Universidade de Aveiro
2022

Roshan Poudel

Explorando transformadores para análise de sentimento baseada em aspecto

Exploring Transformers for Aspect Based Sentiment Analysis

Universidade de Aveiro
2022

Roshan Poudel

Explorando transformadores para análise de sentimento baseada em aspecto

Exploring Transformers for Aspect Based Sentiment Analysis

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Ciência de Dados, realizada sob a orientação científica de Doutor Sérgio Guilherme Aleixo de Matos, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor Rui Pedro Lopes, Professor Coordenador do Instituto Politécnico de Bragança (IPB).

Este trabalho foi apoiado pela FCT – Fundação para a Ciência e Tecnologia no âmbito do projeto DSAIPA/AI/0088/2020.

agradecimentos/ acknowledgments

I would like to express my deepest gratitude to my friends and family who have supported me throughout the journey. Special thanks to my advisors, Doctor Sérgio Matos and Doctor Rui Lopes for motivation, encouraging words and putting up with delays. Furthermore, I am immensely grateful to my closest friends in IEETA for keeping me motivated and providing continuous support. Their encouragement, insights, and assistance have been invaluable. Finally I want to thank Pandia project and LCA Coimbra for providing access to computational resource.

Palavras-chave

Análise de sentimento, análise de sentimento baseada em aspectos (ABSA), Transformers, redes neuronais profundas, processamento de linguagem natural (NLP)

Resumo

Nos últimos anos, a análise de sentimentos ganhou atenção significativa por sua ampla gama de aplicações em vários domínios. A análise de sentimentos baseada em aspectos (ABSA) é uma tarefa desafiadora dentro da análise de sentimentos que visa identificar a polaridade do sentimento em relação a aspectos ou atributos específicos de uma entidade alvo em um determinado texto. Os transformadores, um tipo de arquitetura de rede neural profunda, mostraram resultados promissores em muitas tarefas de processamento de linguagem natural (NLP), incluindo análise de sentimento.

Esta dissertação explora a eficácia de um modelo BERT+BiLSTM+CRF para ABSA e investiga o impacto dos tamanhos dos modelos e congelamento de camadas. Foram realizadas várias experiências usando diferentes conjuntos de dados ABSA, comparando os resultados com modelos de última geração existentes. Os resultados indicam que aumentar o tamanho do modelo não é necessariamente a melhor abordagem para melhorar o desempenho, e congelar um subconjunto de camadas pode levar a resultados comparáveis com requisitos computacionais reduzidos. O estudo também destaca o impacto dos métodos de pré-treino e conjuntos de dados em tarefas posteriores. O sistema *end-to-end* desenvolvido é modular, permitindo a substituição do modelo BERT por qualquer outro modelo baseado em transformador baseado no caso de uso. A pesquisa contribui para a compreensão de modelos baseados em transformadores para ABSA e fornece indicadores para estudos futuros neste campo.

Keywords

Sentiment analysis, Aspect-based sentiment analysis (ABSA), Transformers, Deep learning, Natural language processing (NLP)

Abstract

In recent years, sentiment analysis has gained significant attention for its wide range of applications in various domains. Aspect-based sentiment analysis (ABSA) is a challenging task within sentiment analysis that aims to identify the sentiment polarity towards specific aspects or attributes of a target entity in a given text. Transformers, a type of deep neural network architecture, have shown promising results in many natural language processing (NLP) tasks, including sentiment analysis.
This dissertation explores the effectiveness of a BERT+BiLSTM+CRF model for ABSA and investigates the impact of model sizes and layer freezing. Several experiments are performed using different ABSA datasets, comparing the results with existing state-of-the-art models. The findings indicate that increasing model size is not necessarily the best approach to improve performance, and freezing a subset of layers can lead to comparable results with reduced computational requirements. The study also highlights the impact of pretraining methods and datasets in downstream tasks. The developed end-to-end system is modular, allowing for the replacement of BERT with any other transformer-based model based on the use case. The research contributes to the understanding of transformer-based models for ABSA and provides insights for future studies in this field.

Table of contents

List of figures

List of tables

List of abbreviations

ABSA	Aspect Based Sentiment Analysis
ANN	Artificial Neural Network
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
CRF	Conditional Random Fields
LSTM	Long Short Term Memory
ML	Machine Learning
NLP	Natural Language Processing
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TPE	Tree-structured Parzen Estimator

Chapter 1

Introduction

This chapter provides a very short introduction to the work followed by the motivation behind the work. It also highlights the objectives and contributions of this work.

In recent years, sentiment analysis has gained significant attention due to its potential applications in various domains such as customer feedback analysis, political opinions, and social media monitoring [1]. Aspect based sentiment analysis (ABSA) is a subtask of sentiment analysis that aims to identify the sentiment polarity of specific aspects or attributes of a target entity in a given text [2]. ABSA is a challenging task as it requires the model to not only identify the relevant aspects but also understand the sentiment expressed towards them.

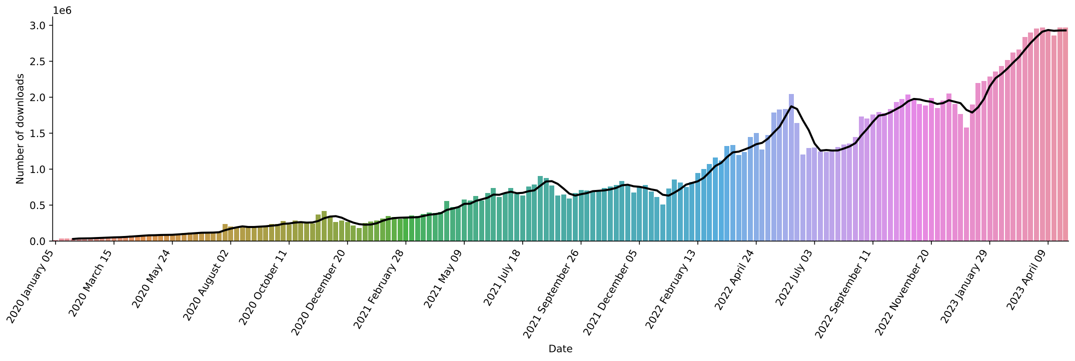


Figure 1.1: Daily installs of Transformers package from PyPI, Created by parsing public BigQuery entries of PyPi

Transformers, a type of deep neural network architecture, have shown promising results in various natural language processing (NLP) tasks including sentiment analysis. In particular, the Bidirectional Encoder Representations from Transformers (BERT) model has achieved state-of-the-art results in many NLP benchmarks. The success of BERT has also led to the development of other transformer-based models such as GPT-2, T5, and GShard. These models have pushed the limits of NLP and achieved human-level performance on several benchmarks. Figure 1.1 shows the number of daily downloads of transformers package from Python Package Index (PyPI) and we can clearly see the boom it has gotten in recent years. There have been works that use Transformers for ABSA, but there is still a lot of experiments to be performed.

In this dissertation, the effectiveness of a BERT+BiLSTM+CRF model for ABSA is explored. Several experiments are performed during the work to observe the impact of freezing layers and model sizes. This work also investigates the performance of this model on several ABSA datasets and compares it with existing state-of-the-art models. The use of LSTM and CRF layers in conjunction with BERT allows to capture the contextual information and improve the model's ability to handle long sequences. Additionally, the impact of fine-tuning BERT on different ABSA datasets and analyze the results to gain insights into the effectiveness of the approach is also explored. The complete end to end system is developed in a modular way, so one can replace BERT with any other transformer based model based on the use case.

In this work, PyTorch is used as main development framework. Other than PyTorch, some additional python packages like spacy and metrics are also used. PyTorch was chosen as huggingface transformers work best with it and is also comparable in performance with other python packages.

1.1 Objectives

The main goal of this dissertation is to experiment the usage of transformer based models for ABSA and answer some important questions relating to freezing of layers and model sizes. We aim to answer the question "Is increasing model size the best way to improve performance of a model?". To see the effectiveness of experiments, the results would be compared with other experiments and current state-of-the-art models as part of evaluation.

1.2 Contributions

This dissertation includes the following contributions in ABSA and Transformers research:

- A modular end to end system that can be used with any transformer based model for ABSA or any other token classification task.
- Experimentation and insights into the impact of model sizes and layer freezing in Transformers.
- Highlights the impact of pretraining method and dataset in downstream tasks.
- This work also contributed to MedProcNER challenge where the model used in this dissertation was used to identify medical procedures.

1.3 Document Structure

This dissertation uses the template provided by University of Aveiro and maintained by Rui Atunes[1]. The document is divided into five chapters. The chapters are organized as follows:

- Chapter 2 provides background knowledge and contextual information related with the work. Introduction to different aspects of Transformers and ABSA is also provided in the chapter.
- Chapter 3 presents how the overall pipeline was developed and explains the workings of each module. It also explains different datasets used and the metrics that are used to compare our experiments.
- Chapter 4 discusses the results of the experiments. After that the results are also compared to existing state-of-the-art methods.
- Chapter 5 concludes the work and provides the findings in brief. This chapter also proposes some future work and use cases of the system.

[1] https://github.com/ruiantunes/ua-thesis-template

Chapter 2

Background

This chapter aims to explain the background topics and concepts relating to Machine Learning and Natural Language Processing methods used in this dissertation.

With the advent of new digital data age, humans are finding new ways of using data and creating systems around it. Taking inspiration from human brain and neurons Artificial Neural Network (ANN) were created to make sense of this new available data. An ANN is composed of layers of interconnected nodes, or neurons that are designed to process and analyze data. In an ANN, data is fed into the input layer, and then it passes through one or more hidden layers, where the neurons process the input and make predictions. Finally, the output layer produces the final output of the model. During training, the neural network adjusts the weights of the connections between the neurons to minimize the error between the predicted output and the actual output and over time the neural network is able to learn from the data and improve its performance. Deep learning then can be simply explained as a neural network with multiple hidden layers [3]. Figure 2.1 shows a simple representation of a deep neural network architecture.

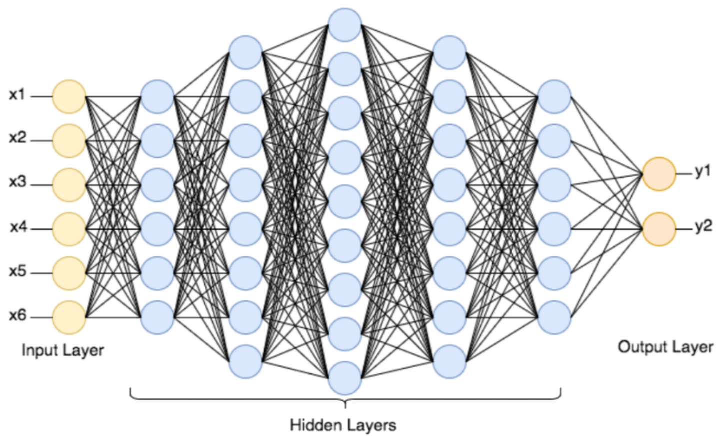


Figure 2.1: A simple neural network architecture with Input layer (6 nodes), 5 hidden layers and Output layer (2 nodes)

2.1 Natural Language Processing

Natural language can be described as the natural way humans communicate with each other (for example text and speech) and Natural Language Processing (NLP) is an area of research and application exploring how computer systems can be used to understand and process the natural language [4]. With the vast availability of data NLP has a wide range of applications including language translation, text summarization, text generation, sentiment analysis and a lot more. In recent years, NLP has boomed a lot due to advancements in Deep Learning and Machine Learning (ML) in general. In recent years the early approaches for named entity recognition with ML (which were based on Support Vector Machine (SVM)s and Conditional Random Fields (CRF)s) are being replaced and outperformed by recurrent architectures and deep learning models [5]. Similarly, some recent models have even outperformed humans in some tasks such as Question Answering or Spam detection [5].

Traditionally NLP tasks were performed with dictionary matching, rules and SVMs/CRFs. Then there were early deep learning approaches that used 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. But with deep learning models such as transformer-based language models, embeddings are contextual and are generated based on input sentence or sequence of words. The contextual information captured by these embeddings can help improve the performance of NLP tasks, such as sentiment analysis, named entity recognition, and machine translation. With contextual embeddings, same word can have a different embedding vector based on the input sequence.

2.2 Recurrent Neural Network

Recurrent Neural Network (RNN) is a type of artificial neural network that is designed to process sequences of data, such as text, speech, or time series data. Originally introduced as Hopfield Networks, RNN perform the same task for every element of a sequence, with the output being dependent on the previous computations [6]. The main idea behind RNN is to maintain a hidden state that captures the context of the past inputs and allows the network to make predictions based on that context. RNN are particularly useful for tasks that involve sequential data, such as language translation or speech recognition. They have also been applied to other types of data, such as video and music, where they can be used to model temporal dependencies in the data.

However, the standard RNN suffers from the problem of vanishing gradients, which means that the gradient signal becomes very small as it propagates back in time, making it difficult for the network to learn long-term dependencies [8]. To overcome this problem Long Short Term Memory (LSTM) architecture was introduced, it introduces a memory cell that allows the network to selectively forget or remember information over time. The

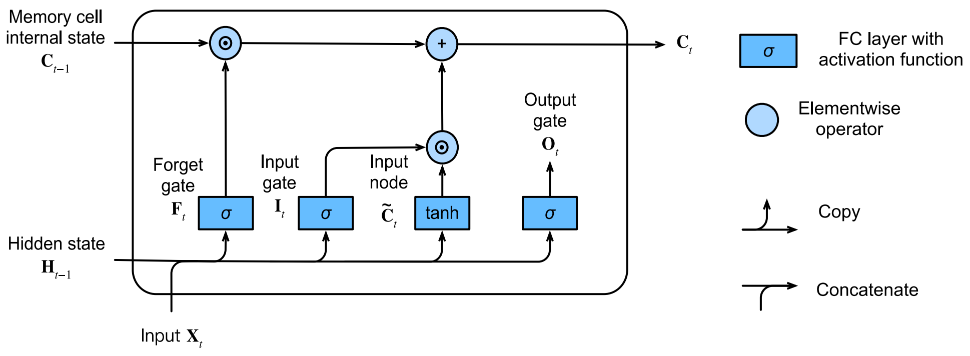

Memory cell internal state $\mathbf{C}_{t-1}$
$\mathbf{C}_t$
Output gate $\mathbf{O}_t$
Forget gate $\mathbf{F}_t$
Input gate $\mathbf{I}_t$
Input node $\tilde{\mathbf{C}}_t$
σ
tanh
Hidden state $\mathbf{H}_{t-1}$
Input $\mathbf{X}_t$
FC layer with activation function
Elementwise operator
Copy
Concatenate

Figure 2.2: Input gate, Forget gate, and Output gate in an LSTM cell. 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. (Source [7])

cell is controlled by three gates that determine how much of the previous cell state and input should be remembered, forgotten, or used to update the current cell state (Figure 2.2). LSTM uses input gate controls the flow of information from the input to the cell state, forget gate controls the flow of information from the previous cell state to the current cell state and output gate controls the flow of information from the current cell state to the output. By selectively forgetting or remembering information, the LSTM is able to maintain long-term dependencies while avoiding the problem of vanishing gradients [9]. Since its introduction LSTM has become a popular choice for many sequential processing tasks, such as speech recognition, machine translation, and sentiment analysis.

2.3 Attention mechanism

Attention in ML is a technique created to mimic human cognitive attention. In simple terms, similarly to how human brain pays attention to certain parts of images or sentence attention in deep learning also works by having a vector of importance weights that allows the network to enhance some parts of the input data while diminishing other parts [10].

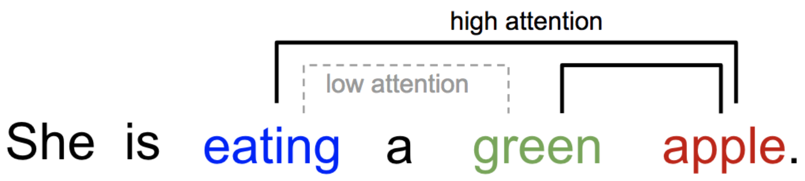


Figure 2.3: Intuitive example of attention in a simple sentence. The word eating and green would have a higher attention towards apple but the attention between eating and green is weak as they do not have meaning to each other. (Source [10])

Attention mechanism was introduced for a task of Neural Machine Translation in 2015 to solve the issue of traditional neural network not being able to handle long sequences [11]. Before the introduction of attention mechanism, the performance of encode-decoder deteriorates with the increase in the length of input sentences [12]. This is caused as an encoder-decoder based neural network must be able to compress all the necessary information of a source into a fixed length vector which created difficulty for the network to cope with longer sentences. The intuition behind attention is rather than compressing the input sequences, the decoder revisits the input sequence at every step but rather than always seeing the same representation of the input, the decoder selectively focuses on parts of the input sequence at particular decoding steps [11]. It can be summarized as Pooling which involves creating a summary representation of the input sequence based on which the attention scores will be calculated and Scoring involves calculating the attention scores, which indicate the relevance or similarity between the query vector and different parts of the input sequence.

Considering $\mathcal{D}=\{(\mathbf{k}_1,\mathbf{v}_1),\dots(\mathbf{k}_m,\mathbf{v}_m)\}$ a database of m tuples of keys and values. Moreover, denote by q a query. Then attention over D can be defined as:

$$\text{Attention}(\mathbf{q},\mathcal{D})=\sum_{i=1}^{m}\alpha(\mathbf{q},\mathbf{k}_i)\mathbf{v}_i,$$

where $\alpha(\mathbf{q},\mathbf{k}_i)\in\mathbb{R}$ are scalar attention weights. This is also called as Attention pooling.

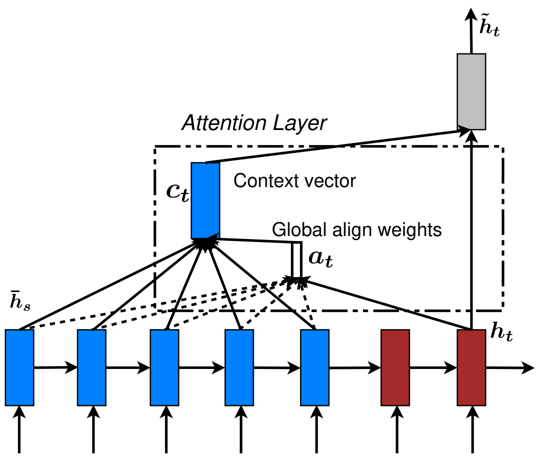
Originally introduced for the task of Neural Machine Translation but since then attention mechanism has been applied to various other tasks in different forms and with different alignment score functions. Table 2.1 shows a summary of different alignment functions (scoring) used in different attention mechanism implementations.

Table 2.1: Different alignment functions used in attention mechanism. Based on the task and data on hand one or other scoring function might be optimal. Scaled dot product attention function is used as a key component of the Transformer architecture.

