Domain Adaptation through Adversarial Training of Albert for Aspect-based Sentiment Analysis

Supervisor: Lianlian Qi

Daniel Atticus Williams[1]

19031356

Submission date: 11 September 2020

[1]**Disclaimer:** This report is submitted in partial fulfillment of the requirements for the degree of the Master of Science of Machine Learning at UCL. It is substantially the result of my own work except where explicitly indicated in the text.
The report will be distributed to the internal and external examiners, but thereafter may not be copied or distributed except with permission from the author.

Abstract

This project involves the application of the Albert language model to the problem of aspect-based sentiment analysis (ABSA). Driven by an expansion in the use of online reviews, an established body of research into sentiment analysis of review targets (e.g. restaurants) has motivated the study of finer-grained target aspects (e.g. restaurant service, ambiance). With the advent of transformer-based neural network architectures such as BERT, resulting gains in performance have required significantly high usage of computational resources. A recent transformer model called Albert has shown competitive performance on benchmark language tasks with a much smaller memory footprint than BERT-based competitors. The goal of this project has been to investigate the performance of Albert on ABSA tasks such as aspect extraction (AE), aspect sentiment classification (ASC), and end-to-end ABSA (E2E-ABSA).

Our achievements are fourfold. First, we have adapted an adversarial training framework for ABSA tasks, substituting the smallest Albert model for further pre-trained BERT models. In addition, for E2E-ABSA we have adapted existing English and Mandarin datasets to use a consistent and unified tagging scheme. Using optimized hyperparameters with variants of our novel *Albat* architecture, we have achieved state of the art performance on all E2E-ABSA metrics, two AE metrics, and one ASC metric. Furthermore, on four AE metrics and four ASC metrics the optimized Albat model variants demonstrate competitive performance regarding the state of the art.

Acknowledgements

I would like to express my deep gratitude to my academic supervisor Lianlian Qi for her guidance in this project. Her wisdom and patience throughout the year have been tremendous and invaluable. I am also grateful to Dr Zafeirios Fountas for his generous and most helpful feedback concerning chapters of this report.

Ultimately I must thank my parents for their inexhaustible encouragement and support over this year. You have always spurred me on to do my best.

Contents

List of Figures **5**

List of Tables **7**

1 Introduction **9**

2 Literature Review **11**

2.1 ABSA 11

2.2 ABSA Approaches 12

2.2.1 Information Retrieval Systems 12

2.2.2 Machine Learning Systems 12

2.3 BERT for ABSA 15

2.3.1 Overview 15

2.3.2 Training Procedures 17

2.3.3 Training Data 20

2.3.4 Computational Considerations 20

2.4 Datasets for ABSA 22

3 Design **25**

3.1 Problem Formulation 25

3.2 Solution Design 26

3.2.1 Albat 26

3.2.2 Albat-PT 29

4 Results **30**

4.1 Experimental Setup . . . 30

4.1.1 Implementation . . . 30

4.1.2 Data . . . 31

4.1.3 Models . . . 34

4.2 Albat with One-Layer Classifier . . . 35

4.2.1 Hyperparameter Tuning . . . 35

4.2.2 Optimized Model Results . . . 37

4.3 Albat with Two-Layer Classifier . . . 38

4.3.1 Hyperparameter Tuning . . . 38

4.3.2 Optimized Model Results . . . 41

4.4 Albat with Three-Layer Classifier . . . 42

4.4.1 Hyperparameter Tuning . . . 42

4.4.2 Optimized Model Results . . . 46

4.5 Albat with Further Pre-training . . . 47

4.5.1 Albat-1LC . . . 47

4.5.2 Albat-2LC . . . 48

4.5.3 Albat-3LC . . . 49

5 Discussion **51**

5.1 Model Performance . . . 51

5.1.1 Metrics . . . 51

5.1.2 E2E-ABSA . . . 55

5.1.3 AE . . . 56

5.1.4 ASC . . . 59

5.1.5 Case Study . . . 65

5.1.6 Confusion Matrices . . . 69

5.1.7 Further Pre-Training . . . 77

5.2 Resource Usage . . . 78

6 Conclusions **84**

6.1 Summary 84

6.2 Contributions 85

6.3 Future Work 85

A Abbreviations Glossary **87**

B E2E-ABSA Literature: F_1 Score Averaging Method **89**

C Classification Performance Metrics **90**

C.1 Binary Classification Metrics 90

C.2 Multi-class Classification Metrics 91

Bibliography **93**

List of Figures

2.1 $BERT_{BASE}$ Architecture [1, 2] 16
2.2 Overview of BERT Pre-Training [3] 17
2.3 BERT Pre-Training Tasks 18
2.4 Fine-tuning [3] 19
2.5 Parameter Sharing [4] 21
2.6 Embedding Matrices 21
2.7 Sentence Order Prediction [4] 22

3.1 Albat Model Overview (with a One-Layer Classifier) 26

5.1 Performance Comparison with State of the Art for E2E-ABSA on Lapt14 (Accuracy) 55
5.2 Performance Comparison with State of the Art for E2E-ABSA on Unified (Accuracy) 55
5.3 Performance Comparison with State of the Art for AE on Lapt14 (F_1 Score) 56
5.4 Performance Comparison with State of the Art for AE on Rest14 (F_1 Score) 56
5.5 Performance Comparison with State of the Art for AE on Camera (F_1 Score) 57
5.6 Performance Comparison with State of the Art for AE on Car (F_1 Score) . . 57
5.7 Performance Comparison with State of the Art for AE on Notebook (F_1 Score) 58
5.8 Performance Comparison with State of the Art for AE on Phone (F_1 Score) . 58
5.9 Performance Comparison with State of the Art for ASC on Lapt14 (Accuracy) 59
5.10 Performance Comparison with State of the Art for ASC on Lapt14 (Macro-averaged F_1 Score) 59
5.11 Performance Comparison with State of the Art for ASC on Rest14 (Accuracy) 60
5.12 Performance Comparison with State of the Art for ASC on Rest14 (Macro-averaged F_1 Score) 60

