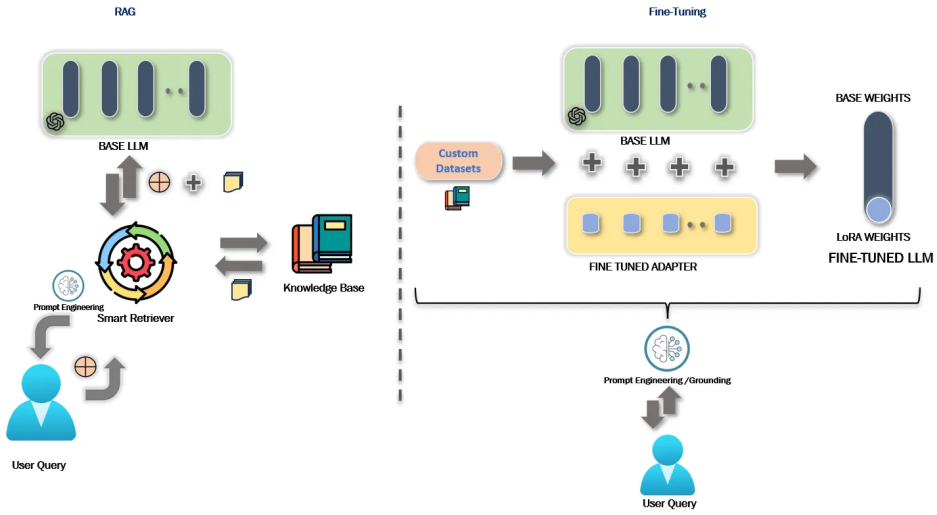


Figure 3.6: Difference between RAG and Finetuning pipeline

requirements compared to full training, but the model still must be able to fit into GPU memory. Currently there are several approaches to finetuning LLMs, one of the easiest approach is to use API based fine tuning. OpenAI allows finetuning of their models through their API. We finetuned a GPT-3.5-Turbo model as it was the fastest and cost effective option[10].

[10]https://platform.openai.com/docs/guides/fine-tuning

Before finetuning, the data had to be prepared. The data needed to converted to jsonl file with following format:

```
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
```

Listing 3.3: OpenAI conversational chat format

OpenAI also has a handy guide showing how to estimate the cost of finetuning by counting Tokens. When we performed the counting on full train dataset, it came out to be around 3M tokens which was more than our expectations. We decided to sample the dataset to only 100 examples to get an idea if finetuning would be beneficial. We trained for 3 epochs with batch size of 1 and used 35,178 training tokens. Our final train loss was at 0.3575 however at step 200 it was at 0.000. As openai api allows us to use checkpointed models as well, we will be experimenting with both final model and model at step 200.

Finetuning with openAI simply involves uploading file to their servers and then creating a finetuning object. Then the model is automatically finetuned and stored by OpenAI services. It took about 30 minutes to finetune the model.

To use the finetuned models, we only needed to specify the model name and it was a similar pipeline to using normal chat completion API.

```
response = client.chat.completions.create(
model="ft:gpt-3.5-turbo-0125:MODEL_ID",
response_format={ "type": "json_object" },
messages=[
    {"role": "system", "content": PROMPT},
    {"role": "user", "content": REVIEW TEXT},
    {"role": "assistant", "content": REVIEW RESPONSE},
    {"role": "user", "content": QUERY REVIEW TEXT}
    ]
)

output = response.choices[0].message.content
```

Listing 3.4: OpenAI pipeline for using finetuned model

As we had specified the response format as json_object, there were no problems regarding the formatting of responses. It did however provide an empty json in cases where the model was not able to generate proper response.

3.7 Evaluation

The performance evaluation of our models is crucial to understand their effectiveness in ABSA and to compare our research with past papers. There are several different metrics that could be used for comparison. One of the most used

metric is f1 score. For multi-label classification f1 score could be either macro or micro averaged. Macro f1 score treats all classes equally and is useful when class distribution is balanced. But in our case as there is class imbalance, micro F1 Score is more useful and it aggregates the contributions of all classes. Other than f1 score, accuracy score is also used as a evaluation metric. Three different types of accuracy scores are used as part of this work:

1. Binary : It is the simplest way to calculate accuracy and this is calculated by treating both target and predicted label sequences as simple binary problem

2. Exact Match : The set of labels predicted for a sample must exactly match the corresponding set of labels in target. Also known as subset accuracy. This measure shows how many of the predicted labels were fully correct.

3. Overlap : The set of labels predicted for a sample must overlap with the corresponding set of labels in target. This score not only considers full correct but also partially correct predictions.

For comparision, f1 score is used when available in the relevant research work. Mainly torcheval[11] and scikit-learn metrics[12] library were used for the implementations of all these metrics. In cases of LLMs, they are also evaluated by the error rates in output format.

For Large language models, we decided to evaluate them further on text generation and hallucinations. We decided to only perform minor post processing to the output, so we can test the models easiness to be integrated into existing systems. Basic pseudo code of post processing is shown below.

```
try:
    _dict = eval(output)
    if not isinstance(_dict, dict):
        _dict = {}
except:
    _dict = {}
```

Listing 3.5: Post processing of LLM output

To properly evaluate the LLM responses, we also added following metrics to our evaluation criteria.

1. Partial Accuracy: Output was generated in desired format and has at least one correct aspect and sentiment

2. Count of Extra Aspects: Total number of new aspects that are introduced by the model

[11] https://pytorch.org/torcheval/stable/torcheval.metrics.html
[12] https://scikit-learn.org/stable/modules/model_evaluation.html

3. Number of Reviews with Extra Aspects: Number of reviews where at least one extra review was introduced

4. Incorrect response: Total amount of responses which were not generated in desired format

Since usage cost (resource requirement) is one of the most important factors while using LLMs, we will also be comparing the different models/approaches with respective cost associated with their usage. We have only made this comparison with LLMs as simpler ML models can usually be run on consumer hardware without any concerns.

3.8 Hardware

Initially we started our experiments on our local PC. It has Nvidia GTX 1650M, Intel i7 processor and 16GB RAM. We also had access to Google Colab and its TPU accelerators. We were able to run all our simpler ML Model experiments with these resources. When it came to LLM experiments, GPU memory on local PC was not enough and we also ran out of free compute units on Google cloab.

To perform our LLM experiments, we bought Google colab pro with 100 compute units. This gave us access to Google's L4 and T4 GPUs. We used L4 GPU with batch size of 100 for all of our LLM inference. And then for finetuning GPT 3.5 turbo, it was done through API. We didn't have to use our own resources but had to acquire API access through OpenAI.