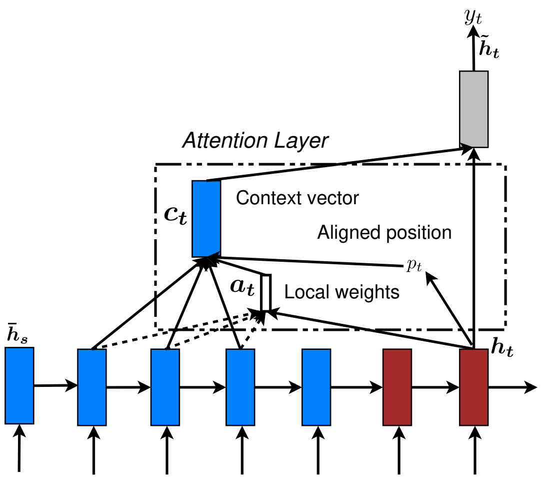
Name	**Alignment score function**	**Paper**
Content-base attention	$\text{score}(s_t,h_i)=\text{cosine}[s_t,h_i]$	Graves2014 [13]
Additive	$\text{score}(s_t,h_i)=\mathbf{v}_a^\top\tanh(\mathbf{W}_a[s_{t-1};h_i])$	Bahdanau2015 [11]
Location-Base	$\alpha_{t,i}=\text{softmax}(\mathbf{W}_a s_t)$	Luong2015 [14]
General	$\text{score}(s_t,h_i)=s_t^\top\mathbf{W}_a h_i$	Luong2015 [14]
Dot-Product	$\text{score}(s_t,h_i)=s_t^\top h_i$	Luong2015 [14]
Scaled Dot-Product	$\text{score}(s_t,h_i)=\frac{s_t^\top h_i}{\sqrt{n}}$	Vaswani2017 [15]

2.3.1 Global and Local attention

Based in whether the attention is applied on all the source positions or on selective few, attention mechanism can be classified as Global and Local [14] and depending on the task, either could be beneficial.



(a) Global attention, at each time step t model infers an alignment vector a_t based on the current target state h_t and all source states $\bar{h_s}$ and a global context vector c_t is computed as the weighted average, according to a_t over all the source states.



(b) Local attention, a single aligned position p_t for the current word at time step t is predicted, then a context vector c_t is computed which is a weighted average of the source hidden states in a window centered around position p_t. Then the weights are inferred from the current target state h_t and source states $\bar{h_s}$ in the window.

Figure 2.4: Global and Local attention mechanism (Source: Figure 2 and 3 in [14])

Global attention mechanism, also known as soft attention, is a method in which each output of the decoder is based on the entire input sequence. In this mechanism, the model learns to assign a weight to each input element, indicating its relevance to the current output. The sum of the weighted inputs is then used to compute the output of the decoder. This allows the model to attend to all input elements at each decoding step, but it can also be computationally expensive.

Local attention mechanism, on the other hand, only attends to a subset of the input sequence at each decoding step. This mechanism is used to improve efficiency, as it only considers a limited window of the input sequence, rather than the entire sequence. In this mechanism, the model learns to align the input and output sequences, and then uses a fixed-size window to attend to the relevant parts of the input sequence. Figure 2.4a and 2.4b shows how the alignment weights are inferred at each time step in global and local attention mechanism.

2.3.2 Soft and Hard attention

In [16] attention mechanism is applied to images to generate captions. The authors used a Convolutional Neural Network (CNN) to encoded to extract features from the image and then a LSTM decoder to generate captions word by word. The weights are learned through attention. The decoder generates a probability distribution over the vocabulary of possible words, conditioned on the previously generated words and the visual features. The attention mechanism is used to compute a weighted sum of the visual features, where the weights are learned by the model and depend on the previously generated words. This allows the model to dynamically focus on different parts of the image as it generates the caption. The mechanism used by the authors is soft attention, which simply means the model attends to multiple parts of the image simultaneously and provides more flexibility in selecting the relevant visual features for generating each word of the caption.

Hard attention is type of attention mechanism where the model chooses a single location in the image to attend to in each time step. The attention weights are 1 for the chosen location and 0 for the other parts in the image.

2.3.3 Self-attention

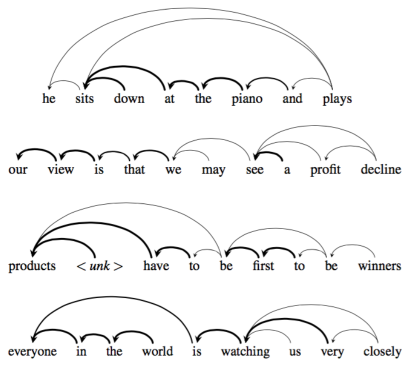
Originally introduced as intra-attention [17], self attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. This means that each word/token can have a different weight or importance, depending on how relevant it is to the other tokens in the sequence. The key advantage of self-attention mechanism is that it allows the model to selectively focus on different parts of the input sequence, rather than treating all tokens equally. This can be especially useful in situations where the relevant information for the task at hand is spread out across the sequence, rather than concentrated in one particular location.

In Figure 2.5a, bold lines between words means higher attention weights and arrow denotes the word in focus when attention is computed (not the direction of relation). The model should eventually learn the semantically important tokens for a particular token and pay more attention to those. Figure 2.5b shows how the word ‘food’ attends to other words in the sequence, the image was visualized with BertViz [1] with bert-base-uncased model.

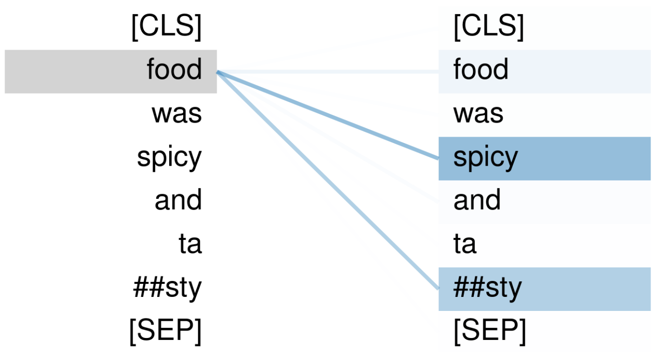
2.3.4 Multi-head attention

Multi-head attention is a variant of the self-attention mechanism used in transformers, which allows the model to jointly attend to different representation subspaces at different positions. Instead of performing a single attention function with d_{model}-dimensional keys, values and queries, the input vectors (i.e., the queries, keys, and values) are linearly projected multiple times to create multiple “heads” [15]. Then each head learns a different

[1] BertViz: Visualize Attention in NLP Models


he sits down at the piano and plays
our view is that we may see a profit decline
products < unk > have to be first to be winners
everyone in the world is watching us very closely

(a) Intra attention, Bold lines indicate higher attention and arrows denote which word is being attended to and not direction of relation (Source: Figure 4 in [17])



(b) What words does 'food' attend to. Darker color means higher attention, food has the highest attention to spicy and then to tasty. It shows the attention of transformer layer 2 of BERT model using Bertviz library. Visualized using BertViz library.

Figure 2.5: Relation between word using self attention

representation of the input vectors by using different learned projection matrices. The outputs of these heads are then concatenated and linearly transformed to produce the final output.

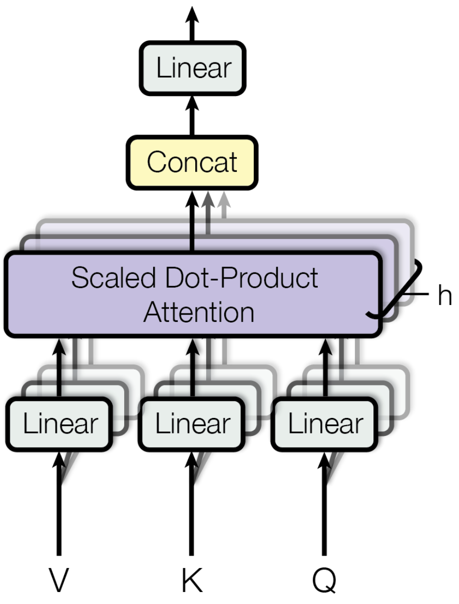


Figure 2.6: Multi head attention cell. Queries, Keys and Values are linear projected h times, multiple attention heads are run in parallel, each with its own set of learned weights. The outputs of the attention heads are then concatenated and linearly transformed to produce the final output of the multi-head attention cell. (Source: Figure 2 in [15])

Mathematically multi-head attention can be represented as:

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= [head_1; ...; head_h]\mathbf{W}^O \\ \text{where, } head_i &= \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \end{aligned} \tag{2.1}$$

Note: $\mathbf{W}_i^Q, \mathbf{W}_i^Q \text{and} \mathbf{W}_i^Q$ are parameter matrices to be learned.

The aim of multi-head attention is to enable the model to attend different aspects of the input vectors in parallel. By using multiple heads, the model can learn to focus on different information in the input vectors, which can lead to improved performance on a variety of natural language processing tasks. This mechanism is a key component of transformer models and has been shown to be very effective in a variety of natural language processing tasks, such as machine translation, language modeling, and question answering.

2.4 Transformers

Transformers introduced in the paper "Attention is All you need" presents a deep learning architecture that leverages the power self-attention mechanism [15]. It has quickly established as the leading architecture for most NLP applications. It presented a lot of improvements to the soft attention and made it possible to do seq2seq modeling without recurrent network units. The proposed "transformer" model is entirely built on the self-attention mechanisms without using sequence-aligned recurrent architecture.

In transformers, the attention mechanism is a key component that enables the model to selectively focus on different parts of the input sequence when making predictions. The transformer views the encoded representation of the input as a set of key $\mathbf{K}$ value $\mathbf{V}$ pairs, both of which are of same dimension as input sequence length (d_k). In the decoder, the previous output is compressed into a query $\mathbf{Q}$ of dimension m. Then, the next output sequence is produced by mapping this query to the set of keys and values.

The attention mechanism can be defined as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \tag{2.2}$$

where, $\mathbf{Q}$ is the query matrix with dimension m, $\mathbf{K}$ is the key matrix with dimension n and $\mathbf{V}$ is the value matrix with dimension d_k.

The dot product of $\mathbf{Q}$ and $\mathbf{K}^T$ contains the pairwise dot products between each query vector and each key vector. This matrix is then divided by $\sqrt{d_k}$, where n is the dimension of the key vectors. The resulting matrix is passed through a softmax function row-wise, which produces a matrix of attention weights.

Finally, the attention weights are used to weight the value vectors in $\mathbf{V}$, producing a weighted sum of the value vectors for each query vector. This results in a matrix of attended values which can be further processed by the transformer layers to make predictions or generate output sequences.

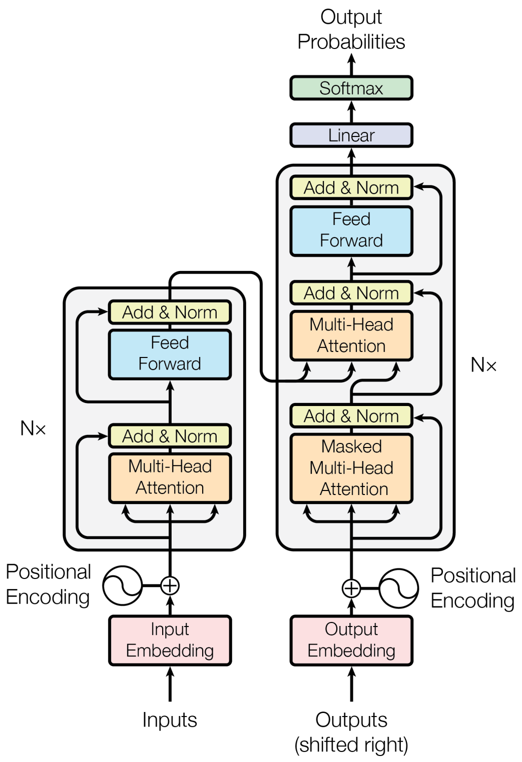


Figure 2.7: Transformer model architecture. It follows stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. Left half is the encoder layer and Right half shows the decoder. (Source: Figure 1 in [15])

The full model architecture of transformers is shown in Figure 2.7. In the original transformers architecture, there are six identical encoder layers and six identical decoder layers which both have multi head attention cells. Each multi head attention cell consists of $h(h = 8, \textit{in case of original architecture})$ parallel attention layers.

- Each encoder layer has a multi-head self-attention layer and a simple Point-wise fully connected feed-forward network.
- In an encoder, all the keys, values and queries come from the output of the previous layer in the encoder. This allows the encoder to attend to all the positions in the previous layers.
- Each decoder layer has two sub-layers of multi-head attention mechanisms and one sub-layer of fully-connected feed-forward network.
- In the decoder the first multi-head attention sublayer is a Masked which prevents it from attending to subsequent positions which disallows the model to look into the future of the target sequence when predicting the current position.

2.5 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) is a neural network architecture with multi-layer bidirectional Transformer encoder which was introduced by researchers at Google in 2018 [18]. 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. This is in contrast to traditional language models that rely on unidirectional processing and can only consider the preceding words.

BERT is pre-trained on large amounts of unlabeled textual data[2] using masked language modelling and next sentence prediction approaches and due to its architecture it develops an understanding of the language. The resulting model can then be fine-tuned on a small labelled dataset for a specific NLP task, such as question answering or sentiment analysis. As shown in Figure 2.8, The architecture used in both pre-training and fine-tuning is same except for the output layer. This allows to reuse the parameters initialized during the pre-training to be reused for down-stream tasks. During fine-tuning some special tokens are introduced to identify the beginning, ending and padding in the input.

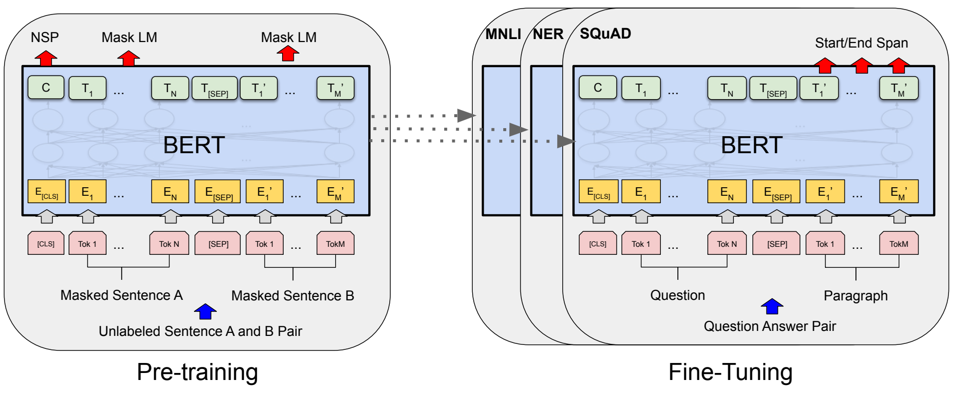


Figure 2.8: BERT training and fine-tuning procedures. Other than the output layers, same architecture are used in both procedures. Same pre-trained model parameters are used to initialize models for different down-stream tasks. (Source: Figure 1 in [18])

- **[CLS]** is a special token added in front of every input example.
- **[SEP]** is a separator token, used to signify end of one input sequence and start of another.
- **[MASK]** is used to mask the words that the model is expected to predict.

As shown in Table 2.2, originally four different variations of BERT were proposed. BERT BASE and LARGE varieties differ in size and the uncased and cased differ in the

[2] Large corpus of unannotated text from Wikipedia (2.5 billion words) and BookCorpus (800 million words)

Table 2.2: Different BERT variations proposed in original paper [18]. Pretrained models of these variants are available to download on Google's GitHub repository.

Model Type	**L**	**H**	**A**	**Total Parameters**
TINY uncased	2	128	12	4.4M
MINI uncased	4	256	12	11.3M
BASE uncased	12	768	12	110M
LARGE uncased	24	1024	16	340M
BASE cased	12	768	12	110M
LARGE cased	24	1024	16	340M

Note: **L** is the number of layers (Transformer blocks), **H** is the hidden size and **A** is the number of self-attention heads.

way they handle casing of text. Cased model takes input as both uppercase and lowercase but Uncased converts everything to lowercase. BERT uncased is better than BERT cased in most applications except in applications where case information of text is important. In recent years, smaller models (Tiny, Mini, Small and Medium) were also introduced with Tiny being smallest (at 4% the size of base model). Some of them are also available as multilingual models[3]. There are other models like Roberta, minilm, etc. which take BERT architecture as base and apply different pre-training methods usually to obtain better or smaller models.

2.6 Conditional Random Field

CRF are a type of discriminative model often applied to pattern recognition and sequence labelling tasks. Unlike other classifiers CRF takes neighbors and context into account while making the classification. The original paper defines CRF as a sequence modelling framework that has all the advantages of Maximum Entropy Markov models but also solves the label bias problem (the transition probabilities of leaving a given state is normalized for only that state) [19]. They also define CRF on observations X and random variables Y as follows:

Let $G = (V, E)$ be a graph such that $Y = (Y_v)\, v \in V$, so that Y is indexed by the vertices of G. Then (X, Y) is a conditional random field in case, when conditioned on X, the random variables Y_v obey the Markov property with respect to the graph: $p(Y_v|X, Y_w \neq v) = p\,(Y_v|X, Y_w, w \sim v)$, where $w \sim v$ means that w and v are neighbors in G.

For sequence labelling tasks like POS and NER, using masked CRF with large negative weights for illegal transitions brings significant improvement without adding computational requirements [20]. As shown on 2.9, CRF predicted a case where B-ORG is followed by I-ORG (which is an illegal transition) but with negative weights on masked CRF, the case is handled correctly.

[3] https://github.com/google-research/bert

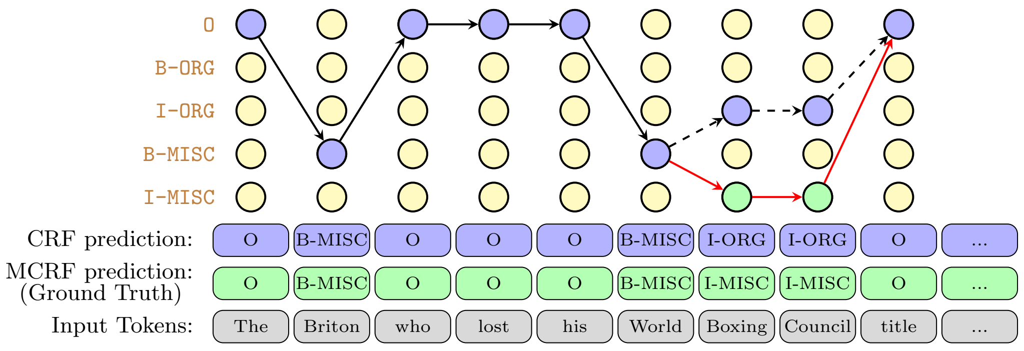
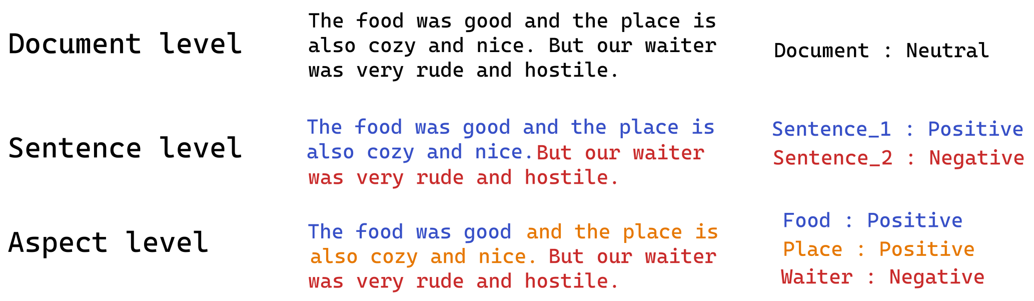

O
B-ORG
I-ORG
B-MISC
I-MISC
CRF prediction:
MCRF prediction: (Ground Truth)
Input Tokens:
O B-MISC O O O B-MISC I-ORG I-ORG O ...
O B-MISC O O O B-MISC I-MISC I-MISC O ...
The Briton who lost his World Boxing Council title ...