5.13 Performance Comparison with State of the Art for ASC on Camera (Accuracy) 61
5.14 Performance Comparison with State of the Art for ASC on Camera (Macro-averaged F_1 Score) . . . 61
5.15 Performance Comparison with State of the Art for ASC on Car (Accuracy) . 62
5.16 Performance Comparison with State of the Art for ASC on Car (Macro-averaged F_1 Score) . . . 62
5.17 Performance Comparison with State of the Art for ASC on Notebook (Accuracy) 63
5.18 Performance Comparison with State of the Art for ASC on Notebook (Macro-averaged F_1 Score) . . . 63
5.19 Performance Comparison with State of the Art for ASC on Phone (Accuracy) 64
5.20 Performance Comparison with State of the Art for ASC on Phone (Macro-averaged F_1 Score) . . . 64
5.21 Performance Comparison with State of the Art for ASC on MAMS (Accuracy) 65
5.22 Confusion Matrix for Sample Review (Albat-1LC) . . . 67
5.23 Confusion Matrix for Sample Review (Albat-2LC) . . . 68
5.24 Confusion Matrix for Sample Review (Albat-3LC) . . . 68
5.25 Confusion Matrices for Lapt14 . . . 69
5.26 Confusion Matrices for Rest14 . . . 70
5.27 Confusion Matrices for Unified . . . 71
5.28 Confusion Matrices for MAMS . . . 72
5.29 Confusion Matrices for Camera . . . 73
5.30 Confusion Matrices for Car . . . 74
5.31 Confusion Matrices for Notebook . . . 75
5.32 Confusion Matrices for Phone . . . 76
5.33 Trainable Parameters for BAT and Albat . . . 79
5.34 Size of Trained Models for BAT and Albat . . . 80
5.35 Training Duration for BAT and Albat Models (AE) . . . 81
5.36 Training Duration for BAT and Albat Models (ASC) . . . 82
5.37 Training Duration for BAT and Albat Models (E2E-ABSA) . . . 83

B.1 Email Correspondence with Mr. Xin Li . . . 89

List of Tables

2.1 Attributes of Main ABSA Datasets 23

4.1 SemEval-2014 Laptop Dataset Properties 31
4.2 SemEval-2014 Restaurant Dataset Properties 32
4.3 Unified Dataset Properties 32
4.4 MAMS Dataset Properties 32
4.5 Camera Dataset Properties 33
4.6 Car Dataset Properties 33
4.7 Notebook Dataset Properties 33
4.8 Phone Dataset Properties 33
4.9 Influence of Dropout Rate on Albat-1LC Performance 35
4.10 Influence of Perturbation Size on Albat-1LC Performance 36
4.11 Influence of Weight Decay Constant on Albat-1LC Performance 36
4.12 Performance of Albat-1LC using Optimized Hyperparameters 37
4.13 Influence of Perturbation Size on Albat-2LC Performance (without ReLU) . 38
4.14 Influence of Perturbation Size on Albat-2LC Performance (using ReLU) . . . 38
4.15 Influence of Weight Decay Constant on Albat-2LC Performance (without ReLU) 39
4.16 Influence of Weight Decay Constant on Albat-2LC Performance (using ReLU) 39
4.17 Influence of First Classification Layer Output Dimension on Albat-2LC Performance (without ReLU) 40
4.18 Influence of First Classification Layer Output Dimension on Albat-2LC Performance (using ReLU) 40
4.19 Performance of Albat-2LC using Optimized Hyperparameters 41
4.20 Influence of Perturbation Size on Albat-3LC Performance (without ReLU) . 42

4.21 Influence of Perturbation Size on Albat-3LC Performance (using ReLU) . . . 42
4.22 Influence of Weight Decay Constant on Albat-3LC Performance (without ReLU) 43
4.23 Influence of Weight Decay Constant on Albat-3LC Performance (using ReLU) 43
4.24 Influence of First Classification Layer Output Dimension on Albat-3LC Performance (without ReLU) 44
4.25 Influence of First Classification Layer Output Dimension on Albat-3LC Performance (using ReLU) 44
4.26 Influence of Second Classification Layer Output Dimension on Albat-3LC Performance (without ReLU) 45
4.27 Influence of Second Classification Layer Output Dimension on Albat-3LC Performance (using ReLU) 45
4.28 Performance of Albat-3LC using Optimized Hyperparameters 46
4.29 Performance of Albat-1LC (PT using 1% Data for 1 Epoch) 47
4.30 Performance of Albat-1LC (PT using 1% Data for 4 Epochs) 47
4.31 Performance of Albat-1LC (PT using 5% Data for 1 Epoch) 48
4.32 Performance of Albat-1LC (PT using 5% Data for 2 Epochs) 48
4.33 Performance of Albat-2LC (PT using 1% Data for 1 Epoch) 49
4.34 Performance of Albat-2LC (PT using 1% Data for 4 Epochs) 49
4.35 Performance of Albat-2LC (PT using 5% Data for 1 Epoch) 49
4.36 Performance of Albat-2LC (PT using 5% Data for 2 Epochs) 49
4.37 Performance of Albat-3LC (PT using 1% Data for 1 Epoch) 50
4.38 Performance of Albat-3LC (PT using 1% Data for 4 Epochs) 50
4.39 Performance of Albat-3LC (PT using 5% Data for 1 Epoch) 50
4.40 Performance of Albat-3LC (PT using 5% Data for 2 Epochs) 50

Chapter 1

Introduction

With ubiquitous access to the Internet, members of the public increasingly share personal testimonials about goods and services on online review platforms. Due to the increasing volume of such reviews, there has been rising interest in automated sentiment analysis involving authors' opinions of review targets. This has created a burgeoning field of research in sentiment analysis of review texts.