Chapter 4

Results & Discussion

This chapter describes the different results obtained. We will also be comparing the results with other models. We used semeval 14 restaurant test set for making comparisons. Comparisons to existing research and other author's work is also made when relevant. We will discuss our results of various models for category, sentiment, and joint classification tasks. We had used SVM, gradient boosting, LSTM, SBERT, Phi3 for category and sentiment polarity task. For the category classification For , SVM model outperformed Gradient all the models including phi3. For Sentiment classification SBERT outperformed obtaining 73.23% of exact match accuracy. For joint classification we explored BERT as well. With 'bert-base-uncased' model for joint classification yielded promising results, with a micro F1 score of 0.7334, showcasing the potential of more advanced models in handling complex tasks. Then with LLMs, we found that minor finetuning of GPT outperforms open-source LLMs.

4.1 Category classification

The first model we implemented was a basic SVM model. We were able to obtain micro f1 score of 0.8189 without any further fine-tuning or experimentation. The model was only able to predict 68.49% of multi-labels exactly but the overlap accuracy score was higher at 82.64%.

Then we also implemented an ensemble learning model, we trained Gradient boosting model for category classification. It obtained a lower micro f1 score (0.7333) than SVM. Similarly the overlap accuracy score (64.35%) and exact match accuracy score (51.40%) are lower in comparison with SVM. SVM generally performs better in high-dimensional spaces and can handle complex decisions which might make them better for text classification tasks. But additional hyperparameter tuning might create better results.

We started by training LSTM model for 20 epochs without early stopping but

as seen on Figure 4.1(a) that the validation loss starts increasing after 6 epochs. So, we retrained the model with callbacks and also enabled save best model. We were able to obtain micro f1 score of 0.7412 which is comparable to our previous models. We obtained an overlap accuracy score of 76.36% which is lower than results obtained by Wang et al., 2016 using LSTM and TD-LSTM. They were able to achieve accuracy score of 82% with LSTM and 84% with TD-LSTM. Shu et al., 2019 used a CNN based model on semeval 16 restaurant dataset and obtained f1 score of 0.7564 which is similar to ours. We finetuned a sentence transformer model with SetFit (SBERT) achieving an accuracy of 59.73% for aspect classification and 73.23% for polarity classification. These results were lower than anticipated, which could be attributed to the limited dataset used for training the model.

To facilitate a comparison between our LLM experiments and simpler models, we also performed same evaluation techniques on LLMs. Extensive experiments were performed with phi3 including zero shot, n-shot and prompt engineering techniques. Mistral and GPT models were also explored to some extent.

With phi-3 we were not able to produce any significant predictions with zero shot approach. Our best result was obtained using a five-shot approach. It didn't achieve high exact accuracy but overlap accuracy was slightly better which is 69.56%. Although the accuracy was low this outcome was expected due to propensity of LLMs to hallucinate and introduce unexpected text, aspects or sentiments into the output. Further details on all the experiments are provided in the next section.

Model	**exact match accuracy**	**overlap accuracy**	**micro f1**
SVM	**68.49%**	**82.64%**	**0.8189**
Gradient Boosting	51.40%	64.35%	0.7333
LSTM	58.34%	76.36%	0.7412
SBERT	59.73%	-	-
Phi3 (5-shot)	54.61%	69.56%	0.7071

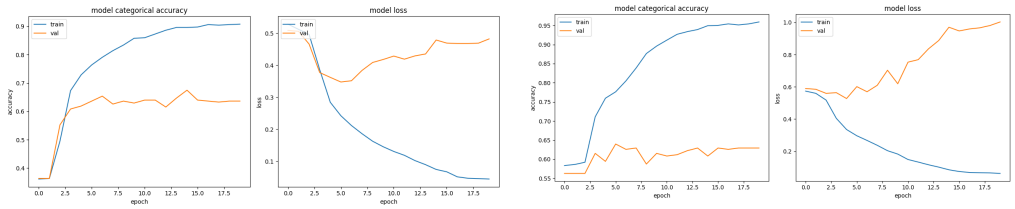
Table 4.1: Results of Category classification in different model

The results of different models that we implemented for category classification is summarized on table 4.1. Best value of each metric is highlighted in bold.

4.2 Sentiment classification

For the task of sentiment polarity classification as well, we started with a SVM model. The results for sentiment polarity was comparable to category classification results. It was able to obtain a micro f1 score of 0.751 and 73.83% overlap accuracy. The results were slightly lower than results obtained by Wagner et al., 2014 using

SVM and lexical rules. They were able to obtain an accuracy score of 85% with SVM and 77% using rule based approach. There have been other works which use SVM as well but a lot of them use some other datasets, like Onwuegbuche et al., 2019 obtained f1 score of 0.71 on five nigerian bank tweet dataset and Najkov, 2023 obtained f1 score of 0.84 on macedonian restaurant reviews. Authors who have used models of similar complexity to SVM, like Random Forest also have reported similar and comparable results on sentiment polarity classification tasks.



(a) Category Classification Task (b) Sentiment Polarity Task

Figure 4.1: Accuracy and loss plots of LSTM model for different tasks

Then we tried LSTM model which showed decent performance. Similar to category classification task, LSTM model loss starts to increase after about 6 epochs without any callbacks. Figure 4.1(b) shows the accuracy and loss values over 20 epochs. Then when retrained with callbacks, we obtained f1 score of 0.7129 and the model was able to get 64.48% of overlap accuracy. Then we experimented With sbert and were able to achieve the highest exact match accuracy so far at 73.23% for polarity classification.

Results from our experiments with LLMs suggest that sentiment classification is easier than category classification. The performance for category classification was suboptimal, while sentiment classification yielded better results. Specifically, the three-shot approach achieved the highest accuracy for sentiment classification, with Phi-3 reaching 63.81%. The overlap accuracy and micro F1 score for this approach were 69.69% and 0.7393, respectively. In contrast, the five-shot approach produced better results for category classification. This indicates that the same approach does not work uniformly across different tasks in LLMs.

Model	**exact match accuracy**	**overlap accuracy**	**micro f1**
SVM	68.62%	**73.83**%	**0.7510**
Gradient Boosting	65.02%	69.69%	0.7075
LSTM	62.61%	64.48%	0.7129
SBERT	**73.23**%	-	-
Phi3 (3-shot)	63.81%	69.69%	0.7393

Table 4.2: Results of Sentiment classification in different model

The results of different models that we implemented for sentiment polarity is summarized on table 4.2. Best value of each metric is highlighted in bold.