Figure 2.9: Example of decoded path of CRF and masked CRF. Black arrow lines represent the transitions that are predicted by both, black dashed represent ones predicted differently by base CRF and red arrow represent ones predicted differently by masked CRF. As seen in tokens prediction of ‘World’ and ‘Boxing’, base CRF predicts B-MISC then I-ORG which should never occur as I-Y cannot follow B-X but with Masked CRF it is less likely occur as the transition is masked. (Source: Figure 1 in [20])

2.7 Sentiment Analysis

Sentiment analysis or also commonly referred as opinion mining is a NLP technique used to extract subjective information such as opinion and attitudes from some form of natural language. It generally focuses on identifying the polarity which is usually classified into Positive, Neutral and Negative. Although most commonly performed on textual data, several authors have performed sentiment analysis on expression and audiovisual data [21] and so on.



Document level	The food was good and the place is also cozy and nice. But our waiter was very rude and hostile.	Document : Neutral
Sentence level	The food was good and the place is also cozy and nice.But our waiter was very rude and hostile.	Sentence_1 : Positive Sentence_2 : Negative
Aspect level	The food was good and the place is also cozy and nice. But our waiter was very rude and hostile.	Food : Positive Place : Positive Waiter : Negative

Figure 2.10: Different granularity levels of Sentiment Analysis, same sentence is analyzed at different levels in the example above.

Sentiment analysis can be classified into Document level, Sentence level and Aspect level based on granularity level [22]. Figure 2.10 shows the sentiment analysis of a sample review with different granularity levels. Document level sentiment analysis simply refers to having only one sentiment for the entire document. For example in case of movie reviews the sentiment value obtained will only show the overall sentiment towards the movie rather than to different aspects of the movie. Sentence level sentiment analysis goes a step further, and we analyze each sentence as an entity. This gives a better sentiment analysis as generally different sentences tend to talk about different ideas or parts of an entity. But both the document level and sentence level sentiment analysis fail to identify exactly what people liked and didn't like. ABSA goes to a much finer level and provides sentiment to each aspect that are present in the sentence.

2.8 Aspect Based Sentiment Analysis

ABSA (also referred as Feature based Sentiment Analysis [23]) is a granular level of sentiment analysis. ABSA allows performing sentiment analysis in a deeper manner as a sentiment is assigned to individual topics or aspects present in the document. This allows us to know the users' opinion towards something particular aspect of an entity or product. But the term itself is very broad and incorporates several subtasks like Aspect Based Term Extraction, Aspect Based Opinion Extraction, etc.

One of the major problem encountered during research for the dissertation was the inconsistency of terms in literature being used in relation with ABSA. Each entry presented in Table 2.3 uses the term ABSA to describe a specific aspect or subtask within the broader domain of sentiment analysis.

Table 2.3: Different terminologies used to describe ABSA

Terminology used to describe	Introduced by
Aspect Based Sentiment Analysis (ABSA)	He et al. [24]
Aspect Based Sentiment Classification (ABSC)	Zhang et al. [25]
Aspect Level Sentiment Classification (ALSC)	Zhang et al. [25]
Aspect Sentiment Classification (ASC)	Luo et al. [26]
Aspect Target Sentiment Classification (ATSC)	Rietzler et al. [27]
Aspect Term Sentiment Analysis (ATSA)	Xue and Li [28]
Aspect Term Sentiment Classification (ATSC)	Park et al. [29]
Target Based Sentiment Analysis (TBSA)	Li et al. [30]
Targeted (Dependent) Sentiment Analysis (TSA)	Deng and Liu [31]
Targeted Sentiment Classification (TSC)	Bai et al. [32]
Aspect Sentiment Triplet Extraction (ASTE)	Peng et al. [33]

ABSA is usually used to refer to the overall concept of analyzing sentiment at a granular level by assigning sentiment to individual aspects or topics present in a document. ABSC is used to describe the task of classifying the sentiment polarity of individual aspects or topics. ALSC refers to the same task but with a focus on the aspect level rather than the document level. ASC and ATSC is usually used to denote process of only classification of sentiment polarity specifically for aspects or aspect terms present in the text. Authors use ATSC in works which focuses on classifying the sentiment polarity of a target aspect within an entity or product. ATSA is to describe the task of extracting aspect terms from the text and assigning sentiment polarity to them. TBSA, TSA and TSC both generally refer to works that focus on sentiment analysis towards a specific targets or aspects within the text. ASTE is more unique approach where a triple consisting of an aspect term, its corresponding sentiment, and its target entity or product is extracted from the text.

These terminologies demonstrate the diversity and specificity within the field of ABSA, highlighting different aspects and subtasks researchers have explored in their work. In this work, ABSA terminology is used as a task of joint extraction of aspects and classification of sentiment.

2.8.1 Aspect, Category and Sentiment

In the literature relating to this dissertation, the actual words present in the input sentence will be used as aspect terms. Then a category based on user fed dictionary and wordnet[4] is also applied on top of the aspect terms during inference. The model uses opinion terms to identify and label sentiment of the terms, but the opinion terms are not labelled in any way. In cases where a single aspect term is longer than one word appropriate labels are used and marked as a single term of multiple words.

2.8.2 Issues and Challenges

Different steps of ABSA come with their own set if issues and challenges. Nazir et al. did a comprehensive review of issues and challenges with ABSA [25]. The first issue related with ABSA appears in the aspect extraction phase, without explicit mention of ambiguity it is very hard for ML models to identify and extract relevant aspects or features from a given text. Even if the model identifies the aspects, the meaning and sentiment can be influenced by the context and tone, this creates a problem for sentiment classification.

ABSA is usually trained on a single language and on a single domain and context, this creates a challenge for domain adaptation such as using a single model for different products, services, and industries is a huge challenge. ABSA models also face a challenge of contextual ambiguity as same word could mean multiple things. Then there comes the challenge of being able to handle multiple languages as different languages have different syntax, grammar and sentiment expression across languages.

2.8.3 Approaches

The task of ABSA can be split into three parts: Term extraction as considered in SemEval-2014 Task 4 Subtask 1, Sentiment analysis as considered in SemEval-2014 Task 4 Subtask 2 and Sentiment evolution. Term extraction means identifying the aspects or features of the product or service that are being discussed in the text, Sentiment analysis is simply classifying the opinion terms related to the aspect term as Positive, Negative or Neutral and finally sentiment evolution means continuing to monitor the evolution of

[4] https://wordnet.princeton.edu/

sentiment towards an aspect over time. The work and literature of this dissertation is limited to the first two steps.

The task of term extraction and sentiment analysis can be performed in a Pipeline, Joint or Collapsed manner. Depending on the approach different labelling scheme is needed. In a pipeline architecture the model first performs an Aspect Term Extract and then follows up with Sentiment Analysis. For each task a different system could be used. For example the task of extracting term could be done manually by human or by an algorithm and then the sentiment analysis could be performed using any ML model.

Table 2.4: Example of a Joint and Collapsed labelling on a simple sentence following BIO labelling scheme.

	The	**food**	**and**	**place**	**was**	**good**	**.**
Joint: Aspect	O	B	O	B	O	O	O
Joint: Sentiment	O	POS	O	POS	O	O	O
Collapsed	O	B-POS	O	B-POS	O	O	O

More recently, Joint and Collapsed models have become more popular. The models use a joint or collapsed labelling scheme to annotate the data (See Table 2.4 to see difference). Joint models handle both Aspect extraction and Sentiment classification simultaneously but uses separate labels for them and on the other hand collapsed models use single label with information about both Aspect and Sentiment. The model developed during this work is also able to extract both the terms and their corresponding sentiments for a given sentence using a collapsed scheme.

BIO scheme

'B' is used to represent beginning of aspect term, 'I' is used to represent inside and end on a multi-word aspect term and 'O' represents everything else that is not an aspect term.

BIOES scheme

Similar to BIO scheme but with additional letters, E represents the end on a multi-word aspect term and S represents single word aspect terms.

XML scheme

Some datasets are represented as XML tree and usually need to be converted to one of the other labelling schemes before feeding into the model. As seen in the example below, usually each sentence is put inside a sentence tag with aspect terms and position being specified in nested tags inside sentence.

```
<?xml version="1.0" encoding="utf-8"?>
```

```xml
<sentences>
<sentence>
  <text>The food was pretty good, but a little flavorless and the portions
     ↪  very small, including dessert.</text>
  <aspectTerms>
    <aspectTerm term="food" polarity="conflict" from="4" to="8"/>
    <aspectTerm term="dessert" polarity="negative" from="89" to="96"/>
    <aspectTerm term="portions" polarity="negative" from="58" to="66"/>
  </aspectTerms>
</sentence>
</sentences>
<!--An example taken from SemEval-2014 Task 4 Subtask 1+2 dataset-->
```

2.9 State of the Art

In recent years, with the growth in user generated contents like reviews and tweets, newer and newer models for ABSA are being introduced. ABSA is being used to better understand the customer/product reviews or even to automate the customer service requests. Jang et al. used topic modelling combined with aspect based sentiment analysis to understand people's reaction to COVID-19 in North America [34]. BERT based ABSA model was used by Chen et al. to investigate the change in the image of China during the COVID-19 pandemic [35]. They used a manually labeled dataset with ABSA annotations and then used a BERT based model to explore image of China. They discovered overall change of sentiment from non-negative to negative in the public but different patterns were discovered while researching different groups of U.S. Congress members, English media, and social bots. Similar kind of analysis can be also done on other scenarios like monitoring the image of political parties during elections. Zhang et al. used aspect based sentiment analysis to find weakness in products by using Chinese reviews [36] and later on Mubarok et al. Naive Bayes to perform ABSA on product reviews. As it is not possible for any manufacturer or business to comb through every review, this kind of automated analysis helps manufactures to identify flaws in their products and resolve them. Tun Thura Thet et al. proposed and evaluated a method for clause level sentiment analysis of movie reviews on discussion boards [37]. They discuss that using a finer sentiment analysis with sentiment orientation and sentiment strength of the reviewer towards various aspects of a movie helps identify the flaws present in a particular movie or even identify what the audience feels about different aspects of the movie.

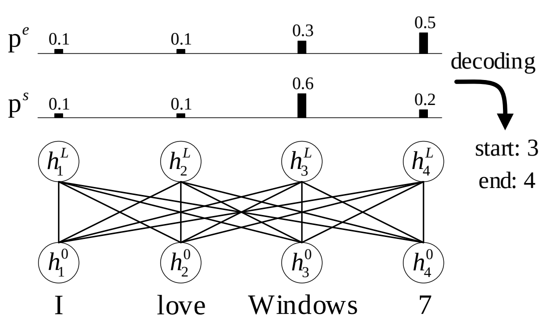
2.9.1 Pipeline Architecture

This kind of approaches perform the task of aspect extraction and classification as two-step tasks. Most of the approaches are developed as a pipeline allowing different parts to be changed based on the data available.

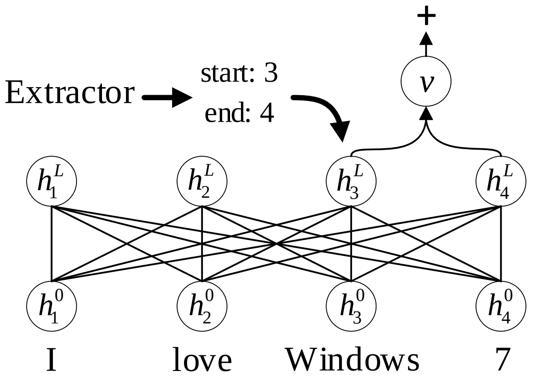
SPAN

SPAN is a model for ABSA proposed by Hu et al. that follows the idea of labeling word spans (marking their beginning and ending) rather than tagging words [38]. A pipeline model is applied on the spans which first extracts the aspect terms and then classifies their polarities based on the span information. The usage of span allows to process all the target words before predicting sentiment which would avoid label inconsistency of multi-word aspect terms.

SPAN uses pre-trained BERT as the backbone network. The specialized architecture is placed on top of the BERT model with L layers. Then [CLS] and [SEP] tokens are placed after wordpiece tokenization to create spans of aspect terms.



(a) Multi-target extractor which proposes one or multiple candidate targets based on the probabilities of the start and end positions



Extractor
start: 3
end: 4
+
v
h_1^L h_2^L h_3^L h_4^L
h_1^0 h_2^0 h_3^0 h_4^0
I
love
Windows
7

(b) Polarity classifier which predicts the sentiment polarity using the span representation of the given target

Figure 2.11: Overview of SPAN Multi-target extractor and Polarity classifier architecture (Source: Figure 3 in [38])

Instead of finding aspect terms using sequence tagging methods, SPAN finds candidate terms by predicting the start and end positions of the target in the sentence. Multi-target extractor shown in proposes multiple candidate opinion terms. The target representation from contextual vector Figure 2.11a are summarized according to its span boundary and a feed-forward neural networks to predict the sentiment polarity (polarity classifier shown in Figure 2.11b).

GRACE

Luo et al. introduced a pipeline structure model called Gradient Harmonized and Cascaded Labeling model (GRACE). The aspect term label set is $T_e = \{B, I, O\}$ and the sentiment label set T_c consists of positive, negative, neutral, conflict and other. The work was mainly focused on the impact of imbalanced dataset for ABSA. To solve the imbalance issue, they extended the gradient harmonized mechanism used in object detection to the ABSA by adjusting the weight of each label dynamically. GRACE also used BERT as its backbone and showed consistent improvements over existing methods.

2.9.2 Single Step Approaches

Rather than performing ABSA in two steps, in recent years research seems to be moving forward to more single step approaches. By using some form of joint or collapsed labelling scheme, ABSA can be performed in one step. As discussed earlier in the chapter, Joint labelling means providing two or more separate labels jointly and Collapsed labelling means using a single label that has all the information needed.

DOER

An architecture called Dual crOss-sharEd RNN framework (DOER) was proposed by Luo et al. which treated the task of ABSA as two sequence labeling problems [39] (Joint Labeling Scheme). The model generated all aspect term and sentiment pairs of the input sentence simultaneously. For each word of the input an aspect term tag out of $\{B, I, O\}$ (Begin, Inside, Outside) and a sentiment tag from the set $\{NG, PO, NT, CF, O\}$ (Negative, Positive, Neutral, Conflict, Other) were assigned. The main architecture of DOER has a dual stacked RNN, one stacked RNN for term extraction and other for sentiment classification. Even though the tasks are treated as two different, both the labels have information to imply boundary of each aspect term and also have strong correlation. To determine this interaction the architecture has a Cross Shared Unit.

BERT E2E

Li et al. proposed a framework that used contextualized embeddings from pre-trained language models like BERT and a classification head over it [40]. The overall model architecture is shown on Figure 2.12. It uses collapsed labeling scheme and predicts the tokens in single step.

BERT-E2E treats ABSA as a sequence labelling task. It used collapsed labelling scheme to represent the term and polarity of each token. For a Given input sequence $x = \{x_1, \ldots x_T\}$ of length T, L transformer layers of BERT is used to calculate the corresponding contextualized vector representations $H^L = \{h^1 \ldots h^L_T\} \in \mathbb{R}^{T \times dim_h}$ where dim_h denotes the dimension of the representation vector. The contextualized vectors are the inputs for the task specific classification layers that predict the tag sequence as $y = \{y_1, \ldots y_T\}$. The authors experiment with BERT+Linear, BERT+GRU, BERT+SAN, BERT+TFM and BERT+CRF. They were able to achieve best results with GRU(Gated recurrent unit) and SAN(Self attention Network) layers. This work was also inspired by this work and follows a similar framework.

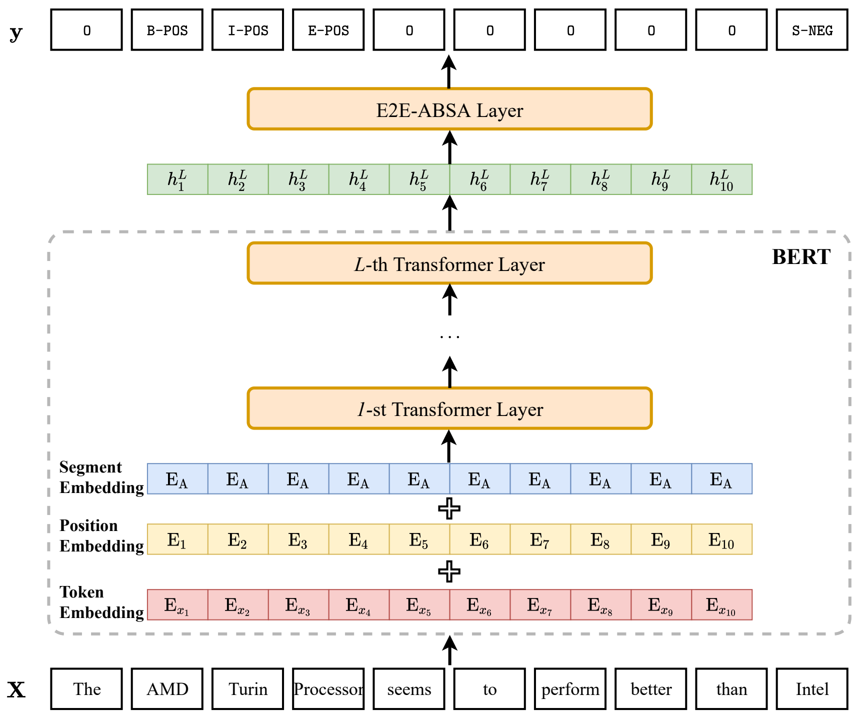

y
O
B-POS
I-POS
E-POS
O
O
O
O
O
S-NEG
E2E-ABSA Layer
h_1^L h_2^L h_3^L h_4^L h_5^L h_6^L h_7^L h_8^L h_9^L h_{10}^L
BERT
L-th Transformer Layer
...
1-st Transformer Layer
Segment Embedding
E_A
Position Embedding
E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}
Token Embedding
E_{x_1} E_{x_2} E_{x_3} E_{x_4} E_{x_5} E_{x_6} E_{x_7} E_{x_8} E_{x_9} $E_{x_{10}}$
X
The
AMD
Turin
Processor
seems
to
perform
better
than
Intel

Figure 2.12: Overview of BERT-E2E model architecture. At first input tokens are fed into BERT component to create contextualized representations. The representations are fed to task specific layers to predict the tag sequence. (Source: Figure 1 in [40])

2.9.3 Other Architectures

In recent years, there have been new ways of performing ABSA. Triplet extraction and approaches that use prompt based instructions are one of the newer approaches.

ASTE

Aspect Sentiment Triple Extraction (ASTE) is the process of extracting three important pieces of information from a given text: the target aspect, the sentiment polarity, and the reason behind the sentiment. This way solves Peng et al. propose a two-stage framework for ATSE in the paper "Knowing What, How, and Why: A Near Complete Solution for Aspect-based Sentiment Analysis" [33]. They propose a deep neural network model that jointly performs ATSE by considering the interdependencies between the three aspects. In the first stage, the model predicts what, how and why in a unified model, and then the second stage pairs up the predicted what, how and why from the first stage to output triplets. The model takes a sentence as input and outputs the target aspect, sentiment polarity, and reason for each aspect in the sentence. The model is evaluated on four benchmark datasets and achieves state-of-the-art performance on all of them. The paper suggests that considering all three aspects of ABSA is crucial for accurate and comprehensive sentiment analysis.