Although most approaches have concentrated on an author's overall opinion of a review target (e.g. a specific restaurant), a growing area of research known as aspect-based sentiment analysis (ABSA) concentrates on detecting sentiments towards finer-grained aspects of review targets (e.g. the quality of service or ambiance of a restaurant). The advent of deep neural networks, particularly transformer architectures, has enabled the development of robust systems for ABSA tasks. Notably the transformer-based BERT architecture [2] has been applied successfully to ABSA. While much recent research has been conducted on improving the performance of BERT on ABSA tasks, BERT's relatively slow and memory-intensive training procedure is disadvantageous. Newer transformer-based models such as Albert [5] have sought to overcome these limitations by using smaller models with fewer parameters. The application of these transformer-based models to ABSA could therefore lead to faster training and better performance.

Motivated by these prospects, our project has focused on applying the Albert language model to ABSA. Our primary goal has been to integrate an adversarial training method with Albert in order to perform E2E-ABSA and other ABSA tasks.

We begin in Chapter 2 with a survey of ABSA literature, addressing both information retrieval and machine learning-based approaches, with the use of BERT in ABSA examined comprehensively. In Chapter 3 we formulate learning problems for three ABSA tasks and propose an adversarially trained Albert-based architecture called *Albat*. We present the

details of implementing Albat in Chapter 4 and perform hyperparameter optimization for three Albat model variants. Using these hyperparameter settings we then evaluate the performance on four datasets. Inspired by similar approaches with BERT for ABSA, we also pre-train Albert further using an online review corpus and evaluate its performance using the three Albat model variants. In Chapter 5 we discuss the experimental outcomes with comparisons to relevant literature results, a case study of a single review, analysis of frequent errors, and a study of the resource usage by the three Albat model variants. We conclude in Chapter 6 with a reflection on the contributions of this investigation to the literature and present a number of directions for future research.

Chapter 2

Literature Review

In this chapter we survey the literature relating to ABSA, with a particular focus on deep neural network approaches. We also discuss commonly-used datasets used to train and evaluate performance on ABSA tasks.

2.1 ABSA

As an instance of the text classification problem, sentiment analysis seeks to identify and classify an author's sentiment towards opinion targets in text, drawing on vocabulary from a specific domain (e.g. laptop computers, restaurants) [6]. ABSA considers the sentiment conveyed at a phrasal level regarding aspects of the opinion target. Such aspects may be terms (*battery life*, *screen size* for a laptop computer), categories (*service* or *price* for a restaurant) or a mix of the two. Aspects may be explicitly mentioned in the source text or implicitly inferred using domain-based knowledge [7].

We can deconstruct the ABSA problem into distinct tasks [8]:

- aspect extraction (AE),
- aspect sentiment classification (ASC),
- aspect sentiment evolution (ASE).

AE involves the identification of explicit or implicit aspects of opinion targets in text [7]. While it is common to focus only on aspects (either terms or categories [8]), in [9] the authors go even further to distinguish between the extraction of opinion target entities, the extraction of the corresponding aspects, and the joint extraction of opinion targets and aspects.

When given a set of opinion targets and aspects, ASC assigns a polarity to aspects in text. This is often described as positive, negative, or neutral [10]. Where a text contains multiple aspects for an opinion target, one may also generate an overall sentiment (consistently positive, negative or neutral, or conflicting [11]). By using ASC with text fragments over time, ASE monitors how an author's sentiment towards an opinion target and its aspects evolves [7].

Most papers concentrate exclusively either on the AE or ASC subtasks [8], however recent research integrates AE and ASC in order to implement end-to-end aspect-based sentiment analysis (E2E-ABSA) [12–15]. In contrast with ASC *per se*, E2E-ABSA does not require texts with pre-labelled aspects. For deep neural network-based architectures in particular, joint training of an aspect tagger and sentiment classifier may also have a regularizing effect due to parameter sharing in lower layers, leading to more robust model performance [7].

2.2 ABSA Approaches

Several approaches to ABSA have been pursued in recent years. These may be broadly categorized into information retrieval systems and machine learning-based systems [8].

2.2.1 Information Retrieval Systems

As a common preliminary step, several models tokenize text and extract feature vectors for use with subsequent classifiers [16–18]. Much attention has focused on generating a polarity lexicon (a dictionary of words with known polarity) to classify text sentiment [18–22]. Other classifiers include graphical models such as conditional random fields [16], traditional supervised learning models such as support vector machines [16,21], and deep neural networks [17, 20]. To overcome a dependency on labeled training data, an alternative approach using topic models for AE and ASC has been explored in [22].

2.2.2 Machine Learning Systems

A major limitation of information retrieval ABSA systems is their reliance on manual feature extraction and engineering [6]. Driven by advances in the use of deep learning for natural language processing (NLP), deep neural networks have come to dominate current approaches to ABSA. These architectures may be further classified as follows, with numerous hybrids:

- recursive and recurrent neural networks,

- convolutional neural networks and capsule networks,
- memory networks,
- attention networks,
- transformer-based networks.