4.3 Joint classification

Joint classification is also a multi label multi class classification task. Both SVM and Gradient boost models were trained for joint classification task as well. They both were able to achieve decent results. SVM model achieved a micro f1 score of 0.6040 and Gradient boost model achieved micro f1 score of 0.5024. The results were lower than separate tasks which is as expected. With joint classification there are a lot more labels (number of category classes x number of sentiment polarity classes) meaning there are less reviews per class. This is most likely the cause of poorer performance. The overlap accuracy of SVM model was 54.47% but gradient boosting got a very poor performance with only 38.98% overlap accuracy. This result and poor performance suggests there is a lot of room for improvement and highlights the performance of more complex ML models. Jihan et al., 2017 achieved much better results with SVM on a similar but different dataset. They were able to achieve f1 score of 0.7303 on semeval 16 restaurant dataset.

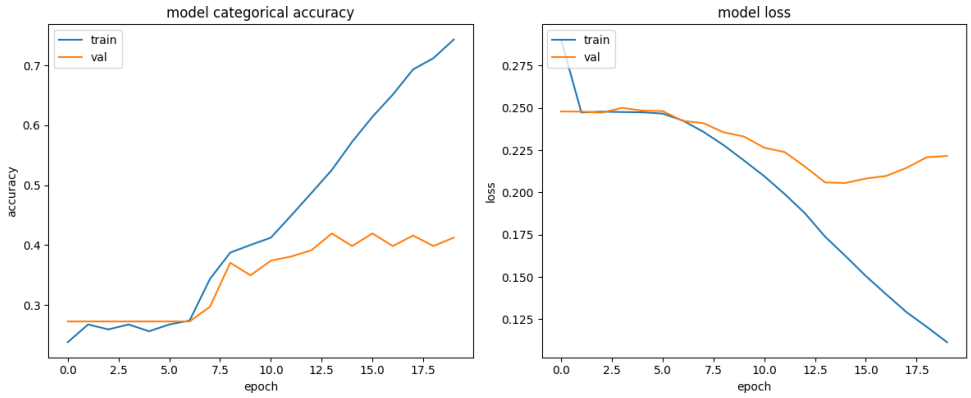


Figure 4.2: Accuracy and loss plot of LSTM model for Joint classification task

We did not see much improvement in performance with more complex models like LSTM. We only achieved exact match accuracy of 31.64% which was very poor. Micro f1 score was slightly better at 0.5042. The result could have been improved with some hyperparamters tuning but we did not delve deeper into LSTM rather we moved forward to training bert model. Figure 4.2 we can see the val loss goes after 12th epoch and accuracy starts to plateau as well. This however was not problematic for our evaluation as we had load best model and early stopping callbacks enabled for final training process.

Finally we experimented with using 'bert-base-uncased' model for joint classification task. We were expecting the bert model to perform really well as

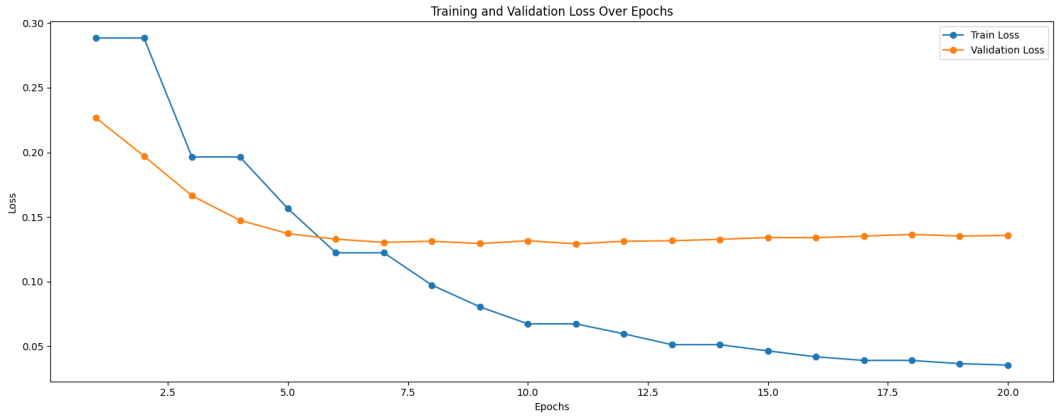


Figure 4.3: Accuracy and loss plot of BERT model for Joint classification task

it understands context and language really well. We trained the model for 20 epoches and got a micro f1 score score of 0.7334. The model predicted 61.28% of labels exactly which is about 20% higher than the second best performing model SVM and about double that of LSTM. As seen on Figure 4.3 the validation loss starts to flatten after 10th epoch but the the training loss was decreasing until the last epoch. Even with the gap between loss values at the end, the model performed as per our expectations. There has been some research with using BERT based models for ABSA as we discussed before. Najkov, 2023 used a bert model on macedonian resturant reviews dataset and achieved f1 score of 0.8. As the datasets are completely different, we cannot make comparisons but the scores are in same realm. Some more performance can be gained by utilizing a transformer based model which is more tuned for classification task. Or even using a more larger model like bert-large or other architectures like deberta could also yield better result. We didn't convert the LLM responses to joing classification format for joint classification evaluation as most of the LLM responses had additional aspects or were only partially correct.

Model	**exact match accuracy**	**overlap accuracy**	**micro f1**
SVM	42.05%	54.47%	0.6040
Gradient Boosting	27.10%	38.98%	0.5024
LSTM	31.64%	44.19%	0.5042
BERT	**61.28%**	**74.36%**	**0.7334**

Table 4.3: Results of Joint in different model

The results of different models that we implemented for joint classification is summarized on 4.3. Best value of each metric is highlighted in bold.

4.4 N-shot classfication

We only used phi-3-mini for n shot classification as it was a lighter model that allowed us to perform more experiments with it. We tried using Mistral model for same experiments as well but non quantized version failed due to memory limitations and AWQ quantized version estimated time at 15 hours, so we had to cancel it.