InstructABSA

In the recent months, with the advent of chatGPT, prompt or instruction based language models have been on the rise. Scaria et al. presented InstructABSA which used prompt based training for ABSA [33]. They use a fine-tuned Tk-Instruct-base-def-pos[5] which is an encoder-decoder Transformer model built upon pre-trained T5 trained to solve NLP tasks following in-context instructions. An example of the training prompt is shown in Figure 2.13.

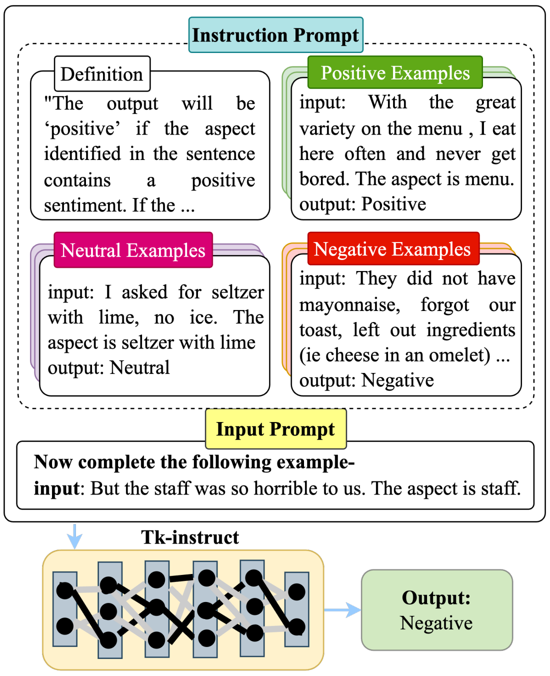

Instruction Prompt
Definition
"The output will be 'positive' if the aspect identified in the sentence contains a positive sentiment. If the ...
Positive Examples
input: With the great variety on the menu , I eat here often and never get bored. The aspect is menu.
output: Positive
Neutral Examples
input: I asked for seltzer with lime, no ice. The aspect is seltzer with lime
output: Neutral
Negative Examples
input: They did not have mayonnaise, forgot our toast, left out ingredients (ie cheese in an omelet) ...
output: Negative
Input Prompt
Now complete the following example-
input: But the staff was so horrible to us. The aspect is staff.
Tk-instruct
Output:
Negative

Figure 2.13: Formulation of InstructABSA for ATSC task. The input consists of instruction prompt and an input prompt and outputs the sentiment class. Instruction prompt consists of prompt definition and examples of positive, neutral and negative sentences. The input prompt consists of the sentence and aspect that is to be predicted. (Source: Figure 2 in [41])

With this approach the authors were able to achieve better results than prior state-of-the-art models with 1.5B parameters, InstructABSA only has 200M parameters.

[5] https://huggingface.co/allenai/tk-instruct-base-def-pos

Chapter 3

Methodology

This chapter explains the workings of different parts of the model and also goes over how the experiments are conducted and compared. It also goes over the datasets and metrics used as part of this work.

In this chapter, we describe the methodology used to develop and train our ABSA models. We used transformers library using PyTorch and Huggingface Trainer API as our foundations. The structure and code is inspired from the PyABSA[1] repository. Our goal was to build a modular model and training pipeline where any transformer based model would work. We begin by discussing the dataset used for our experiments and the preprocessing steps we took to clean and prepare the data for training. We then present the transformer based model architecture and also the training hardware and techniques used. Likewise, we also describe the hyperparameter tuning process used to optimize the model's performance. Next, we discuss the training process, including the use of pre-trained models and fine-tuning techniques to improve the model's performance. Next we discuss the experiments that we perform with the model, like freezing layers, Finally, we discuss the challenges we encountered during the development process and the solutions we used to overcome them. This chapter provides a comprehensive overview of our methodology and serves as a foundation for the experimental results and analysis presented in Chapter 4.

3.1 Datasets

We used a combination of different datasets to train and test our models. We used a combination of training dataset from SemEval-14 Restaurant and MAMS Restaurant datasets. Furthermore, we also performed inference on a self created tweet dataset (PandIA dataset) to get some real world use cases of the methodology. Detailed descriptions and the uses in our work is defined in following subsections.

[1] https://github.com/yangheng95/PyABSA

3.1.1 SemEval-14

SemEval(Sentiment Evaluation) is the most popular dataset used in ABSA task. It was introduced as part of SemEval-2014 Task 4 which consisted four different subtasks(Subtask 1: Aspect term extraction, Subtask 2: Aspect term polarity, Subtask 3: Aspect category detection, Subtask 4: Aspect category polarity). The dataset consists of manually annotated customer reviews of restaurants and laptops, as well as a common evaluation procedure [42]. This dataset is usually used as one of the primary benchmark dataset for ABSA models. We used the V2 of Restaurant training data as part of training data for this work and Restaurant test set to compare with other works. We also tested the model on laptop test set to see its performance in cross domain situations.

```
<sentence id="3161">
  <text>The Bagels have an outstanding taste with a terrific texture, both
  chewy yet not gummy.</text>
  <aspectTerms>
    <aspectTerm term="Bagels" polarity="positive" from="4" to="10"/>
  </aspectTerms>
  <aspectCategories>
    <aspectCategory category="food" polarity="positive"/>
  </aspectCategories>
</sentence>
```

The example above is taken from restaurant training set. This example contains one single aspect which is the case for majority of reviews in this dataset. Restaurant version of the dataset also has aspectCategory which is only used later on to create an aspect to category dictionary.

Table 3.1: Number of sentences and aspects in each class in SemEval-14 Dataset. Most of the reviews have single aspect or no aspects at only and only few contain multi aspect. Aspects which have both positive and negative sentiment associated are labeled as conflict and if they have neither they are labelled as neutral.

Type	**Set**	**Total Sentence**	**Aspects**			**Polarity**			
			None	**Single**	**Multi**	**Positive**	**Negative**	**Neutral**	**Conflict**
Restaurant	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)
Laptop	Train	3045	1557	1306	182	664 (987)	659 (866)	311 (460)	43 (45)
	Test	800	378	382	40	233 (341)	106 (128)	112 (169)	14 (16)

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

There were a lot of reviews without any sentiment aspects in semEval dataset (1020 sentences in Restaurant train set in and 1557 in Laptop train set). The remaining 2021

(3041 − 1020) sentences with 3693 in the Restaurant train set and 1488 (3045 − 1557) sentences with 2358 in the Laptop train 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 Restaurant train set and 43 Aspects (45S). Table 3.1 presents an overall summary of number of reviews and terms for each set.

3.1.2 MAMS

Multi-Aspect Multi-Sentiment (MAMS) was introduced as a more realistic and challenging dataset [43]. Unlike other ABSA datasets that only annotate a single aspect and sentiment for each sentence or review, MAMS annotates multiple aspects and their corresponding sentiments for each review, making it a more challenging and realistic dataset for ABSA tasks. The dataset was created by annotating Citysearch New York dataset[2] and was introduced with two different versions, one for aspect-term sentiment analysis (ATSA) and one for aspect-category sentiment analysis (ACSA). As part of our work we will only be using ATSA version of the dataset. Introduced to mainly address the limitations of semEval dataset, authors have structured the dataset in the same XML tree format as semEval dataset.

```
<sentence>
  <text>The decor is not special at all but their food and amazing prices
  make up for it.</text>
  <aspectTerms>
    <aspectTerm from="4" polarity="negative" term="decor" to="9"/>
<aspectTerm from="42" polarity="positive" term="food" to="46"/>
    <aspectTerm from="59" polarity="positive" term="prices" to="65"/>
  </aspectTerms>
</sentence>
```

The example is taken from MAMS training set and as seen it contains multiple aspects which is a defining characteristic of the dataset.

In the training set there are 4297 sentences with 11186 aspects. Most of the sentences (4173) in train set contains two aspect terms and only 124 contain more than two aspects. The distribution of different polarities is mostly balanced with bit more of aspects being marked as Neutral. Both Test set and Validation set contain 500 sentences with 1336 (Test) and 1332 (Valid) aspects. An overall summary of the dataset can be seen in Table 3.2.

[2] https://people.dbmi.columbia.edu/noemie/ursa/

Table 3.2: Number of sentences and aspects in each class in MAMS Dataset. All the reviews in MAMS contain at least two datasets and a few also contain more than two aspects. MAMS also does not have a conflict class label.

Set	**Total Sentence**	**Aspects**		**Polarity**		
		Two	**More than Two**	**Positive**	**Negative**	**Neutral**
Train	4297	4173	124	2660 (3380)	2343 (2764)	3715 (5042)
Test	500	486	14	293 (400)	279 (329)	442 (607)
Valid	500	480	20	302 (403)	286 (325)	432 (604)

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

3.1.3 PandIA

A real world use case of these models might be to track the image towards a particular aspect, for example, monitoring the sentiment towards a country or political figure during a pandemic. To try and see how the model would perform for such cases without any further fine-tuning we used a small tweet dataset which was created by us as part of PandIA[3] project [44]. This dataset was created by hand labelling a small subset (about 2000) of Portuguese tweets during Covid pandemic[4]. The tweets were hand labelled tweets with multi labelling scheme (Symptom, Covid Positive, Covid Negative, Past, Close Contact, Third Person and Undetermined) based on what the tweet is talking about. Table 3.3 shows the number of tweets in each class. A lot of the tweets in the dataset were about being positive and symptomatic, there is a large overlap (946 tweets) between those two classes. We will mainly analyze these two classes of tweets as well as part of our work. An example of an original, translated tweet and its classification can be seen below:

```
Original: Testei Positivo para COVID 19! Até o momento um pouco de
    mal estar, um pouco de dor e febre baixa, como se fosse uma
    gripe comum! Mas é isso aí, como eu digo por onde passo, isso
    é inevitável, uma hora ou outra todos vão testar positivo!
Translated:
    Tested Positive for COVID 19! So far a little unwell, a little
    pain and a low fever, as if it were a common flu! But that's it,
    as I say wherever I go, this is inevitable, at one time or
    another everyone will test positive!
Classification: Symptomatic, Covid Positive
```

We will only be analyzing the output of model to figure out what sort of aspects people were tweeting about in different categories of tweets and no training or comparative

[3] DSAIPA/AI/0088/2020

[4] https://github.com/bioinformatics-ua/Portuguese-Covid19-Dataset

Table 3.3: Summary of Pandia Tweet Dataset. Tweets are all Portuguese tweets related with COVID-19 and were hand labelled as part of PANDIA project.

Class	Number of Tweets
Close Contact	213
Covid Positive	1290
Covid Negative	69
In Past	79
Third Person	754
Symptomatic	1237
Undetermined	248
Total	1945

analysis is done on this dataset. As the original tweets were in Portuguese and all of our models were only trained on English datasets, we used DeepL API[5] to translate the tweets to English before performing any analysis.

3.1.4 Combined Dataset

As mentioned previously, we used a combination of SemEval and MAMS dataset to train our models. As seen on Table 3.4 'O' tag is present in a large number in comparison to other labels and 'I-sentiment' tags are rare. We will be using micro averaged f1 and also ignore the 'O' tag in most cases, this unbalance should not affect our metrics. Excluding the 'O' labelled tokens training set contains a total of 36244 tokens and validation set is much smaller with just 1278 tokens. We did perform minor experiments with larger validation sets, but it did not really make an impact on performance.

Table 3.4: Number of each token label in combined dataset. Dataset was created by combining semEval restaurant and MAMS dataset together and is the dataset used for training in this work.

Set	O	B-NEG	I-NEG	B-NEU	I-NEU	B-POS	I-POS
Train	259910	6598	1144	10596	3298	10228	4380
Valid	9290	234	41	359	92	385	167

3.2 Preprocessing

The original datasets (except PandIA dataset) were all available in XML format, and we wrote an XML parser to parse and convert the data into PyTorch dataset with BIO labelling scheme.

The original dataset was published as XML file which is very difficult to process and feed into a ML model. We decided to convert the XML tree to a format where every

[5] https://www.deepl.com/docs-api

```
Input: XML file
Output: Parsed XML where data is represented as a list of dictionaries
remove_list ← [‘conflict’];
sentences ← XML tree root;
data ← [ ];
foreach sentence in sentences do
    if sentence has ‘text’ then
        if lower is True then
            text ← text.lower();
        end
        aspects ← sentence.findall(‘aspectTerms’);
        terms ← [ ];
        if aspects is not None then
            foreach aspect in aspects do
                term ← individual aspect term in lower case;
                polarity ← polarity of term;
                from ← starting index of term in text;
                to ← ending index of term in text;
                if polarity not in remove_list then
                    terms.append({‘term’: text,
                                  ‘polarity’: polarity,
                                  ‘from’: from,
                                  ‘to’: to});
                end
            end
        end
        data.append({‘text’: text, ‘aspects’: terms})
    end
end
```

Algorithm 1: XML parsing algorithm

sentence is represented as a dictionary which would give us a big flexibility while processing the data at later stages. As shown in Algorithm 1, parsing starts with loading entire XML tree. Text and aspect both are converted to lowercase if needed. The aspect terms are listed as “aspectTerm” in “aspectTerms” branch in XML tree with “term”, “polarity”, “from” and “to” attributes. If there are any aspects in the sentence then the terms are extracted and stored as a list of dictionary. It is later appended to a dictionary with text.

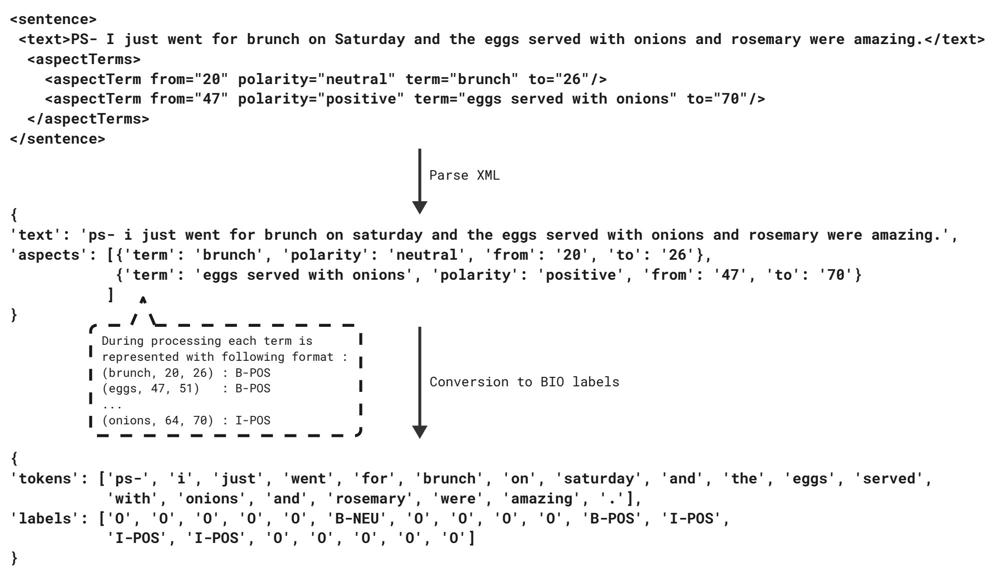
Once we have the data as a list of dictionaries with text and aspects, we wrote converted the data to BIO labelling scheme using Algorithm 2. We start by tokenizing each aspect term text to tokens using spacy[6] as there could be multi token aspects and BIO labelling scheme needs each token to have a label. We convert the existing dictionary to a format which has a tuple with (token, start, end) as key and “B-polarity” for first token and “I-polarity” for the rest in multi token aspects. Once we had a label for each token in aspect

[6] https://spacy.io/api/tokenizer

term, we then go through the text with the same tokenizer and check if the tokens tuples (token, start, end). If they exist we add the "B-polarity" or "I-polarity" for the token to labels list and "O" if it doesn't. This would result in a list of tokens and a list of labels which is then fed into the model. Figure 3.1 shows an example of the full parsing and conversion pipeline for a sample sentence with two aspects.

```
Input: data ← Parsed XML where data is represented as a list of dictionaries
Output: Data in BIO labelling format
$data\_bio \leftarrow [\,]$
foreach $item$ in $data$ do
    $bio\_terms \leftarrow dict()$;
    foreach $aspect$ in $item.aspects$ do
        $polarity \leftarrow aspect.polarity$;
        foreach $token$ in $aspect\_term\_tokens\_with\_idx$ do
            /* Tokens are generated with spacy */
            if first token then
                $bio\_terms[(token, start, end)] \leftarrow B - polarity$;
                /*Polarity is replaced with actual polarity short form (POS, NEG
                  and NEU)*/
            else
                $bio\_terms[(token, start, end)] \leftarrow I - polarity$;
            end
        end
    end
    /*Now converting to two lists, a list of tokens and a list of polarity tags*/
    $labels \leftarrow [\,]$;
    $tokens \leftarrow [\,]$;
    $text\_tokens \leftarrow tokens\_with\_idx(item.text)$;
    /*Contains in format (token, start, end), same as term token keys*/
    foreach $token$ in $text\_tokens$ do
        tokens.append(token [0]); if token in bio_terms.keys() then
            labels.append(bio_term.get(token))
        else
            labels.append("O")
        end
    end
    data.append({'tokens': tokens, 'labels': labels})
end
```

Algorithm 2: Algorithm to convert parsed XML to JSON format with BIO labelling scheme



```
<sentence>
 <text>PS- I just went for brunch on Saturday and the eggs served with onions and rosemary were amazing.</text>
  <aspectTerms>
    <aspectTerm from="20" polarity="neutral" term="brunch" to="26"/>
    <aspectTerm from="47" polarity="positive" term="eggs served with onions" to="70"/>
  </aspectTerms>
</sentence>
```

```
{
'text': 'ps- i just went for brunch on saturday and the eggs served with onions and rosemary were amazing.',
'aspects': [{'term': 'brunch', 'polarity': 'neutral', 'from': '20', 'to': '26'},
            {'term': 'eggs served with onions', 'polarity': 'positive', 'from': '47', 'to': '70'}
           ]
}
```

```
During processing each term is
represented with following format :
(brunch, 20, 26) : B-POS
(eggs, 47, 51)   : B-POS
...
(onions, 64, 70) : I-POS
```

```
{
'tokens': ['ps-', 'i', 'just', 'went', 'for', 'brunch', 'on', 'saturday', 'and', 'the', 'eggs', 'served',
           'with', 'onions', 'and', 'rosemary', 'were', 'amazing', '.'],
'labels': ['O', 'O', 'O', 'O', 'O', 'B-NEU', 'O', 'O', 'O', 'O', 'B-POS', 'I-POS',
           'I-POS', 'I-POS', 'O', 'O', 'O', 'O', 'O']
}
```

Figure 3.1: Example of XML Parsing and conversion to BIO. The pipeline starts with converting XML tagged data to a more friendly JSON format which had all the information about the text and aspect terms. Text is then tokenized and aspect terms (and their sentiment) are represented as a list with BIO labels.