Recursive and Recurrent Neural Networks

A recursive neural network (RvNN) considers a sentence's syntactic structures explicitly in order to learn a tree representation of the text [23]. This representation may then be used for effective AE and ASC, but the training process is computationally expensive [6]. In [23] an RvNN is used to identify aspect terms, while in [24] the relational information embedded in the syntactic tree aids ASC.

A conceptually simpler variant of RvNN is the recurrent neural network (RNN) [6]. In an RNN input segments (usually corresponding to tokenized sentences) are fed in sequentially. At each time step the RNN cell performs a non-linear operation on the current input segment and the cell's previous state. The model can thus learn to detect sequential patterns in data, including natural language. Deep RNNs are suitable for sequence tagging in AE and ASC because of their short latency in generating predictions at inference time [25]. Due to the practical difficulties of capturing relations between distantly-separated input segments using RNNs, both [23] and [24] include structural information from an RvNN.

RNN variants such as the long short-term memory network (LSTM) and gated recurrent unit (GRU) are less affected by the practical issues of exploding or vanishing gradients during model training [6]. Both LSTMs and GRUs may be enhanced with downstream layers [26], bi-directional architectures to better capture forward and backward dependencies between tokens in a sentence [10], or non-linear gating functions and an attention mechanism to extract features from the encoded input representation [27, 28].

Convolutional Neural Networks and Capsule Networks

As effective pattern detectors, convolutional neural networks (CNN) have been increasingly applied to natural language problems such as ABSA [6]. CNNs require the input text to be converted into a numerical representation, either through pre-trained word embeddings or engineered feature vectors. The inputs are then fed through a sequence of convolutional operations followed by non-linear pooling methods (such as max pooling) in order to extract and process progressively higher-level features. Certain models have addressed AE [29, 30]

and ASC [31–33] individually, whereas subsequent models have favored an end-to-end approach [15, 34–36]. In particular, the model in [35] employs a routing scheme to transfer relational information for three end tasks (AE, ASC, and target extraction) among the corresponding task-specific CNN layers. We may also consider incorporating structural information from a text's dependency tree, combining a graph-based CNN with a CNN encoder and downstream task-specific layers [33, 36].

To integrate aspect information with the input sentence representation, we may place a gating function immediately after the input encoder layer, consisting of Hadamard multiplication and sigmoid units [28] or ReLU and tanh units [30, 32]. In [37] this approach is extended by using a final attention layer instead of a max pooling layer. By relying on fewer parameters than a transformer-based model this approach can better handle computational resource constraints.

While the CNN architecture can effectively extract features from different layers, the use of pooling operations leads to spatial information loss [38]. To overcome such problems a capsule network uses abstract vector representations of neuron groups that encode the likelihood of an entity existing and its attributes, transforming this information from input to output [39]. Capsule networks have been successfully used for text classification [40] and ASC [17, 41, 42]. Of particular note: capsule networks may be employed to transfer semantic knowledge from the document level to the sentence level [41] and may be used in conjunction with attention mechanisms [42].

Memory Networks

Memory networks contain a cache of external memory which a deep neural network model can access and modify [38]. This architecture has been applied to ASC [20, 43], with some models combining memory networks with attention mechanisms [23, 44].

Attention Networks

Motivated by the concept of attention in humans, an attention mechanism extracts important features from sequential data while ignoring irrelevant information [6]. Attention is often combined with CNNs [15] or RNN architectures such as the LSTM [45–47] or GRU [48] to enhance the base model's performance, however networks based primarily on attention module units have become increasingly prominent in ABSA [48–54].

Among the attention-based approaches to ABSA, several models draw inspiration from other deep learning architectures. In [55] an attention module based on the radial basis

kernel was used to perform fast AE. As discussed for RNN architectures, gating functions may further extract features from an attention network's output [56], while bi-directional attention networks may better detect the influence of previous and future tokens on the present token [52]. Similar to graph convolution networks, graph attention networks incorporate structural information from the input's dependency tree [57–59]. Models with multiple attention modules in parallel (multi-head self-attention, or *MHSA*) appear to perform more effectively than serial attention networks [38], raising interest in their application to ABSA [48–51].

Transformer-based Networks

An extension of the MHSA concept, transformers calculate a self-attention score for each input token to determine the influence on neighboring tokens [38]. The parallel nature of these operations presents a computational advantage over RNNs and CNNs when handling long sentences. This has led to a proliferation of transformer-based models for NLP applications, with the most prominent being pre-trained language models such as OpenGPT [60] and BERT [2]. Since most new approaches to ABSA have concentrated on exploring the capabilities of BERT, we focus on these models in the next section.

2.3 BERT for ABSA

2.3.1 Overview

One of the most consequential developments for NLP systems [10] has been the BERT model proposed in [2], which has enabled significant performance improvements for AE [14, 61–67], ASC [14, 62–65, 68–73], and E2E-ABSA [9, 13, 14, 66, 71]. We will briefly consider the architecture underlying BERT.

As illustrated in Figure 2.1, BERT comprises multiple layers of transformers arranged in a bi-directional layout. The model accepts a sequence of discrete tokens as input, beginning with a dummy token CLS (used to return a classification decision) and with the boundaries of sentences in the original text indicated by the token SEP. As an example, we may observe that the input pair of sentences "This is red. That is blue." has been transformed into a compatible token sequence. Each input token is encoded using three embedding layers (corresponding to the token value, sentence index, and position in the text) to yield a fixed-length input embedding vector. The input embedding vectors are fed into a transformer layer with multiple self-attention modules (twelve for $BERT_{BASE}$, sixteen for $BERT_{LARGE}$).