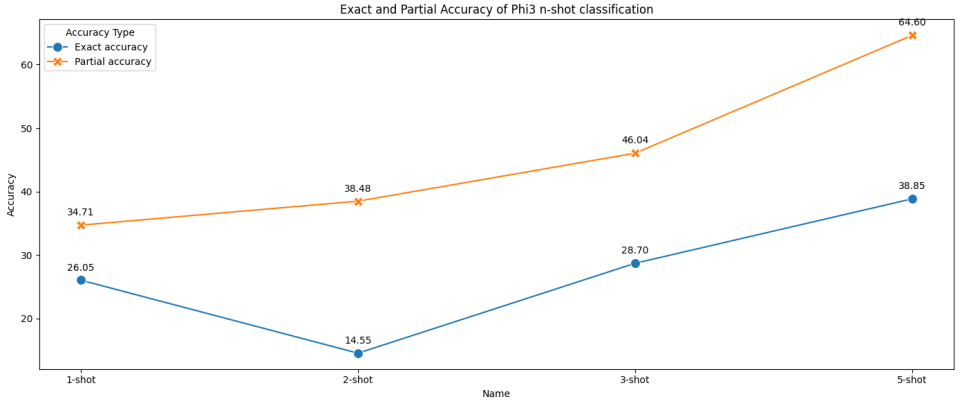


Figure 4.4: Exact and Partial Accuracy of Phi3 n-shot classification

As you can see in Figure 4.4, the graph displays exact and partial accuracy rates of the Phi3 model across different n-shot classifications (1-shot, 2-shot, 3-shot, and 5-shot). The exact accuracy (depicted in blue) starts at 26.05% for 1-shot, drops to 14.55% for 2-shot, and then increases to 28.70% for 3-shot, ultimately reaching 38.85% for 5-shot. On the other hand, the partial accuracy (depicted in orange) starts at 34.71% for 1-shot, increases consistently through 38.48% for 2-shot and 46.04% for 3-shot, and peaks at 61.60% for 5-shot. This indicates that while exact accuracy fluctuates, partial accuracy improves steadily with the increase in the number of shots. The Phi3 model performs best in terms of partial accuracy at the 5-shot classification, achieving its highest rate of 61.60%. The improved performance at higher shot classifications can be attributed to the model having more data points to learn from, thus enhancing its ability to generalize and capture the underlying patterns more effectively.

Model	Extra aspects	Rows with extra aspects	Formatting error rows
1-shot	1284	510	207
2-shot	921	479	129
3-shot	500	337	115
5-shot	329	266	69

Table 4.4: Phi3 error counts

As you can see in Table 4.4, the table presents error statistics for the Phi3 model across different n-shot classifications. The data reveals a decreasing trend in errors as the number of shots increases. Specifically, the number of extra aspects drops from 1,284 in 1-shot to 329 in 5-shot. When performing inference with one shot approach, there were 1284 new aspects in 510 rows out of 749 rows. This means in about 70% of cases, there was at least one extra aspect present in predictions. Even though we specify the possible aspects and sentiments in our prompt, the model hallucinates and produces these errors. Formatting error rows represent cases where the output is not formatted correctly which diminish from 207(1-shot) to 69(5-shot). This suggests that higher n-shot classifications not only enhance accuracy but also reduce various types of errors in the Phi3 model. The model performs best at the 5-shot classification, showing the least number of errors. As the model processes more examples, it refines its predictions and minimizes errors, resulting in more reliable data processing across all 749 rows.

4.5 Prompting

When testing an zero shot approach on single examples, we noticed a huge impact of prompts on the model. Using the words *"Do not add any further explanation"* usually resulted in a much cleaner output (still not perfect but much cleaner). But since we were not able to get properly formatted outputs with zero shot, we tried prompt tuning with one shot classification. The generated response was slightly different but as we had set Temperature to 0 (recommended on huggingface), model didn't provide a varied response. After minor cleanup and post processing of the response, any impact prompt had on the results was lost. The results were exactly same as one shot classification.

It might be because we didn't try prompts that were vastly different but using a much simpler prompt with less instructions produced garbage result all the time. We came to conclude that, if model creativity (temperature) is restricted then effects of prompt can be negated by using n-shot approaches upto a certain point.

4.6 Finetuning

Using finetuned models was straightforward and training process was also smooth. Figure 4.5 shows the loss plot for training process an we can observe occasional spikes in the process. As the loss seems to be lowest at step 200, we have performed evaluation with base model, fine tuned model at step 200 and final fine tuned model.

In Figure 4.5, we can see that loss curve has multiple sharp spikes throughout the training process. These spikes suggest moments where the

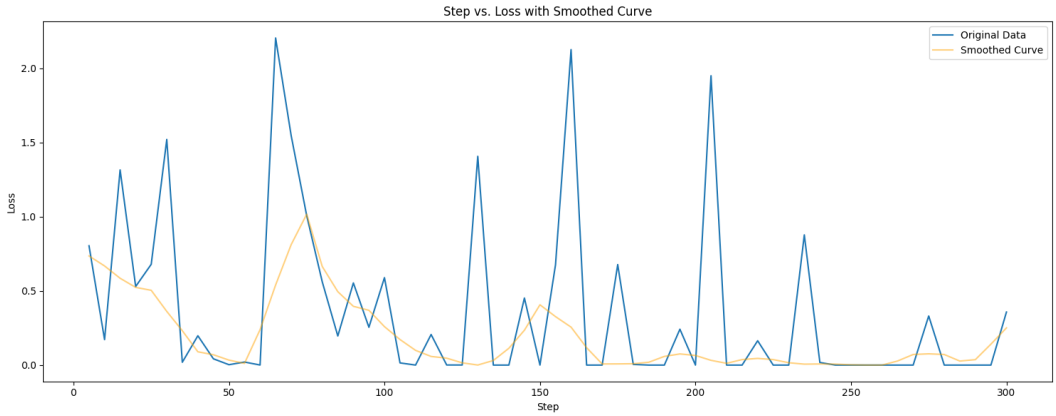


Figure 4.5: Step Vs loss curve for openai finetuning

model's performance temporarily degrades, resulting in higher loss values. Such fluctuations can be attributed to the use of a small batch size (1) and a limited dataset (100 examples). Small batch sizes are prone to higher variance in gradient estimation, leading to less stable updates. This can cause the model to experience periods of poor performance, manifesting as spikes in the loss curve.

The smoothed curve, in contrast, provides a clearer view of the overall trend, reducing the noise caused by individual data points. It indicates a general downward trend in the loss values, suggesting that despite the variability, the model is learning and improving over time.

Small batch size, small dataset size and model sensitivity could be a few reasons behind such spikes. Using a larger batch size can help stabilize the training process by averaging out the gradients, leading to more stable updates and reducing the frequency and magnitude of the spikes. Also increasing the number of training examples can provide a more robust learning process. A larger and more diverse dataset can help the model generalize better and reduce overfitting, resulting in a smoother loss curve.

As seen in Figure 4.6, the performance of the OpenAI GPT-3.5 Turbo model improves significantly with fine-tuning. The base model starts with an exact accuracy of 34.18% and a partial accuracy of 45.65%. After fine-tuning for 200 steps, the exact accuracy increases to 68.36%, and the partial accuracy rises to 88.46%. In the final fine-tuned model, the exact accuracy slightly improves to 69.56%, while the partial accuracy remains stable at 88.50%. This data demonstrates the effectiveness of fine-tuning in enhancing the model's performance.