3.3 Tokenizer

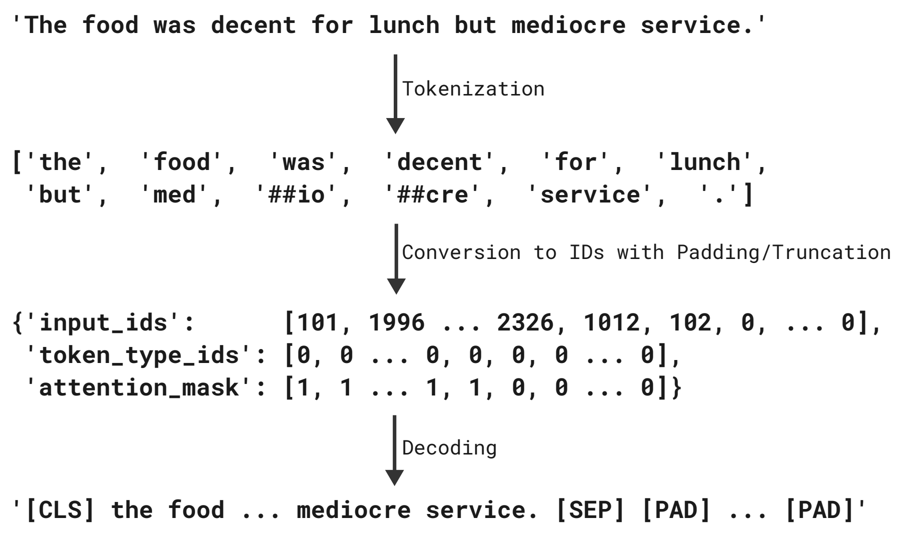
To break down the text into smaller units called tokens, a tokenizer is needed. We used a pre-trained BertTokenizer[7] from Transformers library [45]. BertTokenizer used BERT model to create tokens which are then mapped to their corresponding IDs in the BERT vocabulary. It also adds special tokens to the beginning and end of each input sequence, which are used to denote the start and end of the sequence or any padding that might be needed. Our model also supports any other transformer based tokenizer that can return input_ids, token_type_ids, attention_mask and label_ids as its output.

BERT uses a special kind of tokenization called wordpiece[8] tokenization which uses subwords as tokens. This is done in order to handle words that are not present in the tokenizer's vocabulary and to improve the generalization of the model. Traditional tokenizers simply mark out of vocabulary words as "unknown" but as wordpiece breaks words into smaller parts and subwords that make up a word are more likely to be present in the tokenizer's vocabulary than the whole word itself, it can easily handle those words even with a small dictionary.

As shown on Figure 3.2, tokenization starts by splitting sentence into tokens. The tokenizer also converts the tokens into their input ids and also adds special tokens. BERT uses [CLS] token to mark the start of the sequence, [SEP] to mark end of sequence or

[7] https://huggingface.co/docs/transformers/v4.27.2/en/model_doc/bert#transformers.BertTokenizer

[8] https://ai.googleblog.com/2021/12/a-fast-wordpiece-tokenization-system.html



```
'The food was decent for lunch but mediocre service.'

['the',  'food',  'was',  'decent',  'for',  'lunch',
 'but',  'med',  '##io',  '##cre',  'service',  '.']

{'input_ids':      [101, 1996 ... 2326, 1012, 102, 0, ... 0],
 'token_type_ids': [0, 0 ... 0, 0, 0, 0 ... 0],
 'attention_mask': [1, 1 ... 1, 1, 0, 0 ... 0]}

'[CLS] the food ... mediocre service. [SEP] [PAD] ... [PAD]'
```

Figure 3.2: BERT wordpiece tokenization example. Text is tokenized using bert tokenizer which tokenizes using wordpiece algorithm. Special tokens and padding are also added, it returns a dictionary with input_ids, token_type_ids and attention_mask. Labels from previous steps are also mapped accordingly to tokens.

separation between two different sequences and [PAD] to add padding if the sentences do not produce enough tokens. We used "longest" padding strategy which pads all the sentences in a batch to the longest sequence length in the batch. The tokenizer also adds *token_type_ids* and *attention_mask* to the data which is also fed into the model. Some words like mediocre in Figure 3.2 are split into multiple tokens (wordpiece tokenization algorithm). In our example, **mediocre** $\longrightarrow$ **[med, ##io, ##cre]**, ## symbolizes that the token is an intermediate token.

3.4 Model

Inspired by the performance of BERT based models in Named Entity Recognition tasks, we decided to try a model architecture based on Transformers with a decoder and classification head on top. We decided to use a pretrained transformer model (bert-base-uncased for most experiments) because of the performance of transformer models shown in NLP tasks. As shown in Figure 3.3, our model consists of three parts; pre-trained BERT model, Bi-LSTM and a masked CRF.

The pre-trained BERT models consists of an embedding layer, followed by a stack of twelve transformers encoder layers. Each encoder layer has a multi-head self-attention mechanism and a feedforward neural network. These encoder layers are designed to capture the contextual relationships between words in a sentence. We use pre-trained models from Huggingface[9] hub which has a lot of variants of pre-trained models openly available. Some

[9] https://huggingface.co

models are optimized for larger sequences, some reduce memory consumption and some compute attention mechanism in different ways. There are also variations based on the data the models were trained on or the task they are trained for. This modular approach allows us to use a different model based on the input data and our aim. In section 3.8.2, we describe how we experiment with different pre-trained models with different parameter sizes to see the impact they have on performance.

We decided to use LSTM module from torch.nn package as it has a robust implementation and also allows us to use it as a bidirectional LSTM. The LSTM layer expects same number of features as the hidden size of the BERT layer (input_size = config.hidden_-size) and has half number of features as hidden size. Then, the next Linear classifier layer reduces it down to number of classes.

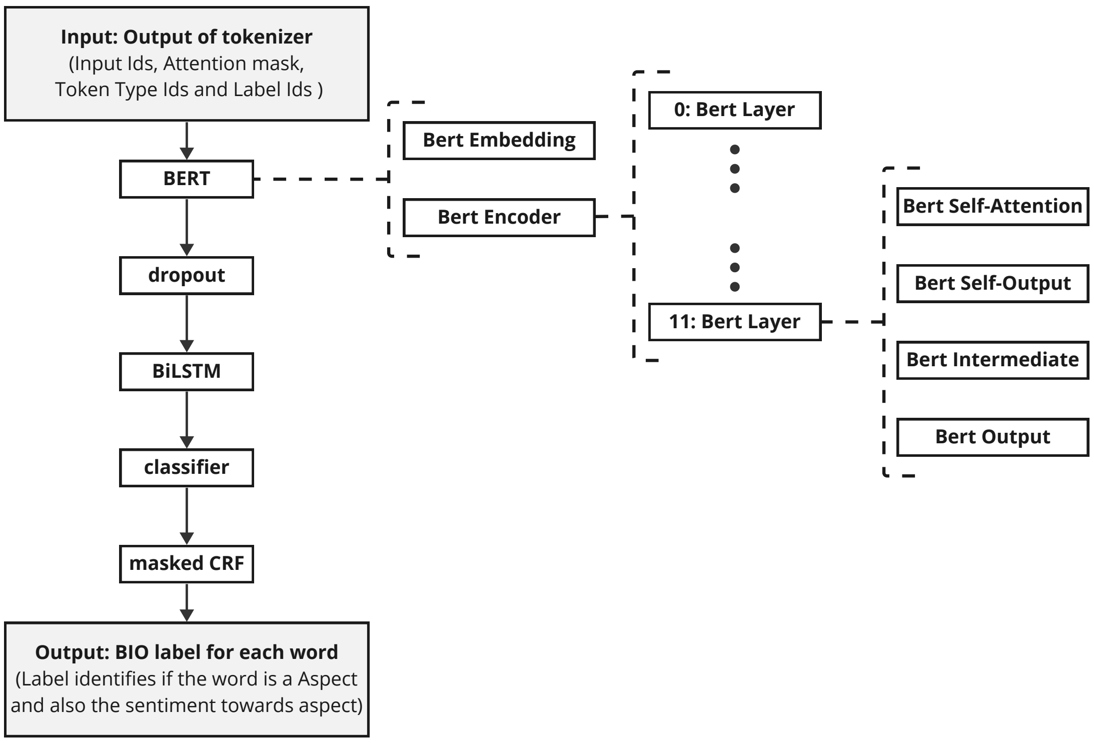

Input: Output of tokenizer
(Input Ids, Attention mask, Token Type Ids and Label Ids)
BERT
dropout
BiLSTM
classifier
masked CRF
Output: BIO label for each word
(Label identifies if the word is a Aspect and also the sentiment towards aspect)
Bert Embedding
Bert Encoder
0: Bert Layer
11: Bert Layer
Bert Self-Attention
Bert Self-Output
Bert Intermediate
Bert Output

Figure 3.3: A summary of base model architecture. Arrows represent the flow of data and dashed lines indicated the composition of a particular cell. Pre-trained bert-base-uncased is used on the base model. Other parts of the model are kept constant in all the experiments.

Our implementation was inspired from PyTorch-crf[10] and allenlp CRF module[11]. As we use BIO labelling scheme to label our data. Under this labelling scheme the model should only predict an I-polarity tag after a B-polarity tag. Furthermore, a start token of a sequence can only be B-polarity. There have been implementations where I-polarity

[10] https://github.com/kmkurn/pytorch-crf

[11] https://github.com/allenai/allennlp/blob/master/allennlp/modules/conditional_random_field.py

is simply replaced with B-polarity if a preceding O tag is present. Rather than using this naive approach, We decide to use large negative number as masks inspired by the work of Almeida et al. [46].

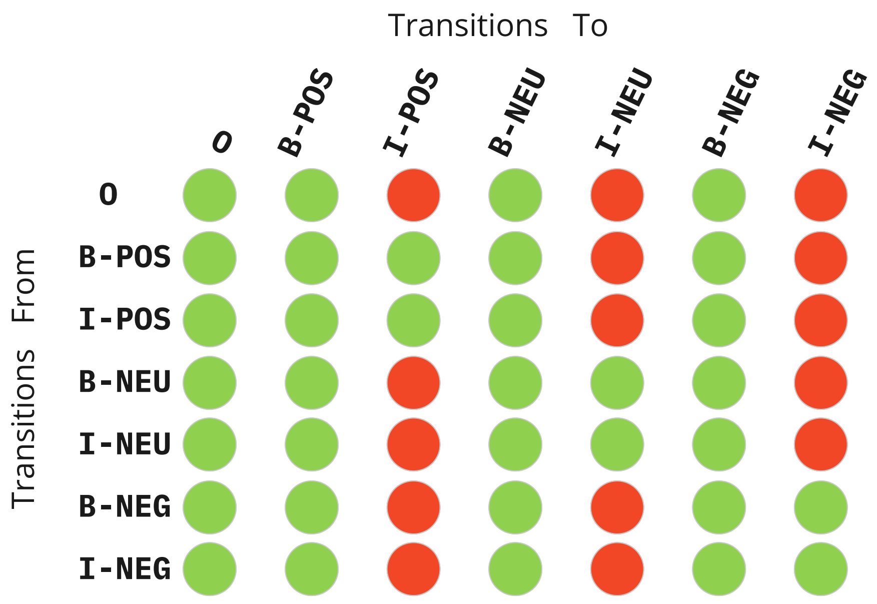


Figure 3.4: Transitions for Masked CRF. Red dot means the transition from the label on corresponding row to label on corresponding column is not possible and green means it is allowed. For impossible transitions are set to large negative number and so the change of such transitions occurring is low but non-zero.

To implement masked transitions in CRF, we introduce a transition mask matrix that defines the allowed and forbidden transitions between different labels. The transition mask matrix is a square matrix of size n x n, where n is the number of labels. The weights of the transition mask matrix are first initialized randomly from a uniform distribution between -0.1 and 0.1. Then the $(i, j)^{th}$ entry of the matrix is set to $-1e9$ if the transition from label i to label j is not allowed, and otherwise is left at the initialized value. Figure 3.4 shows a 7×7 matrix where red dots represent illegal transitions whose weights are set to the large negative number ($-1e9$). For the start transitions we set the weights of 'I-POS', 'I-NEG' and 'I-NEU' to the same large negative number. With this large transition in place the of these transitions occurring is very low but not impossible.

3.5 Training

Once we had the model architecture as defined in section 3.4, we created our trainer taking Huggingface trainer as a basis. Using Huggingface Trainer allowed us to easily implement distributed training on multiple GPUs, and to integrate multiple different services into our training pipeline.

Before conducting any experiments with the model, we needed a baseline run to compare to. We used a lot of defaults from transformer and some values based on our intuition for the base runs. We trained the run for 10 epochs with training batch size of 32. A summary of hyperparameters used are described in 3.5. We will discuss the results and how the base runs compares with other runs in chapter 4.

Table 3.5: Base Hyperparameters used for training the model. Hyperparameters are optimized later on and updated accordingly.

Parameter	**Value**
Optimizer	AdamW with $\beta_1 = 0.9$ and $\beta_2 = 0.999$
Number of frozen layer	0
Seed	109806
Learning rate	1e−5
Number of Training epochs	6
Training Batch size	16
Weight decay	0.1
Warmup ratio	0.1

Then, using Optuna we performed hyperparameter optimization which is described in detail in next section. Once we had optimized parameters we conducted several experimental runs which are described in section 3.8. In order to track all of our experiments, we used Weights & Biases[12] which is a machine learning experiment tracking and management tool that allows to log, track and share machine learning experiments. It also allows us to visualize and compare different runs. We used trainer callback to integrate W&B to our project.

All the configurations were run on a single GPU node with two Nvidia A40s which were provided by High Performance Cluster from University of Coimbra through PandIA project. In order to run the models in the HPC, we allocated the required hardware and then used a script to submit the batch job to the node. The basic hardware configuration of the instance was chosen as following:

```
Nodes : 1
Number of CPUs : 8
Memory per CPU : 8G
Gres Configuration : GPU:a40:1 (One Nvidia a40 GPU)
```

The configuration of an a40 GPU allowed for faster training of models used in the experiments. Most of the CPUs were used as Dataloaders, and they provided ample processing power to support the GPU computation. Overall, the hardware configuration provided a powerful and efficient platform for training our ABSA models and allowed us to run multiple experiments in short amount of time.

[12] https://wandb.ai

3.6 Hyperparameter Optimization

Once our base model was trained we used optuna to tune the hyperparameters of the model. Optuna is a hyperparameter optimization tool that allows us to define search space in a pythonic way and also allows us to use several algorithms for searching and pruning the runs [47]. As part of this work, taking into consideration the recommendations made in optuna documentation[13], we decided to use Tree-structured Parzen Estimator (TPE) algorithm as sampler and Hyperband algorithm as pruner.

TPE sampler works by creating two Gaussian Mixture Model on each trial, first model $l(x)$ is fitted to set of parameter values associated with the best objective values and second model $g(x)$ to the remaining parameter values. By iteratively refining the understanding of the best parameters and maximizing the ratio of $l(x)/g(x)$, the TPE algorithm converges on a set of values that produce good results. Once a parameter for a trial are selected by the sampler, the model is trained for number of steps with the parameters. We introduce early stopping with Hyperband algorithm to speed up the optimization process. Hyperband pruner uses multiple Successive Halving Pruner, and they require a fixed number of configuration n as input. For a finite budget[14] B, B/n resources are allocated to each run, larger number of n would result in low amount of resources to each run which might not be enough to train a particular configuration of run and low n would result in not all possible configurations being run. Hyperband handles this trade-off between B and B/n by trying different values of n for a fixed budget [48]. Once n runs consume the given resources, the worst-performing half of the models are then discarded, and the remaining models are trained with additional resources. This process is repeated several times, with the amount of resources increasing at each iteration for the remaining models. This allows Hyperband pruning to allocate more computational resources to the most promising models while discarding the less promising ones early. In our work, we ran a total of 500 optuna runs, the results of which are described on Section 4.1.

Table 3.6: Search space used for hyperparameter optimization. Optuna searches for optimal hyperparameters by creating different combinations within the search space.

Parameter	**Search Space**
Learning rate	Log uniform distribution; [1e-6, 1e-3]
Number of Training epochs	Uniform distribution; [1, 10]
Training Batch size	[4, 8, 16, 32, 64, 128]
Weight decay	Uniform distribution with 0.01 step; [0.0, 0.20]
Warmup ratio	Uniform distribution with 0.01 step; [0.0, 0.20]

Following our intuition and the work of Sun et al. [49], we tried to optimize Learning

[13] https://optuna.readthedocs.io/en/stable/tutorial/10_key_features/003_efficient_optimization_algorithms.html
[14] Budget can be any resource like Time, Number of Epochs, etc.

rate, Number of Training epochs, Training Batch size, Weight decay and Warmup ratio. A summary of the search space used is shown in table 3.6. For each run, optuna selects a random value from the defined search for each hyperparameter. In our work, we ran a total of 500 optuna runs, the results of which are described on Section 4.1. We also tried using different optimizer with the base configuration and found AdamW had better performance, and so we decided to use if for all the other runs as well. Other factors like model sizes and freezing of layers are further explored in section 3.8.

3.7 Metrics

The performance evaluation of our models is crucial to understand their effectiveness in ABSA. One way to assess the model's performance is by measuring how accurately it predicts the token positions in the given text. Another metric is multilabel classification, where the model predicts the sentiment polarity in the sentence even if it fails to predict the exact token position. To measure these metrics, various evaluation packages and libraries are available. This section introduces some evaluation metrics and tools used as part of this work.

3.7.1 Token Position Quality

One way to measure performance of the model is to see how accurately model is predicting the token positions. Huggingface metrics has seqeval[15] package for sequence labeling evaluation. Seqeval provides us with accuracy, precision, recall and f1 for each class (Positive, Negative and Neutral) and also overall scores. We use these scores (primarily f1) to measure the token position quality of our predictions.

3.7.2 Multilabel Classification

In some cases, the model might not successfully predict the exact token position but successfully predict the sentiment polarity present in the sentence. To measure these cases, we use the model to also compare the multilabel classification performance. As part of this work, we used f1 score (micro and per class) and exact match Accuracy to compare the results of different experiment. Using micro f1 lets us compare the results even in cases where the classes are imbalanced and individual f1 score helps us understand which classes the model is struggling on. For some experiments we will be using some more metrics like hamming accuracy, overlap accuracy and confusion matrix to get some more insights into the model. We used torchmetrics[16] library which had implementations all these metrics.

[15] https://huggingface.co/spaces/evaluate-metric/seqeval

[16] https://torchmetrics.readthedocs.io/en/stable/all-metrics.html

3.8 Experiments

To test the performance of out model architecture, we decided to conduct several experiments. The experiments are designed to observe any improvement in the performance of models in terms of score, time or storage space. We use the optimized hyperparameters we found from our tuning and then use different models and datasets to perform these experimental runs. All of our experiments can be browsed in wandb project report[17].

3.8.1 Freezing Transformer Layers

Transformers have grown in size in recent years. Starting from the original transformer architecture introduced in 2017, subsequent models based on BERT, GPT, and T5 have increased in size, with hundreds of millions or even billions of parameters. While these models have achieved impressive performance on a wide range of NLP tasks, their large size can make them challenging to train and deploy in real-world applications. One technique for addressing this challenge is to freeze layers in the model during fine-tuning.

Freezing layers disables gradient computation and backpropagation for the weights of these layers. It is a common technique used in NLP and Computer Vision when working with small datasets. It usually reduces the chances of your model to overfit and also allows for faster convergence. Not having to compute gradients for some layers also means we gain a speed boost and reduced memory usage during training process. 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, we can adapt the pre-trained model to a downstream NLP task while preserving the more general features learned during pre-training.

In our case, we start with pre-trained weights and by freezing layers we expect to gain more space and time efficiency. We experimented with just freezing the embedding layer of BERT, freezing $1, 2, 4, 8, 10, 12$ BERT encoder layers. We use two different flags in our YAML config file, **freeze** to control overall freezing and freezing of embedding layers and **num_frozen_layers** controls the number of encoder layers to freeze.