Additional transformer layers are stacked serially (twelve for $BERT_{BASE}$, twenty-four for $BERT_{LARGE}$), with the output of the final layer generating a contextualized representation of the input text [74]. The output embedding corresponding to the CLS token may be fed into a downstream neural network (e.g. a linear layer) for classification problems, while other problems requiring sequential output may use the entire output embedding sequence.

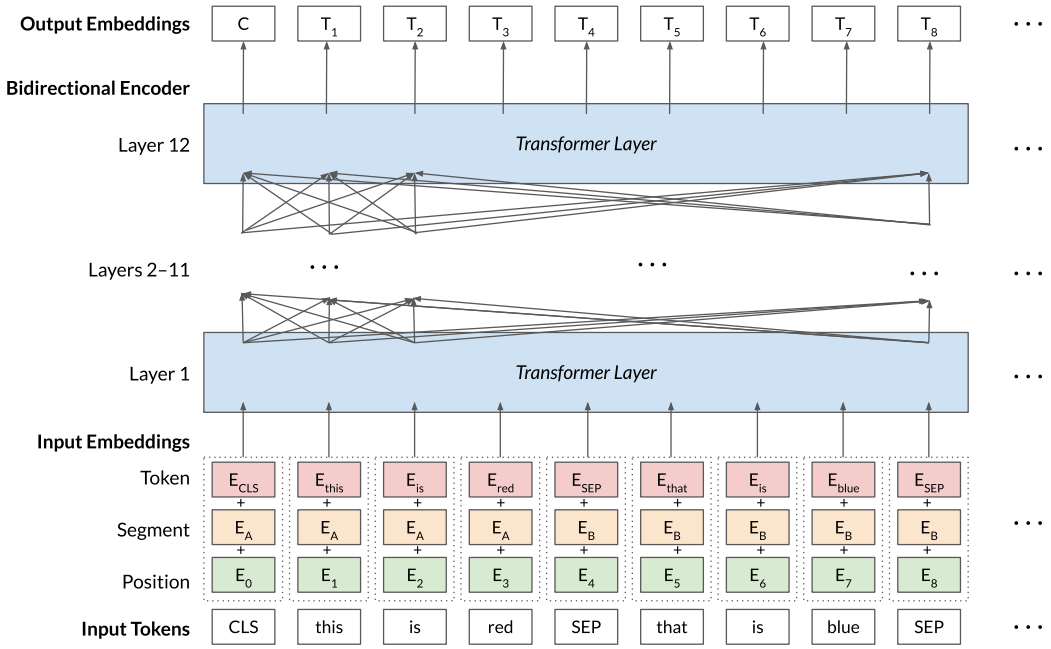


Figure 2.1: $BERT_{BASE}$ Architecture [1, 2]

2.3.2 Training Procedures

In training BERT-based systems, we distinguish between *pre-training*, in which all parameters of the BERT model are optimized for general language tasks, and *fine-tuning*, in which the BERT model's parameters are frozen and the downstream layer optimized for a specific task (e.g. spam email classification) [3, 75].

Pre-Training

As depicted in Figure 2.2, during pre-training BERT acquires a general understanding of language usage through two tasks. In *masked language modeling* (MLM) a random word is masked and must be predicted from context (Figure 2.3a). In *next sentence prediction* (NSP) BERT determines whether two sentences are consecutive (Figure 2.3b).