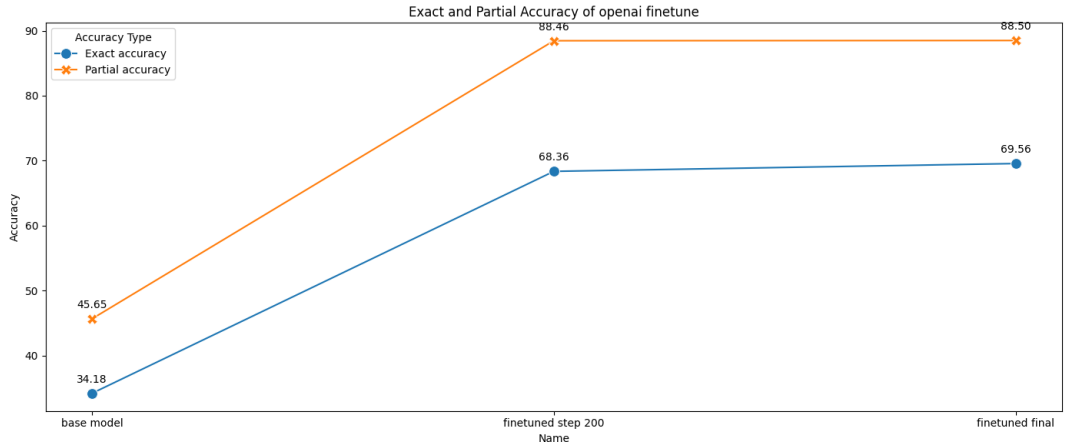


Figure 4.6: Exact and Partial Accuracy of GPT3.5 turbo

Model	Extra aspects	Rows with extra aspects	Formatting error rows
base model	43	41	374
finetuned step 200	117	113	0
finetuned final	86	83	0

Table 4.5: openai gpt finetuning error counts

Table 4.5 shows various types of errors for GPT 3.5 finetuning process. The base model shows a significant number of formatting error rows (374), whereas both fine-tuned steps exhibit no formatting errors. The number of extra aspects and rows with extra aspects also increase from the base model to the fine-tuned step 200, indicating an initial learning curve. However, these counts decrease slightly in the final fine-tuned model, suggesting an optimization and refinement in the model's learning process.

The poor performance of the base model can be attributed to its lack of task-specific training. Without fine-tuning, the model is not specialized and thus performs inadequately on specific tasks, resulting in lower accuracy and a high number of formatting errors. The initial fine-tuning phase at step 200 enhances the models understanding and reduces errors significantly, as the model starts to adapt to the specific requirements of the task.

To further improve performance, several strategies can be considered. Increasing the size of the training dataset beyond the 100 data samples used can provide the model with a broader range of examples to learn from, potentially enhancing its generalization capabilities. Additionally, implementing a more extensive fine-tuning process with incremental adjustments and validations can lead to more refined performance improvements. Exploring advanced fine-tuning techniques such as learning rate scheduling, data augmentation, and regularization

methods can also contribute to achieving higher accuracy and reduced error rates.

4.6.1 Cost Estimations

We used Google colab Pro for running LLM experiments. Including all the failed experiments and minor adjustments we needed about 150 compute units. All of the reported experiments were ran on L4 GPU which consume approximately 4.82 compute units per hour. It costs 87.32 DKK per 100 compute units which means it would cost about 4.21 DKK per hour to run experiments on L4 GPU.

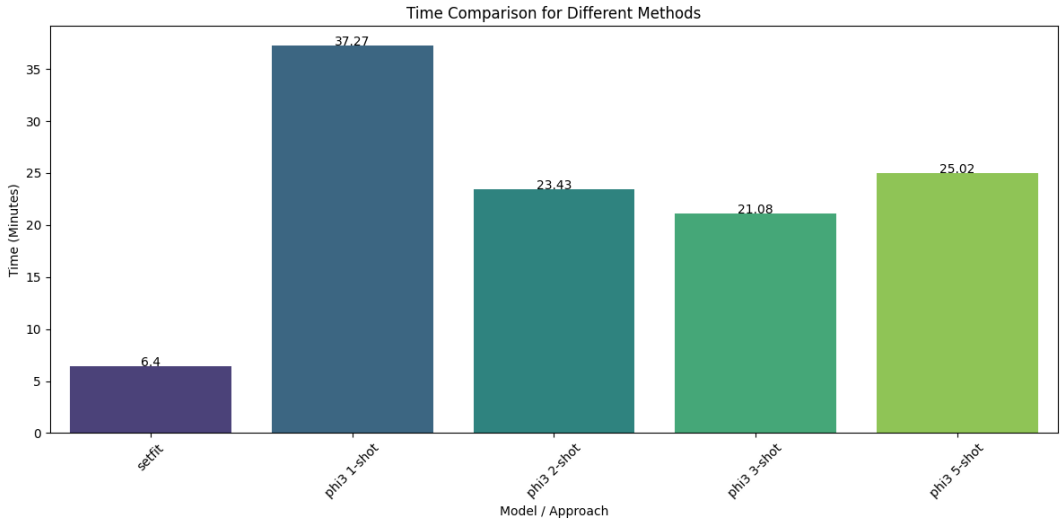


Figure 4.7: Time comparision for different methods

As we can see in Figure 4.7, Setfit for ABSA took minimal amount of time in comparison to phi 3 models. Then with phi-3 One shot approach took the longest at 37.27 minutes. It decreases for 2 shot and 3 shot but again increases for 5 shot. This is probably because of model caching as we ran all the experiments from single notebook.

If we take the L4 GPU cost, it would cost about 1.65 DKK at lowest. The cost at this scale is very minimal but could increase rapidly when we start using these products as part of our daily work. We can also compare this cost with OpenAI API. Using OpenAI API for finetuning cost us about 7.08 DKK for finetuning the models (only 100 examples) and about 3.5 DKK for evaluation (per approach).

OpenAI API might be expensive but if we look at the performance, it also performs better than phi3. We tried using a larger size model (mistral 7B) but it crashed due to memory limitations. Depending on the amount of budget and accuracy needed, either gpt or phi3 could be a useful appropach for ABSA.

Chapter 5

Conclusion

Over the years, research in the field of NLP has grown immensely with the introduction of transformers. This research is aimed at creating a understanding and historical approaches for ABSA. The knowledge and findings gained during this work is going to be immensely useful for the final dissertation work. This research has also equipped me with knowledge which could also be applied towards other NLP tasks.