3.8.2 Model Sizes

We wanted to see if we could achieve similar results as BERT base with smaller model or could we improve our accuracy using a larger model. Using a smaller model leads to less computation resulting in faster training times and sometimes also can lead to better performance on simpler tasks but typical have lower overall performance on complex tasks. For such complex tasks using larger model might be beneficial. The main advantage of larger models is their ability to capture more complex patterns and relationships in the data, which can lead to state-of-the-art performance on a wide range of tasks. Larger

[17] https://api.wandb.ai/links/gundruke/xpy5fckj

models can also learn more nuanced and subtle features that may be missed by smaller models, which can be particularly important for tasks such as image captioning, machine translation, or sentiment analysis. But they need large amount of data and computational resources which can make them difficult and expensive to train.

Table 3.7: Size of different transformer models used in this work. #Layers represent the number of layers in the model, #Hidden is the hidden size and #Parameters is the total learnable parameters of the model. In theory, a larger model should be able to better capture the contextual information as it has more learnable parameters.

Model name	#Layers	#Hidden	#Parameters
bert-tiny-uncased	2	128	4.4M
bert-mini-uncased	4	256	11.3
bert-base-uncased	12	768	110M
bert-large-uncased	24	1024	336M
miniLM-L12	12	384	21M
roberta-base	12	768	125M

As we have chosen bert-base-uncased as our base model, we have selected smaller models (smaller BERT variants and minilm[18]) and larger models (bert-large-uncased[19] and Roberta[20]) to experiment with. Table 3.7 shows the size of different models used as part of our experiments.

3.8.3 Cross Domain

Even though we only trained the model on Restaurant version of the datasets we wanted to see how it would perform on other domains. We decided to use laptop version of the same semEval dataset which was annotated with same rules. Testing the model on Laptop dataset would give us idea about how well the model generalizes to other domains and can help identify any weaknesses in the model's ability to handle new and different language patterns.

We also performed a simple inference test on the PandIA dataset to see how it would perform on a completely different setting (tweets about Covid). As we did not have a gold standard test set available on hand, we couldn't calculate any form of scores, but we perform an analysis on the aspects and their sentiments captured by the model.

3.9 Category classification

From a real world perspective, ABSA is much more useful if it can be generalized into simpler categories. So, we wanted to add some form of categorization of aspects as part

[18] https://huggingface.co/microsoft/Multilingual-MiniLM-L12-H384
[19] https://huggingface.co/bert-large-uncased
[20] https://huggingface.co/roberta-base

of inference process as it helps to get better idea about sentiment of public towards a particular category. We decided to use very simple approach with a predefined dictionary and NLTK wordnet[21]. The predefined dictionary was created by using the aspect category available in the semEval 14 restaurant dataset and also manual intuition. For restaurant dataset, we chose to limit out categorization to Food, Drinks, Service, Atmosphere and Others. Others category is meant to be a general category capturing all the aspects that were not classified. The dictionary contains about 800 words and categories. A sample of dictionary for restaurant dataset is as follows:

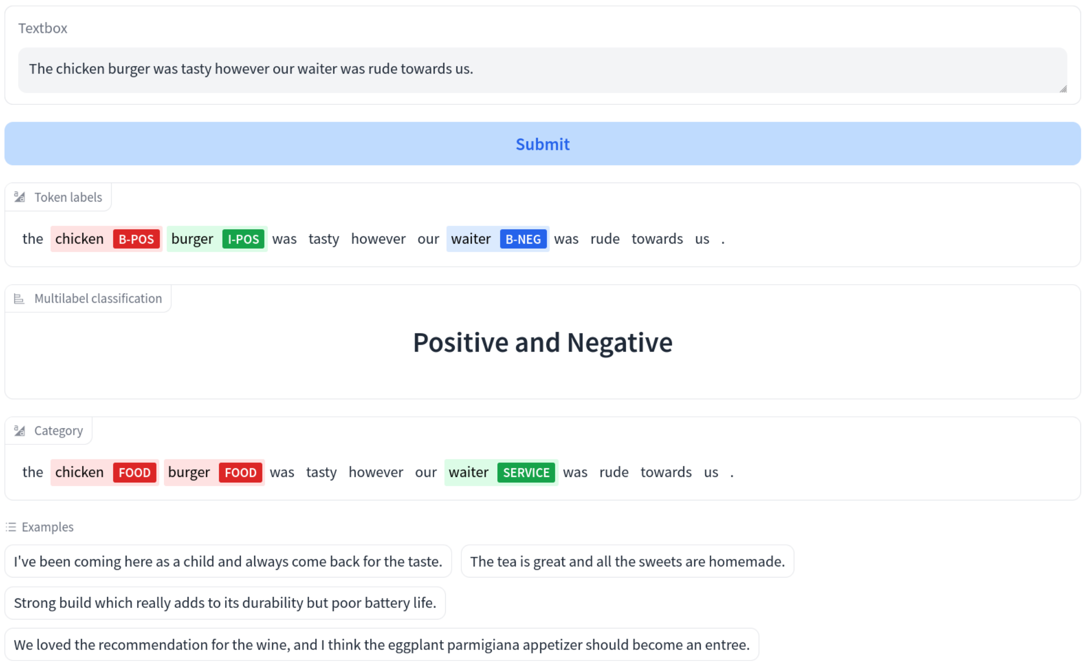
```
{
"asian": "food",
"asparagus": "food",
"attitude": "service",
"back": "other",
"bar": "atmosphere",
"bartender": "service",
"beef": "food",
"beer": "drinks",
"big": "other",
"bill": "price",
...
}
```

Another simpler dictionary was created for PandIA dataset which had about 100 words. This dictionary had to be created just using intuition as there was no predefined categorization provided.

If the word is present in the dictionary, its corresponding value is used else it uses WordNet's synsets and hypernyms to find a more general term for a given word. If the general term is in our scope, it is used and if neither condition is met, the word is classified as "other". Once the model identifies aspect terms and associated sentiment, it is fed into the categorization system. One problem with this approach to categorization is one word could belong to multiple different categories depending on the context and this approach does not have any contextual information. Also, a lot of times hypernyms are not generic enough, so the aspect just gets categorized as others.

One of the major plus points of using Huggingface models is the ability to easily deploy them on Huggingface hub for inference. We created a simple gradio interface on a Huggingface space that creates a model with all the provided parameters and then loads the trained model weights. It then performs inference on the input sentence. The space can be browsed at gundruke/ua-thesis-absa[22]. An Example of gradio UI can be seen in Figure 3.5.

[21] https://www.nltk.org/howto/wordnet.html
[22] https://huggingface.co/spaces/gundruke/ua-thesis-absa


Textbox

The chicken burger was tasty however our waiter was rude towards us.

Submit

Token labels

the chicken B-POS burger I-POS was tasty however our waiter B-NEG was rude towards us .

Multilabel classification

Positive and Negative

Category

the chicken FOOD burger FOOD was tasty however our waiter SERVICE was rude towards us .

Examples

I've been coming here as a child and always come back for the taste.

The tea is great and all the sweets are homemade.

Strong build which really adds to its durability but poor battery life.

We loved the recommendation for the wine, and I think the eggplant parmigiana appetizer should become an entree.

Figure 3.5: User Interface for classification. Once a text is input, first tokenizer is loaded and text is tokenized. Then the model is loaded from huggingface hub and tokenized text is passed to generate token labels and multilabel classes. Finally, category labels are added where necessary. In the token labels, no label means token is labelled as ‘O’.

One other advantage of having model on Huggingface hub is, for future use cases the model can be simply loaded by importing transformers library and then fetching the pre-trained file from gundruke/bert-lstm-crf-absa. It also allows for hosted inference on the hub or through API.

Chapter 4

Results

This chapter describes the different results obtained. It also compares the results of one experiment with other to see the effectiveness of different approaches.

We will describe in detail the results of different datasets and also discuss some reasons behind the performance impacts. We will mainly be using performance metrics but also use training time and computational resource requirements where relevant.

4.1 Baseline

For our baseline run, we ran our model with the hyperparameters discussed in Table 3.5. The model performed really well in terms of multilabel classification but wasn't as good in terms of token quality metrics.

Table 4.1: Results of our first run. The hyperparameters described before as base params are used for this run and are not optimal. The results of this run are expected to improve with more optimal parameters.

	Token Quality					Multilabel					
	f1 (NEG)	f1 (POS)	f1 (NEU)	f1 (micro)	accuracy (overall)	f1 (NEG)	f1 (POS)	f1 (NEU)	f1 (micro)	accuracy (overlap)	accuracy (exact)
semeval test (Restaurant)	0.6842	0.7966	0.5024	0.7224	0.9839	0.7609	0.9192	0.6383	0.8306	0.8893	0.7850
MAMS test	0.6829	0.6000	0.6214	0.6303	0.9659	0.8545	0.8662	0.8905	0.8731	0.9900	0.6240
val set	0.6992	0.6391	0.5913	0.6353	0.9758	0.8601	0.8511	0.8351	0.8474	0.9200	0.6700

In terms of token quality metrics, the model performed best on semEval restaurant test set with a micro averaged f1 score of 0.7224, but it shows but worse performance (micro f1 of 0.6303) even though MAMS test set is also in of the same domain (restaurant) reviews. The accuracy scores are much higher as it also takes into account predictions of all the tokens. Looking at individual classes, the model performed excellently on positive and negative aspects but struggles with neutral aspects with a difference of 0.2942 in f1 score between neutral and positive aspects in semEval test set. In case of validation set the performance on all three types of aspects is comparable.

The scores in multilabel classification are higher which is expected as the number and exact position of tokens do not matter. In terms of f1 score per class, neutral class on semeval test set has a lower score compared to positive and negative class (0.6383 compared to 0.9192 and 0.1609). But in MAMS dataset similar to token quality metrics, multilabel f1 scores for all class are comparable with a range of 0.036. Multilabel overlap accuracy shows the amount of predictions with at least one correct label predicted. MAMS dataset had 99.0% accuracy in terms of overlap but only 62.40% in terms of exact which suggests model usually misses (or predicts wrong) some aspects whenever multiple are present. This difference in overlap and exact accuracy is lower in semeval dataset as more than half of the dataset contains only one aspect per review. As you can see in example below the model predicts the tokens in such way, it's fully correct for multilabel classification (both exact and overlap) but is not for token quality metrics as it predicts multiple B-NEU tokens as O. All the results of base run are summarized in Table 4.1. Even tough we observed that the model performs poorly on certain cases, base runs provide a useful benchmark for evaluating the performance of other experimental runs.

```
tokens  breakfast   ,  lunch  or  dinner   ...  each   and   every  time  .
true        O       O    O    O   B-NEU    ...   O    B-POS    O     O    O
pred      B-NEU     O  B-NEU  O   B-NEU    ...   O    B-POS    O     O    O
```

Even though we have a baseline established, some additional performance can be gained by using more optimal parameters. In practice, the optimal parameters for a Transformer based model depends on various factors, such as the size of the model, the size of the dataset, and the available hardware resources. It is often necessary to experiment with different batch sizes to find the optimal value for a particular task and hardware setup. To find the optimal parameters for our use case, we performed an optuna study with 500 runs to find optimal hyperparameters for our case using methodology previously discussed (see section 3.6) with micro f1 score of val set as objective value.

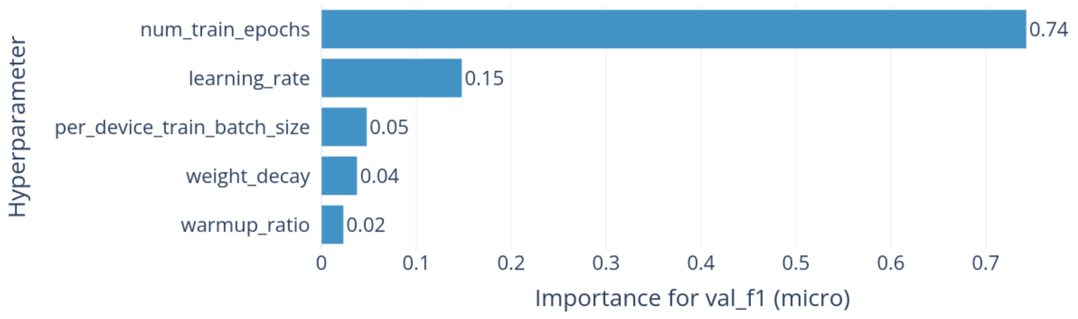


Figure 4.1: Importance of different hyperparameters on determining f1 score (token quality f1 score on semeval restaurant test set) during hyperparameter search. The plot was generated using the data from optuna study.

Different hyperparameters play different role and have different importance in determining the objective value. As we can see in Figure 4.1, number of training epochs has the biggest importance followed by learning rate and train batch size. Number of training epochs having the biggest impact makes sense as it directly affects how much the model learns from the training data, and too few or too many epochs can result in underfitting or overfitting, respectively. The second most important hyperparameter is learning rate which determines the step size for the optimizer during training and can greatly affect the speed and quality of the model's convergence. Our study suggests train batch size is relatively less important than the previous two hyperparameters. However, this parameter can still significantly affect the model's training performance by determining how many training examples are processed at once on each device. The weight decay and warmup ratio parameters have relatively lower importance scores of 0.04 and 0.02, respectively. Weight decay is used to prevent overfitting by adding a penalty term to the loss function, while warmup ratio controls how much of the training time is used for learning the model versus warming up the optimizer. These parameters are still important but have a smaller impact on the model's performance compared to the other hyperparameters.

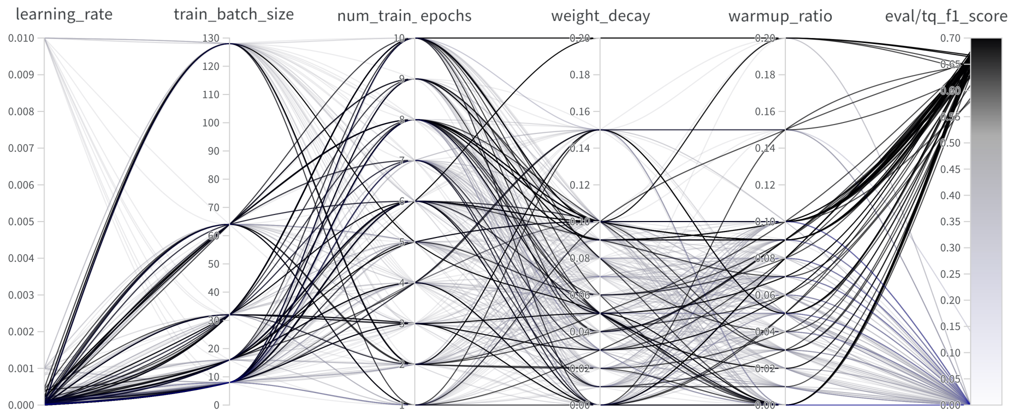


Figure 4.2: Coordinate plot showing different hyperparameter runs. Each line represents a combination of hyperparameters used. Darker shades represent higher f1 score. Token quality f1 score on semeval restaurant test set is used.

The coordinate plot of our study shown in Figure 4.2 displays the performance of our model on the validation dataset for various combination of our hyperparameters. Most of the runs that have high f1 score start with small learning rate as a learning rate higher than 0.001 leads to unstable convergence, but too small of a learning rate would need too many iterations to converge to the best values. Almost all the runs with training epochs higher than 8 have achieved f1 score higher than 0.60. Increasing number of epoch does not always help as the f1 score begins to decrease due to overfitting with very large number of epochs.

We found that the hyperparameters presented in Table 4.2 results in the highest f1 score. For all the experiments, this set of hyperparameters would be used. We updated our hyperparameters the ones found from our optimization study and re-ran the model training on same dataset and hardware configuration, the results of which are summarized in Table 4.3. In further experiments, the results will be compared to this run (referred as

Table 4.2: Optimal hyperparameters as obtained by Optuna study. These are the parameters that are used in all the runs unless specified otherwise.

Parameter	**Value**
Seed	109806
Learning rate	1e−4
Number of Training epochs	10
Training Batch size	32
Weight decay	0.1
Warmup ratio	0.2

baseline run).

Table 4.3: Results of model training with optimal parameters. Only change made in this run is that optimal hyperparameters used and architecture of the model is not altered in any way. Results are only slightly better than previous run and this run results are used as Baseline.

	Token Quality					Multilabel					
	f1 (NEG)	f1 (POS)	f1 (NEU)	f1 (micro)	accuracy (overall)	f1 (NEG)	f1 (POS)	f1 (NEU)	f1 (micro)	accuracy (overlap)	accuracy (exact)
semeval test (Restaurant)	0.6851	0.8033	0.4588	0.7225	0.9849	0.7475	0.8962	0.5825	0.8028	0.8690	0.7532
MAMS test	0.7198	0.6210	0.6143	0.6430	0.9667	0.8854	0.9147	0.8965	0.8987	0.9980	0.6900
val set	0.7073	0.6791	0.6276	0.6676	0.9776	0.8578	0.8801	0.8370	0.8581	0.9220	0.6900

When comparing the results to the base run, there is some amount in terms of token quality but in some cases there is minor loss in performance when it comes to multilabel metrics. Since the model parameters were tuned for token quality f1 score, it makes sense that the model has aligned more in gaining performance in that regard. The biggest difference in terms of f1 score (token quality) is 0.0369, seen in negative class. Overall f1 score of validation set has only increased by 0.0323 (token quality) and 0.0107 (multilabel). There might be a bit more performance that could be achieved by performing a more extensive study but the results we have sets a reasonable baseline for comparison with other experiments. If we wanted to find a combination of parameters that achieves better multilabel metrics, we can perform a study setting it as objective value or even a combination of both metrics can be used if needed.

4.2 Freezing of layers

Transformer based models require a huge amount of data and computational requirements for full training. For most of the tasks, using a pretrained model with some encoder layers frozen is beneficial. As the number of layers that needs to be frozen depends on the task and the dataset being used, we experimented with a various number of layers frozen. We expect some gain in performance as our dataset is not very large and maybe not enough to train the full model.

Table 4.4: Token Quality results of experiments related with freezing of layer. Only token quality metrics on semEval restaurant dataset are shown as multilabel metrics follow the similar trends to token quality and also only differ marginally on these experiments.

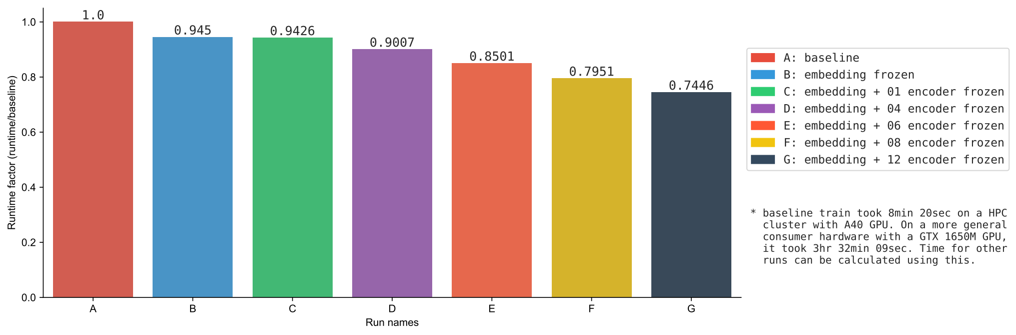
Frozen layers		Token Quality				
embedding	encoder layers	f1 (NEG)	f1 (POS)	f1 (NEU)	f1 (micro)	accuracy (overall)
True	00	0.6885	0.7830	0.3484	0.7108	0.9838
True	01	**0.6905**	0.7934	0.4409	0.7254	**0.9848**
True	**04**	0.6835	**0.8035**	0.4566	**0.7331**	**0.9848**
True	06	0.6804	0.7914	0.4354	0.7232	0.9841
True	08	0.6733	0.7783	0.4155	0.7105	0.9830
True	12	0.6611	0.7573	**0.5279**	0.6984	0.9816
False	00	0.6851	0.8033	0.4588	0.7225	0.9849

We will be using the SemEval Restaurant test set (see Table 4.4) to compare the results with baseline. Freezing just the embedding layer actually had a counter impact on the metric and the overall micro f1 score for token quality dropped by 0.0223. The overall accuracy also dropped slightly by 0.001. With just embedding layer being frozen, we did not see any improvement in any metrics in terms of token quality. The results for runs with frozen embedding layer and 2 or 4 or 6 encoder layers frozen were slightly better than baseline. The best results were shown by having 4 encoder layers frozen in terms of overall f1 and accuracy score. The results start to deteriorate when we start freezing more than 6 layers. The run where the entire BERT model was frozen (embedding + 12 encoder layers) and only the classifier head was trained showed the worse performance in these group of runs. But even the worse run had a micro f1 score of 0.6984 which might be acceptable considering the lower computational requirements.