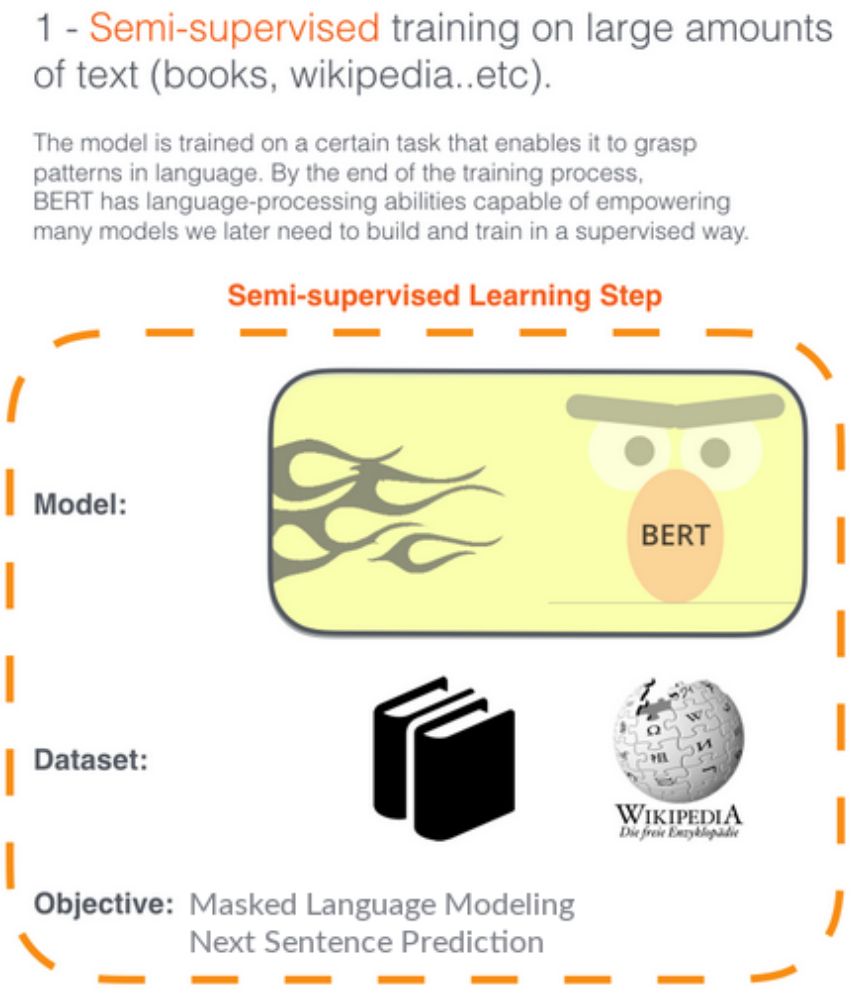
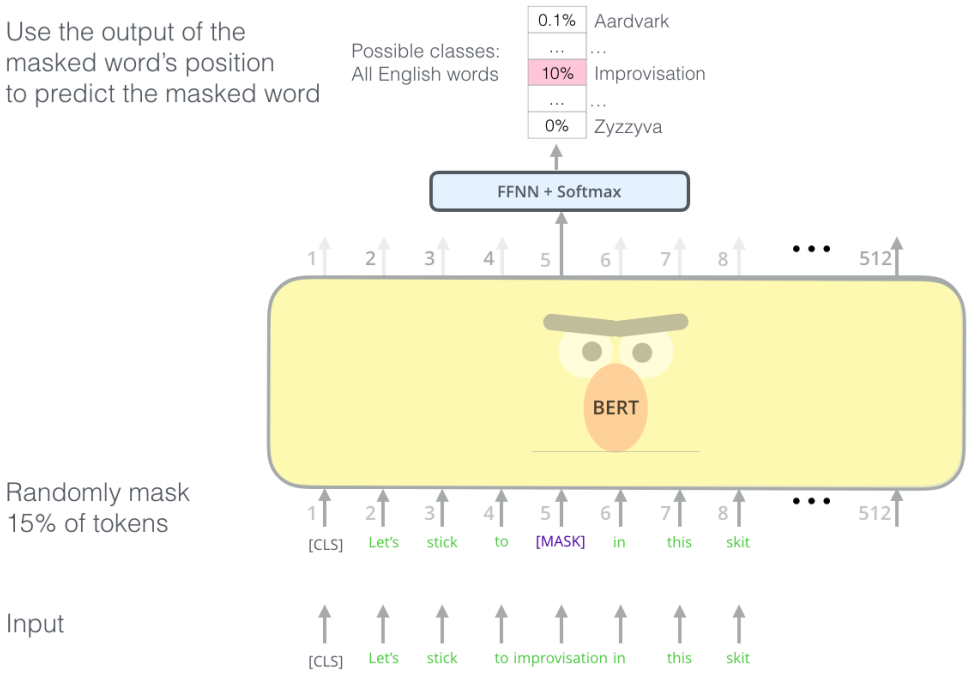
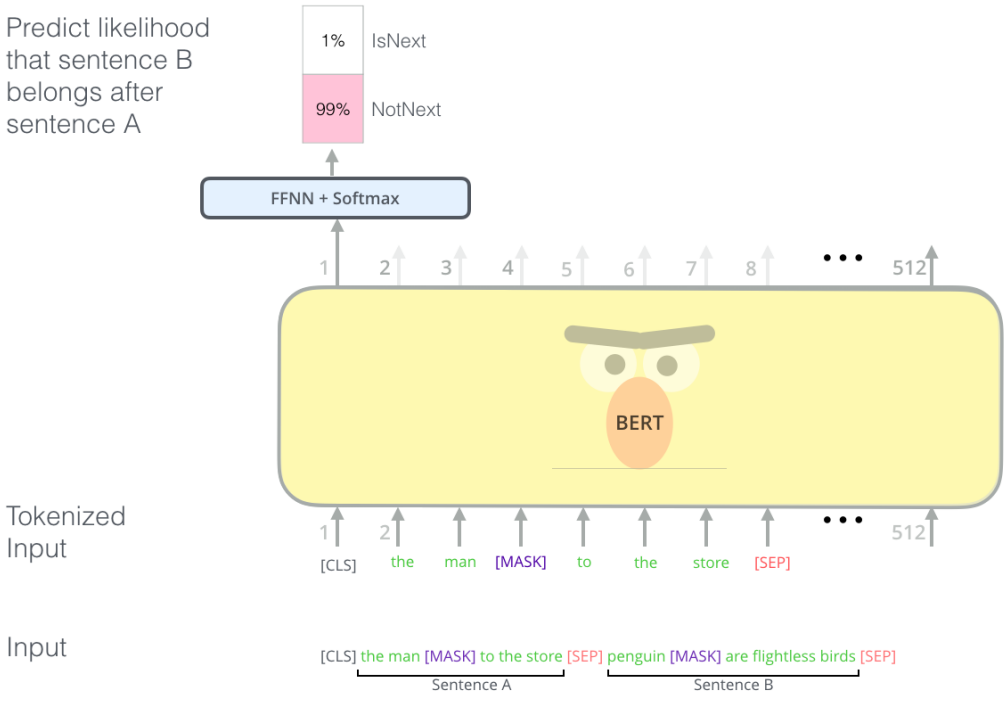


Figure 2.2: Overview of BERT Pre-Training [3]



(a) Masked Language Modeling [3]



(b) Next Sentence Prediction [3]

Figure 2.3: BERT Pre-Training Tasks

Fine-Tuning

Using the BERT language model pre-trained on general text, we may add additional layers to the model output and fine-tune these layers to perform tasks such as text classification or tagging (see Figure 2.4). The simplest models pass the value of the final BERT layer's *CLS* unit (the hidden unit corresponding to the *CLS* token) as the encoded input representation to a linear layer combined with a softmax unit to generate label probabilities [13, 61–64, 67, 68, 70]. As variants of this configuration, intermediate layers' *CLS* unit values may be pooled to yield the encoded representation [65, 71], while the linear layer's output may be fed to non-linear gating functions for further feature extraction [65]. Other instances of downstream task-specific layers may rely on RNNs [13, 52], CNNs [66], capsule networks [73], and attention networks [13, 48, 51, 69].