In this thesis, we explored the application of Large Language Models (LLMs) for Aspect-Based Sentiment Analysis (ABSA). Our investigation started with traditional machine learning models such as Support Vector Machine (SVM) and Gradient Boosting, then advancing to more advanced models like Long Short-Term Memory (LSTM), Bidirectional Encoder Representations from Transformers (BERT), Sentence-BERT (SBERT), and Phi-3. We evaluated these models based on their performance in both category classification and sentiment polarity classification tasks. We also used LLM specific evaluation techniques for evaluating LLMs.

The models were implemented in scikit learn, keras, pytorch and hugginface transformers. An evaluation framework was also created so that the models could be compared with each other and also with past literature.

5.1 Key Findings

Here are some of the key conclusions from this research:

1. In simple classification tasks like category classification and sentiment polarity, traditional ML models like SVM still outperform or compete modern LLMs. In our case, SVM performed best on Category classification and sbert performed best in sentiment polarity.

2. In joint classification tasks also simpler transformer models like BERT perform comparitive to LLMs.

3. LLMs however come with an advantage that they can be used without any training as well with n-shot approaches.

4. If accuracy is important, performing even minor amount of finetuning seems to have a big impact on performance. This was shown by GPT-3.5 turbo model as its accuracy increased to 88.5% from 45.65% after finetuning on 100 sample.

5. The cost analysis revealed that running experiments on Google's L4 GPU was relatively affordable, with fine-tuning costs being minimal compared to the performance benefits obtained.

6. Using open source model is cheaper than paid APIs but come at a cost of lower performance as well.

7. Open source models like Phi-3 do show comparable performance but when cost of training and maintenance is taken into account, API based approaches seem to be more beneficial.

8. Paid APIs should be used with caution as the prices per token might increase in future leading to a higher budget and data safety also should be considered while using these APIs.

5.2 Limitations

Despite the promising results, several limitations were identified in this study:

1. **Data Availability:** The models were trained and tested on the SemEval-2014 Task 4 dataset, which may not fully represent the diversity of real-world data. The dataset is also relatively small, limiting the ability to generalize the results to larger and more varied datasets.

2. **Resource constrains:** Advanced models like Mistral and GPT require significant computational resources for training and fine-tuning. This was a barrier for us to perform further experiments. We couldn't perform any experiments with Mistral and Mixtral due to this limitation.

3. **hallucinations:** Generative models such as GPT and Phi3 are extremely prone to hallucinations. This can cause a lot of problems in classification tasks such as ABSA.

4. **Aspect Ambiguity:** In cases where a sentence contains multiple aspects or ambiguous language, accurately identifying and classifying these aspects remains challenging for all tested models.

5. **Domain Adaptation:** Models trained on the restaurant domain may not perform well in other domains without further fine-tuning and adaptation, highlighting the need for domain-specific training data.

6. **Limited Knowledge:** My limited knowledge and past experience with NLP models and tools might have led to oversights or missed opportunities for a more detailed understanding. Despite these limitations, we have made our best efforts to provide valuable insights within the given constraints, acknowledging the inherent challenges in studying a complex and evolving field such as ABSA.

5.3 Future Work

Although we have explored various approaches to aspect based sentiment analysis, there are some more area we can look into. The field of aspect-based sentiment analysis continues to evolve with the introduction of new models and techniques. Several strategies can be adopted to further improve model performance. Future research should focus on conducting analysis with latest papers and techniques in sentiment analysis and natural language processing.

1. As different models perform best with different prompts, future research should focus on performing more more extensive prompt engineering at per model level.

2. Fine tuning open source models might also give them a big boost in performance similar to GPT. This should be further explored in future along with experimenting with different parameters for finetuning.

3. Larger batch sizes and data sizes for fine tuning can also be explored in future.

4. Also comparing phi-3 with other open source models would also be possible for future work.

The application of LLMs for ABSA shows promising results, particularly in sentiment classification. Fine-tuning pre-trained LLMs like GPT-3.5 Turbo significantly enhances their performance, making them viable for specific ABSA tasks. Future work should focus on optimizing model architectures, expanding datasets, and exploring advanced fine-tuning techniques to improve the robustness and accuracy of these models in real-world applications.

Bibliography

Zhang, H., Cheah, Y.-N., Alyasiri, O. M., & An, J. (2023, June). *A Survey on Aspect-Based Sentiment Quadruple Extraction with Implicit Aspects and Opinions* (Preprint). In Review. https://doi.org/10.21203/rs.3.rs-3098487/v1

Lauriola, I., Lavelli, A., & Aiolli, F. (2022). An introduction to Deep Learning in Natural Language Processing: Models, techniques, and tools. *Neurocomputing*, 443–456. https://doi.org/10.1016/j.neucom.2021.05.103

Tsytsarau, M., & Palpanas, T. (2012). Survey on mining subjective data on the web. *Data Mining and Knowledge Discovery*, *24*(3), 478–514. https://doi.org/10.1007/s10618-011-0238-6

Techtarget, N. B. (2023). What Is Sentiment Analysis (Opinion Mining)? | Definition from TechTarget.

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093–1113. https://doi.org/10.1016/j.asej.2014.04.011

Hoang, M., Bihorac, O. A., & Rouces, J. Aspect-Based Sentiment Analysis using BERT. In: In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*. Turku, Finland: Linköping University Electronic Press, 2019, September, 187–196.

Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., & Manandhar, S. SemEval-2014 Task 4: Aspect Based Sentiment Analysis (P. Nakov & T. Zesch, Eds.). In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* (P. Nakov & T. Zesch, Eds.). Ed. by Nakov, P., & Zesch, T. Dublin, Ireland: Association for Computational Linguistics, 2014, August, 27–35. https://doi.org/10.3115/v1/S14-2004

Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., & Androutsopoulos, I. SemEval-2015 Task 12: Aspect Based Sentiment Analysis (P. Nakov, T. Zesch, D. Cer, & D. Jurgens, Eds.). In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (P. Nakov, T. Zesch, D. Cer, & D. Jurgens, Eds.). Ed. by Nakov, P., Zesch, T., Cer, D., & Jurgens, D.