Table 4.5: Complete result of best performing run among layer freezing experiments. The optimal result was obtained by run with frozen embedding layer and 04 encoder layers. This run is also used as representative run for freezing experiment when comparing with other experiment types.

	Token Quality					Multilabel					
	f1 (NEG)	f1 (POS)	f1 (NEU)	f1 (micro)	accuracy (overall)	f1 (NEG)	f1 (POS)	f1 (NEU)	f1 (micro)	accuracy (overlap)	accuracy (exact)
semeval test (Restaurant)	0.6835	0.8035	0.4566	0.7331	0.9848	0.7762	0.9036	0.5815	0.8239	0.8830	0.7888
MAMS test	0.6988	0.6000	0.6169	0.6330	0.9659	0.8991	0.8963	0.8513	0.8791	0.9900	0.6240
val set	0.7061	0.6714	0.5967	0.6562	0.9778	0.8750	0.8963	0.7928	0.8544	0.9460	0.6860

The results of run with embedding + 4 encoder layers frozen also shows comparable results on other datasets as well. Table 4.5 shows the summary of results of the run on all the dataset. On both MAMS and validation set, micro f1 score for token quality increased slightly in comparison to baseline. The only loss in performance was seen in multilabel accuracy (overlap) of semEval restaurant dataset and even in that case the loss was very minor (0.0063). Overall it seems like in most cases freezing some layers is beneficial as comparable results could be achieved with lower computational requirements.



* baseline train took 8min 20sec on a HPC cluster with A40 GPU. On a more general consumer hardware with a GTX 1650M GPU, it took 3hr 32min 09sec. Time for other runs can be calculated using this.

Figure 4.3: Relative runtime ratio of different freezing experiment runs. Each bar represents the ratio of runtime of a particular run to baseline run. On run B 0.945 on means it took 0.945 times the training time of baseline training time.

Sometimes even lower performance are acceptable if there are significant savings to be had in terms of time. Figure 4.3 shows the runtime factor of all six runs which was calculated by dividing experiment runtime by baseline runtime. As expected we can see that the runtime decrease with increase in number of frozen layers. Our best run (embedding + 4 encoder layers frozen) is ∼11% faster than the baseline run, which is a very small gain but it might be useful in cases with limited time and computational resources. For the extreme case of freezing the entire BERT model, training runtime was ∼34% faster but as previously discussed it comes at a cost of loss in metrics.

4.3 Model Sizes

Using a larger model generally results in better performance as having more parameter allows the model to capture and represent more complex patterns and relationships in the data they are trained on. But it also comes with certain trade-offs, such as increased computational requirements, longer training times, and higher costs. So, depending on the task, resources available and performance needed different model sizes might be suitable.

If we compare the models based on purely on metrics, BERT large uncased obtained similar f1 score (1.01 times) as BERT base (baseline) which was 1.19 times higher than BERT tiny (smallest model). But the model is 3.09 times larger than the BERT base and 77.27 times larger than the BERT tiny. Overall accuracy of the models is more or less similar for all the models and BERT base achieved accuracy of 98.49% which was the best result and 0.08% higher than BERT large. Bert mini which has 6.9M more parameters

Table 4.6: Token Quality results of runs with models of different size. These results should be used carefully as using purely metrics like f1 and accuracy for models with different size is not optimal as they do not factor in the computational resources used.

Model		Token Quality				
name	number of parameters	f1 (NEG)	f1 (POS)	f1 (NEU)	f1 (micro)	accuracy (overall)
bert-base-uncased (baseline)	110M	**0.6851**	0.8033	0.4588	0.7225	**0.9849**
bert-tiny-uncased	4.4M	0.5030	0.7060	0.4077	0.6165	0.9795
bert-mini-uncased	11.3M	0.6273	0.7525	0.5139	0.6819	0.9824
bert-large-uncased	340M	0.6825	**0.8103**	0.4281	**0.7320**	0.9841
miniLM-L12	33M	0.6771	0.7882	**0.5219**	0.7145	0.9829
roberta-base	125M	0.6718	0.7919	0.5085	0.7227	0.9814

(2.57 times) than BERT small improved the results a lot and achieved f1 score of 0.6819. The mini model was able to achieve 0.94 times the f1 score of base model with 90% smaller size. This result suggests that models could be reduced in size a lot with just minimal loss in terms of metrics. All the models still struggle mainly in neutral class which is expected as it is usually the gray area even for humans.

Table 4.7: Complete result of best performing run among model size experiments. The optimal result was obtained by bert-large-uncased. Even though bert-large performs best in terms of metrics, it takes longer to train, so it might not be an optimal model to use in most cases.

	Token Quality					Multilabel					
	f1 (NEG)	f1 (POS)	f1 (NEU)	f1 (micro)	accuracy (overall)	f1 (NEG)	f1 (POS)	f1 (NEU)	f1 (micro)	accuracy (overlap)	accuracy (exact)
semeval test (Restaurant)	0.6825	0.8103	0.4281	0.7320	0.9841	0.7778	0.8902	0.5394	0.8063	0.8690	0.7621
MAMS test	0.7016	0.6138	0.6322	0.6437	0.9672	0.8783	0.9122	0.8586	0.8807	0.9920	0.6360
val set	0.6929	0.6486	0.5939	0.6412	0.9769	0.8529	0.8743	0.7954	0.8406	0.9180	0.6620

We expected the larger BERT large uncased model to perform much better as it should be better at learning the contextual information with the more parameters available. Contrary to the expectation the model achieved a f1 score of 0.6437 in MAMS test set which is slightly higher than the score achieved by baseline. In multilabel metrics both models (BERT base and BERT large) show similar performance and in the validation set BERT large performs slightly worse. The lower performance in validation set could be because the hyperparameters used were optimized for BERT base. This result suggests the size of our datasets is too small for a larger model to be beneficial.

Another approach to reduce the model size without losing performance is to alter the pretraining dataset or the model training method. MiniLM uses knowledge distillation to achieve a model that is ∼70% smaller but achieves comparable f1 score. Roberta base

Table 4.8: Relative size and f1. The values represent a ratio of model size and f1 score between Run A to Run B. This means 25.00S, 1.17F between bert-base (Run A) and bert-tiny (Run B) means bert-base is 25 times larger but only achieves 1.17 times the f1 score. The matrix is inversely flipped along main diagonal.

		Run B					
		bert-base	bert-tiny	bert-mini	bert-large	miniLM-L12	roberta-base
Run A	bert-base	1.00 S 1.00 F	25.00 S 1.17 F	9.73 S 1.06 F	0.32 S 0.99 F	3.33 S 1.01 F	0.88 S 1.00 F
	bert-tiny	0.04 S 0.85 F	1.00 S 1.00 F	0.39 S 0.90 F	0.01 S 0.84 F	0.13 S 0.86 F	0.04 S 0.85 F
	bert-mini	0.10 S 0.94 F	2.57 S 1.11 F	1.00 S 1.00 F	0.03 S 0.93 F	0.34 S 0.95 F	0.09 S 0.94 F
	bert-large	3.09 S 1.01 F	77.27 S 1.19 F	30.09 S 1.07 F	1.00 S 1.00 F	10.30 S 1.02 F	2.72 S 1.01 F
	miniLM-L12	0.30 S 0.99 F	7.50 S 1.16 F	2.92 S 1.05 F	0.10 S 0.98 F	1.00 S 1.00 F	0.26 S 0.99 F
	roberta-base	1.14 S 1.00 F	28.41 S 1.17 F	11.06 S 1.06 F	0.37 S 0.99 F	3.79 S 1.01 F	1.00 S 1.00 F

Note: S refers to model size
Note: F refers to overall token quality f1 score on semeval restaurant test set

which differs in terms of their training strategies and model variants is slightly larger than BERT base (15M more parameters) achieved similar result in f1 score. The different training strategies used in roberta do not seem to be beneficial for our task. But depending on the task and available dataset, variants of roberta could show better performance.

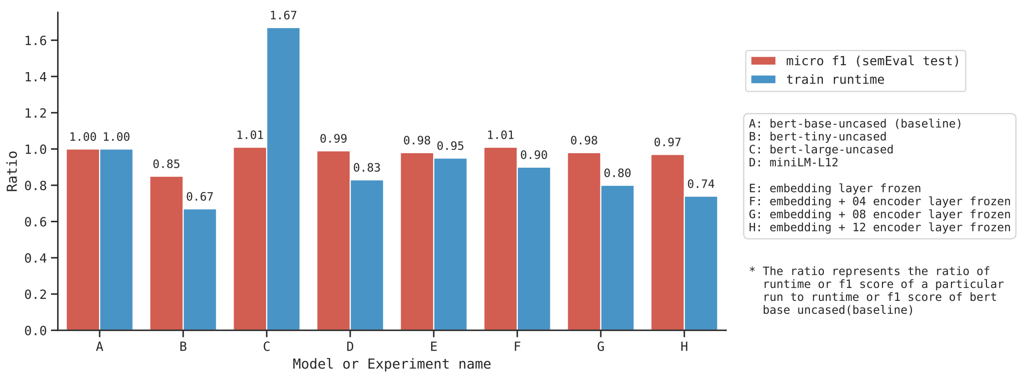


Figure 4.4: Barplot showing relative train runtime and f1 score. Different model runs from model size and layer freezing experiments are used.

Depending on the available resources using different model size or freezing layers might be beneficial. Figure 4.4 shows the runtime and performance ration of different model sizes and layer freezing runs to baseline run. Both BERT large and BERT base with embedding + 4 encoder layers frozen were able to achieve better result than the baseline, but the larger model took 67% more time to train while frozen model achieved similar result taking 10% less train time than baseline. Other than these two, miniLM-L12 was the closest to the baseline in terms of training time, but it only took 17% less training time. This suggests that rather than increasing the size of model, trying out alternative model training schemes or freezing some layers of pretrained model might improve performance while also saving model train runtime.

4.4 Cross domain

All our models were trained on a combined dataset of semEval Restaurant and MAMS, both of the datasets are related to restaurant review domain. To access the performance of our model across different domain and to get insights into its generalization capabilities, we will be using semEval Laptop dataset. The performance of relevant experiments on semEval Laptop test set are summarized in Table 4.9(Token Quality) and 4.10(multilabel).

Table 4.9: Token quality metrics of testing models trained on restaurant domain on laptop domain. Evaluation was done using semEval Laptop test set. Runs that performed best previously on domain same as training set do not necessarily perform the best on cross domain. This could be because of different pre-training methodologies and datasets used during pre-training.

	Token Quality				
	f1 (NEG)	f1 (POS)	f1 (NEU)	f1 (micro)	accuracy (overall)
optimized-run-02	0.2870	0.3390	0.1317	0.2764	0.9819
freezing-04-layer-05	0.3574	0.3369	0.0112	0.2825	0.9819
freezing-12-layer-07	0.3535	0.3625	0.0230	0.2941	0.9830
bert-tiny-08	0.0994	0.2056	0.0268	0.1407	0.9814
miniLM-L12	**0.4089**	**0.4780**	**0.2950**	**0.4160**	**0.9849**
bert-large-uncased	0.3982	0.3939	0.0683	0.3294	0.9824
roberta-base	0.3220	0.4339	0.0226	0.3262	0.9825

None of our runs were able to achieve best results which is expected as we are using models trained on a different domain. The baseline run falls behind all the freezing experimental runs and larger models. It only achieved an overall micro f1 score of 0.2764. Bert tiny showed the worse performance with only achieving f1 score of 0.1407 which could be because of smaller model size which means smaller domain knowledge. Especially in cases of neutral classes where models have struggled in restaurant domain as well, f1 score ranges from 0.0112(embedding + 4 encoder layers frozen) to 0.2950(miniLM). The best performing model was miniLM in all classes and achieved an overall f1 score of 0.4160. This might be because of wider variety of datasets used to pretrain miniLM. The accuracy score is higher than 0.98 for all runs, which might be skewed because of large number of 'O' tokens present in the dataset. Performing some sort of additional fine-tuning for a few epochs on semEval laptop dataset could improve the performance further.

Table 4.10: Multilabel classification results of testing models trained on restaurant domain on laptop domain. Evaluation was done using semEval Laptop test set. On both token quality and multilabel cross domain test, Microsoft minilm shows the best performance.

	Multilabel					
	f1 (NEG)	f1 (POS)	f1 (NEU)	f1 (micro)	accuracy (overlap)	accuracy (exact)
optimized-run-02	0.4362	0.4924	0.2123	0.4126	0.5496	0.5204
freezing-04-layer-05	0.4800	0.5140	0.0656	0.4320	0.5725	0.5496
freezing-12-layer-07	0.4681	0.5147	0.0339	0.4183	0.5878	0.5700
bert-tiny-08	0.1926	0.3925	0.1697	0.2976	0.5178	0.5025
miniLM-L12	**0.5436**	**0.6158**	**0.4390**	**0.5543**	**0.6603**	0.6107
bert-large-uncased	0.4821	0.5422	0.1633	0.4571	0.5509	0.5267
roberta-base	0.4972	0.6000	0.1000	0.4828	0.6590	**0.6361**

The performance in terms of multilabel metrics is a bit better. Other than BERT mini, all models have achieved scores higher than 0.4. Once again miniLM outperforms other runs in terms of multilabel metrics as well achieving an overall f1 score of 0.5543 and overlap accuracy of 0.6603. All models achieved exact accuracy of higher than 0.50 which means the model exactly predicts the multi class for at least half the cases. Best exact accuracy achieved by roberta base with a score of 0.6361 and miniLM achieved 0.6107. It's worth noting that while MiniLM outperforms BERT on this cross-domain tests, BERT remains a highly effective language model with strong performance across a wide range of tasks and domains.

4.5 Comparison with existing work

To validate our work and see how it compares to existing state-of-the-art models, we will be comparing out results with some of the models that have been previously discussed in Section 2.9. The results of other models were taken from their associated published papers and were not reproduced by us. They are all summarized in Table 4.11.

Our work shows similar performance to existing state-of-the-art models. Our model obtains similar results as just using BERT for an end to end system or using only CRF with BERT. In the experiments performed by Li et al. BERT end to end achieved f1 score of 0.6710 and adding CRF on top increased the score to 0.7317. Triple extraction method by Peng et al. used a very different approach and achieved f1 score of 0.7195 in joint tasks, they have reported higher scores in individual Aspect extraction and Aspect classification tasks. The other two models SPAN and InstructABSA have reported better results than our model. Hu et al. reported their SPAN model has achieved f1 score of 0.7492.

Table 4.11: Comparison of our work with existing works. Results of other works were taken from the original paper or their official GitHub repository (whichever is latest).

Model name	micro f1 (semEval 14 restaurant test)	Paper
BERT + LSTM + CRF (embed + 4 encoder layers frozen)	0.7331	OURS
BERT E2E	0.6710	[40]
ATSE (Knowing What, How, and Why)	0.7195	[33]
BERT + CRF	0.7317	[40]
SPAN	0.7492	[38]
Instruct ABSA	**0.7947**	[41]

Out of the models we researched, Instruct et al. by Scaria et al. has reported the best result. It uses Tk-Instruct and a chat style training as its backbone. It is also the newest model introduced for ABSA. On semEval 14 restaurant dataset, they report a f1 score of 0.7947 which is 0.0626 (∼8.5%) higher than our model. Using newer GPT based models and different training methodologies could further improve the performance.

4.6 Categorical classification

As a gold standard for classification into categories is not provided with any datasets, we won't be discussing the results in terms of metrics rather we will only be doing general analysis. We used Food, Drinks, Service, Price, Atmosphere and Others as our categories and each aspect is classified into one of the categories.

Table 4.12: Number of Reviews with Food and Service categories in semEval restaurant and MAMS test datasets. Only food and services were chosen as other categories have very small number of reviews.

Dataset	**Food**			**Service**		
	Negative	**Neutral**	**Positive**	**Negative**	**Neutral**	**Positive**
SemEval	92	144	612	40	30	90
MAMS	81	362	369	130	31	56

As expected most of the aspects in both datasets are related to food (50.87% in semeval and 46.720% in MAMS) followed by service and atmosphere categories (excluding others category as includes everything unknown). Table 4.12 presents a summary of sentiments in semeval and MAMS dataset for Food and Service categories. Food category has mostly received positive reviews but in case of other categories neutral sentiment is more prevalent. Especially in MAMS dataset, Negative reviews of Service appear more often and looking closely we found that most of it is related to restaurant staff (waiters and managers).

We found that even the reviews with bad sentiment towards staff had positive sentiment towards food. This suggests that people are more harsh and sensitive towards staff when writing reviews. If we look closer at the other category, we can see that our approach of using dictionary and wordnet mostly fails in cases of words written in different grammatical forms and words relating to the establishment. Even tough wordnet classifies the words to similar hypernyms (dish and food) as our categories, our approach fails to catch them. This categorization could be further improved by using a more sophisticated classifier.

4.7 Inference on PandIA dataset

Our models do not perform very well in cross domain scenarios, and we also lack any datasets in form of tweets, so the classification is suboptimal. We used Microsoft MiniLM as it performed the best in our cross domain testing. Due to lack of gold standard, we will only discuss the results in terms of sentiments towards specific categories and not in terms of metrics. Since this data is related to Covid tweets, we will be using Symptoms, Medication and Vaccination, Personal experience, Politics and Other as categories. A lot of the aspect terms detected seemed incorrect, but it was expected as dataset was very different from the training data.

Table 4.13: Result of performing categorization on Pandia dataset aspect terms. As pandia dataset is too different from the training dataset, results are suboptimal.

Category	Neutral	Positive	Negative
Symptoms	93	39	5
Medication & Vaccination	5	84	3
Personal experience	8	2	15
Politics	9	4	3
Other	508	673	93

As expected most of the terms were categorized as Others and were mostly Negative as the data was relating to Covid in general. Out of 1534 aspects extracted by the model, 137 aspects were categorized as Symptom with 93 classified as Negative, 39 as Neutral and 5 as Positive. Medication and Vaccination was mainly Neutral and personal experience was present in very low amount, but they were almost always negative which shows the personal struggles people had during Covid. We expected to detect a lot more Politics and Country, they were only present in very low numbers which might be due to shortcomings of our categorization methods and poor performance of model on cross domain. On a closer look at other category, we observed that our model had detected a lot of pronouns, location and travel related words as aspect. By using some additional tweet data as part of training process and also using a better categorization methods, much better results could be achieved, but our work shows that this approach is possible to obtain general categories.