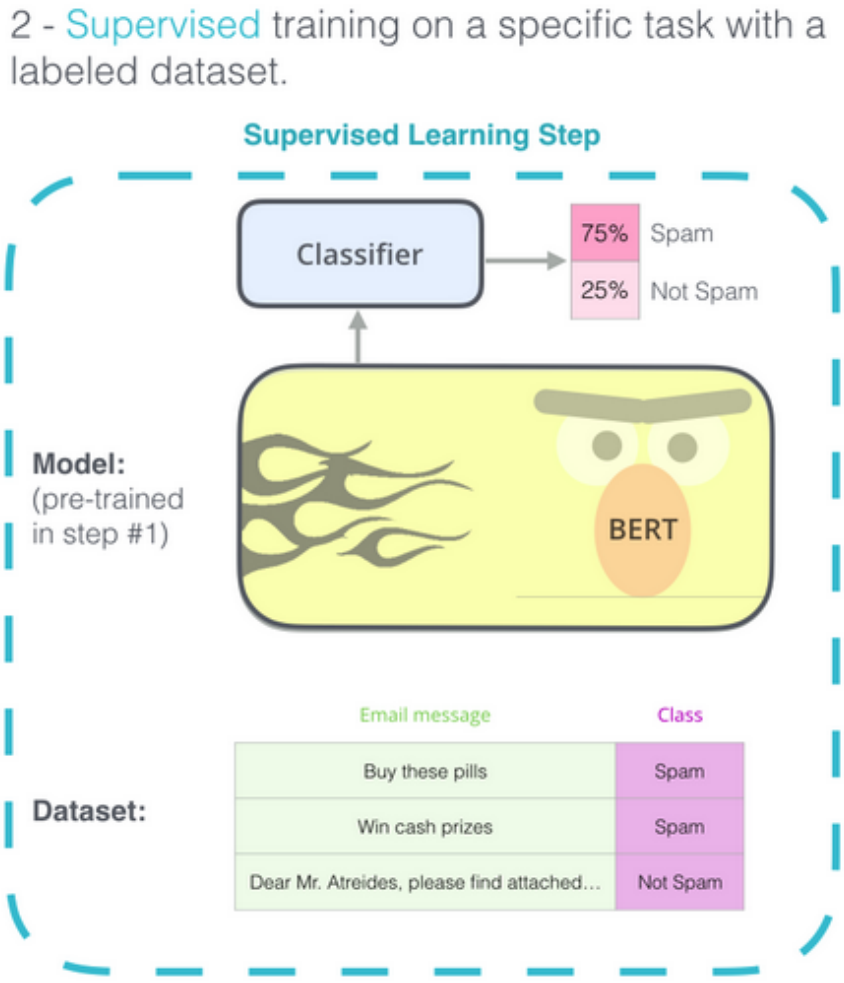


Figure 2.4: Fine-tuning [3]

While most of the BERT-based ABSA models feed a single review sentence to a BERT block with a downstream layer, various alternative structures have been explored. The inclusion of an auxiliary sentence as input may improve performance on ABSA tasks, formulated as specific aspect words [68, 69], aspect words combined with sentiments, or questions [62, 65]. More recently, a self-distillation procedure for learning from both labeled data and a self-ensemble of previous student models has been proposed to enhance the fine-tuning process [76].

2.3.3 Training Data

For both pre-training and fine-tuning, the two most important categories of data are in-domain (containing text from the same domain as the intended domain e.g. laptop reviews), and in-task (containing annotated examples of the intended task e.g. AE). While the two data categories are not mutually exclusive, due to data scarcity it is common to exploit transfer learning by further pre-training on in-task out-of-domain data (unsupervised but more plentiful), before supervised fine-tuning with in-domain data [14, 63, 64, 72, 77]. Further investigations into the specific composition of data for training and fine-tuning have sought to improve performance on domains with scarce data, by performing dynamic data re-balancing [14], data augmentation [78], adversarial training examples [64] or meta-fine-tuning [77].

2.3.4 Computational Considerations

Despite the substantial performance improvements for NLP tasks enabled by BERT, the large number of trainable parameters entails a high computational cost: one study's training of an adapted BERT model using German-language law court decisions used four GPUs over twelve hours [79]. This has motivated the development of derivative language models with smaller memory footprints [38]. DistillBERT uses knowledge distillation during the pre-training stage, allowing for a 40% reduction in the size of the BERT model with 97% of the performance of BERT on the *GLUE* NLP benchmark tests [80].

By making a number of structural changes to BERT, Albert has significantly reduced the number of model parameters (an 89% reduction for the base Albert model compared to $BERT_{BASE}$) without compromising performance [5]. Whereas BERT has used distinct parameters for each transformer layer, Albert has shared the same layer parameters between layers (see Figure 2.5).