Denver, Colorado: Association for Computational Linguistics, 2015, June, 486–495. https://doi.org/10.18653/v1/S15-2082

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., AL-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., De Clercq, O., Hoste, V., Apidianaki, M., Tannier, X., Loukachevitch, N., Kotelnikov, E., Bel, N., Jiménez-Zafra, S. M., & Eryiğit, G. SemEval-2016 Task 5: Aspect Based Sentiment Analysis (S. Bethard, M. Carpuat, D. Cer, D. Jurgens, P. Nakov, & T. Zesch, Eds.). In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)* (S. Bethard, M. Carpuat, D. Cer, D. Jurgens, P. Nakov, & T. Zesch, Eds.). Ed. by Bethard, S., Carpuat, M., Cer, D., Jurgens, D., Nakov, P., & Zesch, T. San Diego, California: Association for Computational Linguistics, 2016, June, 19–30. https://doi.org/10.18653/v1/S16-1002

Nazir, A., Rao, Y., Wu, L., & Sun, L. (2022). Issues and Challenges of Aspect-based Sentiment Analysis: A Comprehensive Survey. *IEEE Transactions on Affective Computing*, *13*(2), 845–863. https://doi.org/10.1109/TAFFC.2020.2970399

Ma, Y., Peng, H., & Cambria, E. (2018). Targeted Aspect-Based Sentiment Analysis via Embedding Commonsense Knowledge into an Attentive LSTM. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1). https://doi.org/10.1609/aaai.v32i1.12048

Kiritchenko, S., Zhu, X., Cherry, C., & Mohammad, S. NRC-Canada-2014: Detecting Aspects and Sentiment in Customer Reviews (P. Nakov & T. Zesch, Eds.). In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* (P. Nakov & T. Zesch, Eds.). Ed. by Nakov, P., & Zesch, T. Dublin, Ireland: Association for Computational Linguistics, 2014, August, 437–442. https://doi.org/10.3115/v1/S14-2076

Wagner, J., Arora, P., Cortes, S., Barman, U., Bogdanova, D., Foster, J., & Tounsi, L. DCU: Aspect-based Polarity Classification for SemEval Task 4 (P. Nakov & T. Zesch, Eds.). In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* (P. Nakov & T. Zesch, Eds.). Ed. by Nakov, P., & Zesch, T. Dublin, Ireland: Association for Computational Linguistics, 2014, August, 223–229. https://doi.org/10.3115/v1/S14-2036

Bhoi, A., & Joshi, S. (2018, May). Various Approaches to Aspect-based Sentiment Analysis.

Wei, W., Liu, Y.-P., & Wei, L. (2020). Feature-level sentiment analysis based on rules and fine-grained domain ontology. *Knowledge Organization*, *47*, 105–121. https://api.semanticscholar.org/CorpusID:216207172

Onwuegbuche, F. C., Wafula, J. M., & Mung'atu, J. K. (2019). Support Vector Machine for Sentiment Analysis of Nigerian Banks Financial Tweets. *Journal of Data Analysis and Information Processing*, *7*(4), 153–173. https://doi.org/10.4236/jdaip.2019.74010

Jihan, N., Senarath, Y., Tennekoon, D., Wickramarathne, M., & Ranathunga, S. Multi-Domain Aspect Extraction Using Support Vector Machines (L.-W. Ku & Y. Tsao, Eds.). In: *Proceedings of the 29th Conference on Computational Linguistics and Speech Processing (ROCLING 2017)* (L.-W. Ku & Y. Tsao, Eds.). Ed. by Ku, L.-W., & Tsao, Y. Taipei, Taiwan: The Association for Computational Linguistics and Chinese Language Processing (ACLCLP), 2017, November, 308–322.

Najkov, D. (2023, March). From Decision Trees to Transformers: Comparing Sentiment Analysis Models for Macedonian Restaurant...

Zhu, L., Xu, M., Bao, Y., Xu, Y., & Kong, X. (2022). Deep learning for aspect-based sentiment analysis: A review. *PeerJ Computer Science*, *8*, e1044. https://doi.org/10.7717/peerj-cs.1044

Do, H. H., Prasad, PWC, Maag, A., & Alsadoon, A. (2019). Deep Learning for Aspect-Based Sentiment Analysis: A Comparative Review. *Expert Systems with Applications*, *118*, 272–299. https://doi.org/10.1016/j.eswa.2018.10.003

Shu, L., Xu, H., & Liu, B. (2019, May). Controlled CNN-based Sequence Labeling for Aspect Extraction.

Noh, Y., Park, S., & Park, S.-B. (2019). Aspect-Based Sentiment Analysis Using Aspect Map. *Applied Sciences*, *9*(16), 3239. https://doi.org/10.3390/app9163239

Wang, X., Li, F., Zhang, Z., Xu, G., Zhang, J., & Sun, X. (2021). A unified position-aware convolutional neural network for aspect based sentiment analysis. *Neurocomputing*, *450*, 91–103. https://doi.org/10.1016/j.neucom.2021.03.092

Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *WIREs Data Mining and Knowledge Discovery*, *8*(4), e1253. https://doi.org/10.1002/widm.1253

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Tang, D., Qin, B., Feng, X., & Liu, T. (2016, September). Effective LSTMs for Target-Dependent Sentiment Classification.

Wang, Y., Huang, M., Zhu, X., & Zhao, L. Attention-based LSTM for Aspect-level Sentiment Classification (J. Su, K. Duh, & X. Carreras, Eds.). In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (J. Su, K. Duh, & X. Carreras, Eds.). Ed. by Su, J., Duh, K., & Carreras, X. Austin, Texas: Association for Computational Linguistics, 2016, November, 606–615. https://doi.org/10.18653/v1/D16-1058

Cai, G., & Li, H. Joint Attention LSTM Network for Aspect-Level Sentiment Analysis (S. Zhang, T.-Y. Liu, X. Li, J. Guo, & C. Li, Eds.). In: *Information Retrieval* (S. Zhang, T.-Y. Liu, X. Li, J. Guo, & C. Li, Eds.). Ed. by Zhang, S., Liu, T.-Y., Li, X., Guo, J., & Li, C. Vol. 11168. Cham: Springer International

Publishing, 2018, pp. 147–157. ISBN: 978-3-030-01011-9 978-3-030-01012-6. https://doi.org/10.1007/978-3-030-01012-6_12

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/ARXIV.1810.04805

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020, July). Language Models are Few-Shot Learners.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., … Fiedel, N. (2022, October). PaLM: Scaling Language Modeling with Pathways.

Tay, Y., Dehghani, M., Tran, V. Q., Garcia, X., Wei, J., Wang, X., Chung, H. W., Shakeri, S., Bahri, D., Schuster, T., Zheng, H. S., Zhou, D., Houlsby, N., & Metzler, D. (2023, February). UL2: Unifying Language Learning Paradigms.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023, February). LLaMA: Open and Efficient Foundation Language Models.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., … Rush, A. M. (2020, July). HuggingFace's Transformers: State-of-the-art Natural Language Processing. https://doi.org/10.48550/arXiv.1910.03771

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. Attention is All you Need. In: In *Advances in Neural Information Processing Systems*. *30*. Curran Associates, Inc., 2017.