Chapter 5

Conclusion

Over the years, research in the field of NLP has grown immensely with the advent of transformers. This dissertation aimed to create a transformer based model for ABSA and test its performance on different settings. It also aimed to answer some questions related to the impact of model size and freezing of layers on ABSA tasks. The findings found in this work could also be applied towards other NLP tasks.

The document started by providing a general idea of problems and research questions that we aimed to answer with this work. Then, a summary of background research and literature review was provided with the aim of showcasing current state-of-the-art and also highlight why a transformer based model is useful for this task.

In Chapter 3 details about the implemented model and system around it was provided. In brief, we implemented a BERT+BiLSTM+Masked CRF model leveraging the power of huggingface trainer and transformers package. The model was implemented in PyTorch and is also publicly available on huggingface hub. Pre-trained BERT (or similar transformer) was used as tokenizer and encoder in our model which allowed it to use strong contextual information and also handle out of vocab words with wordpiece tokenization. As this work used collapsed labelling system and handled ABSA task as a token classification problem, LSTM was used as a decoder. Using a LSTM as a decoder in combination with a pre-trained BERT encoder can effectively model the sequential nature of token classification tasks, handle variable-length outputs, capture dependencies between tokens, and provide training efficiency. Then as final classifier a masked CRF was used. Masked CRF allowed us to set negative transitions so that probabilities of transitions like B-X to I-Y were less likely to occur. Once a base model was developed, several experiments were performed to see impact of different factors on the model.

Our baseline model achieved promising results in terms of multilabel classification but showed room for improvement in token quality metrics. Through an optimization study, we identified a set of optimal hyperparameters that improved the token quality metrics while maintaining comparable performance in multilabel classification. The token quality f1 score on validation set only increased by 0.0323 using optimal hyperparameters. Some more increment could have been achieved by a deeper hyperparameters search but due to computational resource constraints we were only able to search on a small search space. In the hyperparameter search we saw that number of training epochs and learning rate had the biggest impact. We also saw that our model struggles mainly with neutral cases which was expected as neutral cases are usually gray area for humans as well. This optimized model served as a baseline benchmark for further experiments.

We also investigated the impact of freezing layers in the model and found that freezing a subset of layers can lead to comparable performance with reduced computational requirements. In our case freezing embedding and 4 encoder layers gave us 1% better results with 10% reduction in training time. Additionally, we explored the effect of model size on performance. We observed that using a larger model does not necessarily increase performance as miniLM (a model with similar size as bert-base) outperformed bert-large. On cross domain tests as well miniLM, a model which uses different pre-training approach to bert-base obtained best results which highlight the importance of proper pre-training approach.

As future work, we would like to continue experimenting with Transformer and ABSA (and NLP in general).

- Use our model for some other NLP tasks to see if the results can be generalized.
- Perform a deeper and run wise hyperparameter search as optimal parameters might differ from configuration to configuration.
- Combine model size and layer freezing experiments to see if larger models also show better performance with frozen layers.
- Computer Assisted Annotation, Create a system that allows the creation of new ABSA dataset while continuously fine-tuning existing model to make predictions better as it continues and create a PandIA ABSA dataset using this system.
- Use a multilingual model trained on Portuguese tweets to find sentiments towards Portuguese political parties during COVID-19.

Overall, our findings highlight the effectiveness of transformers in ABSA tasks. The combination of optimal hyperparameters, layer freezing, and model size selection can lead to improved performance while considering computational constraints. Our research contributes to the understanding of transformer-based models for ABSA and provides insights for future studies in this field. Using the results from this dissertation, it can be confidently said that there are better ways (like different pre-training approaches and freezing of encoder layers) to achieve better performing model than increasing model size. Also, as a plus points, alternatives to increasing model size usually reduce the computational resource and time requirements as well. Transformers have shown great potential in ABSA, and further advancements and fine-tuning of models can enhance their performance and applicability in real-world sentiment analysis tasks across diverse domains.

References

[1] Carlos A. Iglesias and Antonio Moreno. "Sentiment Analysis for Social Media." In: *Applied Sciences* 9.23 (Nov. 2019), p. 5037. ISSN: 2076-3417. DOI: 10.3390/app9235037.
URL: https://www.mdpi.com/2076-3417/9/23/5037 (cit. on p. 1).

[2] Wenxuan Zhang, Xin Li, Yang Deng, Lidong Bing, and Wai Lam. "A Survey on Aspect-Based Sentiment Analysis: Tasks, Methods, and Challenges." In: (2022). DOI: 10.48550/ARXIV.2203.01054.
URL: https://arxiv.org/abs/2203.01054 (cit. on p. 1).

[3] *What Are Neural Networks? | IBM.*
URL: https://www.ibm.com/topics/neural-networks (cit. on p. 5).

[4] *What Is Natural Language Processing? | IBM.*
URL: https://www.ibm.com/topics/natural-language-processing (cit. on p. 6).

[5] Ivano Lauriola, Alberto Lavelli, and Fabio Aiolli. "An Introduction to Deep Learning in Natural Language Processing: Models, Techniques, and Tools." In: *Neurocomputing* 470 (Jan. 2022), pp. 443–456. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2021.05.103.
URL: https://www.sciencedirect.com/science/article/pii/S0925231221010997 (cit. on p. 6).

[6] J J Hopfield. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities." In: *Proceedings of the National Academy of Sciences* 79.8 (Apr. 1982), pp. 2554–2558. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.79.8.2554.
URL: https://pnas.org/doi/full/10.1073/pnas.79.8.2554 (cit. on p. 6).

[7] *10.1. Long Short-Term Memory (LSTM) — Dive into Deep Learning 1.0.0-Beta0 Documentation.*
URL: https://d2l.ai/chapter_recurrent-modern/lstm.html (cit. on p. 7).

[8] Alex Sherstinsky. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network." In: *Physica D: Nonlinear Phenomena* 404 (Mar. 2020), p. 132306. ISSN: 0167-2789. DOI: 10.1016/j.physd.2019.132306.

URL: https://www.sciencedirect.com/science/article/pii/S0167278919305974 (cit. on p. 6).

[9] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory.” In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735.
URL: https://direct.mit.edu/neco/article/9/8/1735-1780/6109 (cit. on p. 7).

[10] Lilian Weng. *Attention? Attention!* June 2018.
URL: https://lilianweng.github.io/posts/2018-06-24-attention/ (cit. on p. 7).

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate.* May 2016. DOI: 10.48550/arXiv.1409.0473. arXiv: 1409.0473 [cs, stat].
URL: http://arxiv.org/abs/1409.0473 (cit. on p. 8).

[12] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches.* Oct. 2014. DOI: 10.48550/arXiv.1409.1259. arXiv: 1409.1259 [cs, stat].
URL: http://arxiv.org/abs/1409.1259 (cit. on p. 8).

[13] Alex Graves, Greg Wayne, and Ivo Danihelka. *Neural Turing Machines.* Dec. 2014. DOI: 10.48550/arXiv.1410.5401. arXiv: 1410.5401 [cs].
URL: http://arxiv.org/abs/1410.5401 (cit. on p. 8).

[14] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. *Effective Approaches to Attention-based Neural Machine Translation.* Sept. 2015. DOI: 10.48550/arXiv.1508.04025. arXiv: 1508.04025 [cs].
URL: http://arxiv.org/abs/1508.04025 (cit. on pp. 8, 9).

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need.* Dec. 2017. DOI: 10.48550/arXiv.1706.03762. arXiv: 1706.03762 [cs].
URL: http://arxiv.org/abs/1706.03762 (cit. on pp. 8, 10–13).

[16] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.* Apr. 2016. DOI: 10.48550/arXiv.1502.03044. arXiv: 1502.03044 [cs].
URL: http://arxiv.org/abs/1502.03044 (cit. on p. 10).

[17] Jianpeng Cheng, Li Dong, and Mirella Lapata. *Long Short-Term Memory-Networks for Machine Reading.* Sept. 2016. DOI: 10.48550/arXiv.1601.06733. arXiv: 1601.06733 [cs].
URL: http://arxiv.org/abs/1601.06733 (cit. on pp. 10, 11).

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* May 2019. DOI: 10.48550/arXiv.1810.04805. arXiv: 1810.04805 [cs]. URL: http://arxiv.org/abs/1810.04805 (cit. on pp. 14, 15).

[19] J. Lafferty, A. McCallum, and Fernando Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data." In: *International Conference on Machine Learning.* June 2001. URL: https://www.semanticscholar.org/paper/Conditional-Random-Fields%3A-Probabilistic-Models-for-Lafferty-McCallum/f4ba954b0412773d047dc41231c733de0c1f4926 (cit. on p. 15).

[20] Tianwen Wei, Jianwei Qi, Shenghuan He, and Songtao Sun. "Masked Conditional Random Fields for Sequence Labeling." In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Online: Association for Computational Linguistics, June 2021, pp. 2024–2035. DOI: 10.18653/v1/2021.naacl-main.163. URL: https://aclanthology.org/2021.naacl-main.163 (cit. on pp. 15, 16).

[21] J Jayalekshmi and Tessy Mathew. "Facial Expression Recognition and Emotion Classification System for Sentiment Analysis." In: *2017 International Conference on Networks & Advances in Computational Technologies (NetACT).* July 2017, pp. 1–8. DOI: 10.1109/NETACT.2017.8076732. (Cit. on p. 16).

[22] Bing Liu. *Sentiment Analysis and Opinion Mining.* Synthesis Lectures on Human Language Technologies. Cham: Springer International Publishing, 2012. ISBN: 978-3-031-01017-0 978-3-031-02145-9. DOI: 10.1007/978-3-031-02145-9. URL: https://link.springer.com/10.1007/978-3-031-02145-9 (cit. on p. 16).

[23] Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews." In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD '04. New York, NY, USA: Association for Computing Machinery, Aug. 2004, pp. 168–177. ISBN: 978-1-58113-888-7. DOI: 10.1145/1014052.1014073. URL: https://doi.org/10.1145/1014052.1014073 (cit. on p. 17).

[24] Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. *An Interactive Multi-Task Learning Network for End-to-End Aspect-Based Sentiment Analysis.* June 2019. DOI: 10.48550/arXiv.1906.06906. arXiv: 1906.06906 [cs]. URL: http://arxiv.org/abs/1906.06906 (cit. on p. 17).

[25] Chen Zhang, Qiuchi Li, and Dawei Song. *Aspect-Based Sentiment Classification with Aspect-specific Graph Convolutional Networks.* Oct. 2019. DOI: 10.48550/arXiv.1909.03477. arXiv: 1909.03477 [cs]. URL: http://arxiv.org/abs/1909.03477 (cit. on pp. 17, 18).

[26] Huaishao Luo, Lei Ji, Tianrui Li, Nan Duan, and Daxin Jiang. *GRACE: Gradient Harmonized and Cascaded Labeling for Aspect-based Sentiment Analysis*. Sept. 2020. DOI: 10.48550/arXiv.2009.10557. arXiv: 2009.10557 [cs]. URL: http://arxiv.org/abs/2009.10557 (cit. on p. 17).

[27] Alexander Rietzler, Sebastian Stabinger, Paul Opitz, and Stefan Engl. *Adapt or Get Left Behind: Domain Adaptation through BERT Language Model Finetuning for Aspect-Target Sentiment Classification*. Nov. 2019. DOI: 10.48550/arXiv.1908.11860. arXiv: 1908.11860 [cs]. URL: http://arxiv.org/abs/1908.11860 (cit. on p. 17).

[28] Wei Xue and Tao Li. *Aspect Based Sentiment Analysis with Gated Convolutional Networks*. May 2018. DOI: 10.48550/arXiv.1805.07043. arXiv: 1805.07043 [cs]. URL: http://arxiv.org/abs/1805.07043 (cit. on p. 17).

[29] Hyun-jung Park, Minchae Song, and Kyung-Shik Shin. "Deep Learning Models and Datasets for Aspect Term Sentiment Classification: Implementing Holistic Recurrent Attention on Target-Dependent Memories." In: *Knowledge-Based Systems* 187 (Jan. 2020), p. 104825. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2019.06.033. URL: https://www.sciencedirect.com/science/article/pii/S0950705119303004 (cit. on p. 17).

[30] Xin Li, Lidong Bing, Piji Li, and Wai Lam. *A Unified Model for Opinion Target Extraction and Target Sentiment Prediction*. Feb. 2019. DOI: 10.48550/arXiv.1811.05082. arXiv: 1811.05082 [cs]. URL: http://arxiv.org/abs/1811.05082 (cit. on p. 17).

[31] Li Deng and Yang Liu, eds. *Deep Learning in Natural Language Processing*. Singapore: Springer, 2018. ISBN: 978-981-10-5208-8 978-981-10-5209-5. DOI: 10.1007/978-981-10-5209-5. URL: http://link.springer.com/10.1007/978-981-10-5209-5 (cit. on p. 17).

[32] Xuefeng Bai, Pengbo Liu, and Yue Zhang. "Investigating Typed Syntactic Dependencies for Targeted Sentiment Classification Using Graph Attention Neural Network." In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 503–514. ISSN: 2329-9304. DOI: 10.1109/TASLP.2020.3042009. (Cit. on p. 17).

[33] Haiyun Peng, Lu Xu, Lidong Bing, Fei Huang, Wei Lu, and Luo Si. *Knowing What, How and Why: A Near Complete Solution for Aspect-based Sentiment Analysis*. Nov. 2019. DOI: 10.48550/arXiv.1911.01616. arXiv: 1911.01616 [cs]. URL: http://arxiv.org/abs/1911.01616 (cit. on pp. 17, 23, 24, 53).

[34] Hyeju Jang, Emily Rempel, David Roth, Giuseppe Carenini, and Naveed Zafar Janjua. "Tracking COVID-19 Discourse on Twitter in North America: Infodemiology Study Using Topic Modeling and Aspect-Based Sentiment Analysis." In: *Journal of Medical Internet Research* 23.2 (Feb. 2021), e25431. DOI: 10.2196/25431. URL: https://www.jmir.org/2021/2/e25431 (cit. on p. 20).

[35] Huimin Chen, Zeyu Zhu, Fanchao Qi, Yining Ye, Zhiyuan Liu, Maosong Sun, and Jianbin Jin. "Country Image in COVID-19 Pandemic: A Case Study of China." In: *IEEE Transactions on Big Data* 7.1 (Mar. 2021), pp. 81–92. ISSN: 2332-7790. DOI: 10.1109/TBDATA.2020.3023459. (Cit. on p. 20).

[36] Wenhao Zhang, Hua Xu, and Wei Wan. "Weakness Finder: Find Product Weakness from Chinese Reviews by Using Aspects Based Sentiment Analysis." In: *Expert Systems with Applications* 39.11 (Sept. 2012), pp. 10283–10291. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2012.02.166.
URL: https://www.sciencedirect.com/science/article/pii/S0957417412004290 (cit. on p. 20).

[37] Tun Thura Thet, Jin-Cheon Na, and Christopher S.G. Khoo. "Aspect-Based Sentiment Analysis of Movie Reviews on Discussion Boards." In: *Journal of Information Science* 36.6 (Dec. 2010), pp. 823–848. ISSN: 0165-5515, 1741-6485. DOI: 10.1177/0165551510388123.
URL: http://journals.sagepub.com/doi/10.1177/0165551510388123 (cit. on p. 20).

[38] Minghao Hu, Yuxing Peng, Zhen Huang, Dongsheng Li, and Yiwei Lv. *Open-Domain Targeted Sentiment Analysis via Span-Based Extraction and Classification.* June 2019.
URL: https://arxiv.org/abs/1906.03820v1 (cit. on pp. 21, 53).

[39] Huaishao Luo, Tianrui Li, Bing Liu, and Junbo Zhang. *DOER: Dual Cross-Shared RNN for Aspect Term-Polarity Co-Extraction.* June 2019. DOI: 10.48550/arXiv.1906.01794. arXiv: 1906.01794 [cs].
URL: http://arxiv.org/abs/1906.01794 (cit. on p. 22).

[40] Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. *Exploiting BERT for End-to-End Aspect-based Sentiment Analysis.* Oct. 2019. DOI: 10.48550/arXiv.1910.00883. arXiv: 1910.00883 [cs].
URL: http://arxiv.org/abs/1910.00883 (cit. on pp. 22, 23, 53).

[41] Kevin Scaria, Himanshu Gupta, Siddharth Goyal, Saurabh Arjun Sawant, Swaroop Mishra, and Chitta Baral. *InstructABSA: Instruction Learning for Aspect Based Sentiment Analysis.* Apr. 2023. DOI: 10.48550/arXiv.2302.08624. arXiv: 2302.08624 [cs].
URL: http://arxiv.org/abs/2302.08624 (cit. on pp. 24, 53).

[42] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. "SemEval-2014 Task 4: Aspect Based Sentiment Analysis." In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014).* Dublin, Ireland: Association for Computational Linguistics, 2014, pp. 27–35. DOI: 10.3115/v1/S14-2004.
URL: http://aclweb.org/anthology/S14-2004 (cit. on p. 26).

[43] Qingnan Jiang, Lei Chen, Ruifeng Xu, Xiang Ao, and Min Yang. “A Challenge Dataset and Effective Models for Aspect-Based Sentiment Analysis.” In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6280–6285. DOI: 10.18653/v1/D19-1654.
URL: https://aclanthology.org/D19-1654 (cit. on p. 27).

[44] Richard Adolph Aires Jonker, Roshan Poudel, Olga Fajarda, Sérgio Matos, José Luís Oliveira, and Rui Pedro Lopes. “Portuguese Twitter Dataset on COVID-19.” In: *2022 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. Istanbul, Turkey: IEEE, Nov. 2022, pp. 332–338. ISBN: 978-1-66545-661-6. DOI: 10.1109/ASONAM55673.2022.10068592.
URL: https://ieeexplore.ieee.org/document/10068592/ (cit. on p. 28).

[45] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. “Transformers: State-of-the-Art Natural Language Processing.” In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6.
URL: https://aclanthology.org/2020.emnlp-demos.6 (cit. on p. 32).

[46] Tiago Almeida, Rui Antunes, João F. Silva, João R Almeida, and Sérgio Matos. “Chemical Identification and Indexing in PubMed Full-Text Articles Using Deep Learning and Heuristics.” In: *Database* 2022 (July 2022), baac047. ISSN: 1758-0463. DOI: 10.1093/database/baac047.
URL: https://academic.oup.com/database/article/doi/10.1093/database/baac047/6625810 (cit. on p. 35).

[47] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. “Optuna: A Next-generation Hyperparameter Optimization Framework.” In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage AK USA: ACM, July 2019, pp. 2623–2631. ISBN: 978-1-4503-6201-6. DOI: 10.1145/3292500.3330701.
URL: https://dl.acm.org/doi/10.1145/3292500.3330701 (cit. on p. 37).

[48] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization*. June 2018. DOI: 10.48550/arXiv.1603.06560. arXiv: 1603.06560 [cs, stat].
URL: http://arxiv.org/abs/1603.06560 (cit. on p. 37).

[49] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. *How to Fine-Tune BERT for Text Classification?* Feb. 2020. DOI: 10.48550/arXiv.1905.05583. arXiv: 1905.05583 [cs].
URL: http://arxiv.org/abs/1905.05583 (cit. on p. 37).