Hao, Y., Dong, L., Wei, F., & Xu, K. (2019, August). Visualizing and Understanding the Effectiveness of BERT.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding (T. Linzen, G. Chrupa\la, & A. Alishahi, Eds.). In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (T. Linzen, G. Chrupa\la, & A. Alishahi, Eds.). Ed. by Linzen, T., Chrupa\la, G., & Alishahi, A. Brussels, Belgium: Association for Computational Linguistics, 2018, November, 353–355. https://doi.org/10.18653/v1/W18-5446

Mewada, A., & Dewang, R. K. (2023). SA-ASBA: A hybrid model for aspect-based sentiment analysis using synthetic attention in pre-trained language BERT model with extreme gradient boosting. *The Journal of Supercomputing*, *79*(5), 5516–5551. https://doi.org/10.1007/s11227-022-04881-x

Radford, A., & Narasimhan, K. Improving Language Understanding by Generative Pre-Training. In: 2018.

Mingzheng, L., Zelin, H., Jiadong, L., & Wei, L. Aspect-level sentiment analysis model fused with GPT and multi-layer attention. In: In *Third International Conference on Artificial Intelligence and Computer Engineering (ICAICE 2022). 12610*. SPIE, 2023, April, 279–284. https://doi.org/10.1117/12.2671363

Simmering, P. F., & Huoviala, P. (2023, October). Large language models for aspect-based sentiment analysis. https://doi.org/10.48550/arXiv.2310.18025

Scaria, K., Gupta, H., Goyal, S., Sawant, S. A., Mishra, S., & Baral, C. (2023, May). InstructABSA: Instruction Learning for Aspect Based Sentiment Analysis.

Wang, Z., Xie, Q., Ding, Z., Feng, Y., & Xia, R. (2023, April). Is ChatGPT a Good Sentiment Analyzer? A Preliminary Study.

Reimers, N., & Gurevych, I. (2019, August). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., & Sayed, W. E. (2023, October). Mistral 7B.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., … Sayed, W. E. (2024, January). Mixtral of Experts. https://doi.org/10.48550/arXiv.2401.04088

Abdin, M., Jacobs, S. A., Awan, A. A., Aneja, J., Awadallah, A., Awadalla, H., Bach, N., Bahree, A., Bakhtiari, A., Behl, H., Benhaim, A., Bilenko, M., Bjorck, J., Bubeck, S., Cai, M., Mendes, C. C. T., Chen, W., Chaudhary, V., Chopra, P., … Zhou, X. (2024, April). Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. https://doi.org/10.48550/arXiv.2404.14219

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2018). Language Models are Unsupervised Multitask Learners.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2023, September). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. https://doi.org/10.48550/arXiv.1910.10683

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022, March). Training language models to follow instructions with human feedback.

Bang, Y., Cahyawijaya, S., Lee, N., Dai, W., Su, D., Wilie, B., Lovenia, H., Ji, Z., Yu, T., Chung, W., Do, Q. V., Xu, Y., & Fung, P. (2023, November). A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity.

Ghashami, M. (2023, November). On Tokenization In LLMs.

google deepmind. (2024). Building Open Models Responsibly in the Gemini Era.

run:ai. (2023). LLM Training: How It Works and 4 Key Considerations.

Xie, S. M., Raghunathan, A., Liang, P., & Ma, T. (2022, July). An Explanation of In-context Learning as Implicit Bayesian Inference.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023, February). A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT.

Weng, L. (2023, March). Prompt Engineering.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Guo, Q., Wang, M., & Wang, H. (2024, January). Retrieval-Augmented Generation for Large Language Models: A Survey.

Rawat, A. S., Thrampoulidis, C., Soltanolkotabi, M., & Oymak, S. Towards understanding the role of attention in prompt-tuning. In: In *Icml 2023 (to appear)*. 2023.

Wang, C., Yang, Y., Gao, C., Peng, Y., Zhang, H., & Lyu, M. R. No More Fine-Tuning? An Experimental Evaluation of Prompt Tuning in Code Intelligence. In: In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2022, November, 382–394. https://doi.org/10.1145/3540250.3549113 arXiv: 2207.11680 [cs].

Raschka, S. (2023, April). Finetuning Large Language Models.

NVIDIA. (2024). SFT and PEFT.

Nathan Lambert, L. v. W. A. H., Louis Castericato. (2022). Illustrating Reinforcement Learning from Human Feedback (RLHF).

Song, X. (2021, December). A Fast WordPiece Tokenization System.

Tunstall, L., Reimers, N., Jo, U. E. S., Bates, L., Korat, D., Wasserblat, M., & Pereg, O. (2022). Efficient few-shot learning without prompts. https://doi.org/10.48550/ARXIV.2209.11055

Singh, K. (2024, May). [Machine Learning] Fine Tuning Open Source Large Language Models (PEFT QLoRA) on Azure Machine https : / / blog .

devgenius . io / fine - tuning - large - language - models - on - azure - machine - learning-358338f4e66a

Appendix A

Prompts use for prompt engineering

```
Your task is to analyse and perform aspect based sentiment analysis on the
given restaurant reviews text. The aspects should only be ["food",
"service","ambience", "price", "other"] and sentiments should only be
["positive", "neutral","negative"]. Output should only contain mentioned
aspects and their respective sentiments as a dictionary. Each aspects
should only have one sentiment and not every aspect is necessarily
present. Do not provide any further explanation.
```

Listing A.1: Prompt-1 used for prompt engineering with Phi3

```
You are a skilled data analyst. Your task is to extract and analyze
sentiments from user reviews and structure the information into a JSON
format. Analyze the provided user reviews and create a JSON summary. Focus
on these key aspects: ["food", "service", "ambience", "price", "other"]
and sentiments: ["positive", "neutral", "negative"].
```

Listing A.2: Prompt-2 used for prompt engineering with Phi3

```
Your task is to perform aspect-based sentiment analysis on the following
restaurant review texts. Focus specifically on the aspects: ["food",
"service", "ambience", "price", "other"]. Each aspect should be assigned a
sentiment from the options ["positive", "neutral", "negative"]. Provide
the output in a dictionary format where the keys are the aspects and the
values are the corresponding sentiments. Each aspect should have only one
sentiment, and it's not necessary for all aspects to be present in every
review. Do not include any further explanation or additional text.
```

Listing A.3: Prompt-3 used for prompt engineering with Phi3