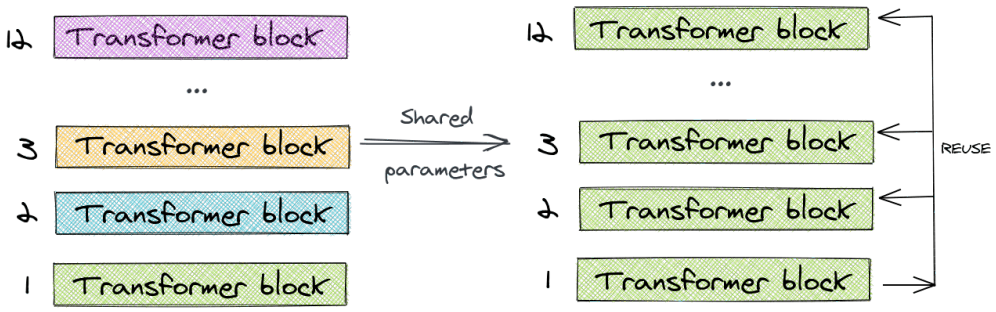
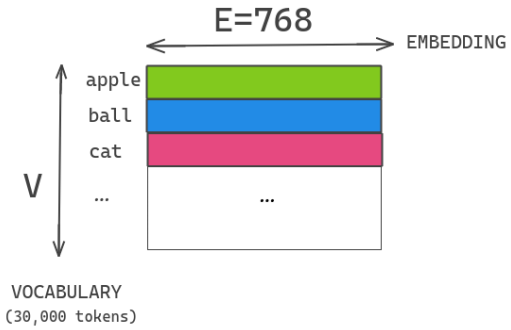
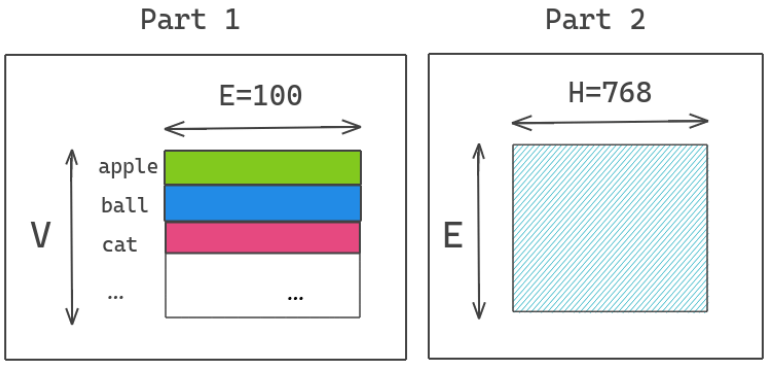


Figure 2.5: Parameter Sharing [4]

To reduce the number of parameters involved in converting input tokens to embeddings, Albert factorizes the single embeddings matrix of size $V \times E_0$ used by BERT (Figure 2.6a) into two smaller matrices (Figure 2.6b). By projecting the vocabulary embeddings first into a lower-dimensional representation ($E_1 = 100$) before scaling up to a higher dimension ($H = 768$), Albert requires much fewer parameters ($V \times E_1 + E_1 \times H \ll V \times E_0$).



(a) BERT Embedding Matrix [4]



(b) Albert Embedding Matrix [4]

Figure 2.6: Embedding Matrices

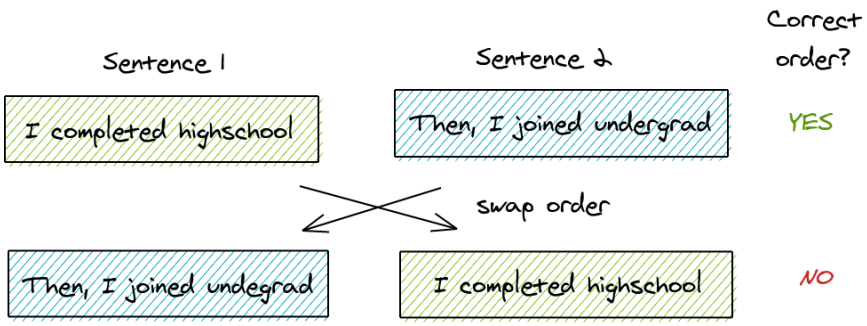


Figure 2.7: Sentence Order Prediction [4]

Finally to encourage the model to learn coherence rather than predict similar topics, the NSP objective has been replaced with a sentence order prediction task, in which the model must decide the correct ordering of two sentences found consecutively in a text (see Figure 2.7). Together these three structural modifications have led to superior performance on three NLP benchmark sets (GLUE [81], SQuAD [82], and RACE [83]) when Albert and BERT have been trained for the same duration.

We note that the use of Albert in place of BERT for ABSA has not been attempted previously. We hypothesize that using an Albert-based model may achieve better performance than BERT-based models while appreciably reducing model size.

2.4 Datasets for ABSA

The development of ABSA systems has been constrained by the availability of suitable data for training and testing. Inspired by standardized datasets for sentiment analysis, the most widely used English datasets for ABSA are the 2014–2016 SemEval Challenge datasets [11, 26, 84, 85], followed by the 2014 Twitter dataset [86] and SentiHood [87]. In the restaurant domain, the newer MAMS dataset seeks to provide more data with multiple aspects and sentiment polarities [73]. For Mandarin, the Camera, Car, Notebook and Phone datasets enable AE, ASC and E2E-ABSA [50]. Smaller datasets continue to be collected for ABSA in order to fill gaps in unexplored languages or domains (e.g. Indonesian tourism [88] and Telugu films [89]). We summarize key attributes of the main datasets in Table 2.1. We note that in SemEval-2014.4 the **Conflict** label indicates aspects with both positive and negative sentiments; in SemEval-2015.12 the hotel domain is used for out-of-domain category ASC, and in SentiHood the **None** label indicates that no sentiments are expressed towards the aspect.

Table 2.1: Attributes of Main ABSA Datasets

Dataset	**Tasks**	**Language**	**Domain**	**Polarities**
SemEval-2014.4	AE (Term) ASC (Term) AE (Category) ASC (Category)	English	Laptop Computers Restaurants	Positive Negative Neutral Conflict
SemEval-2015.12	AE (Category) Opinion Target Extraction ASC (Category)	English	Laptop Computers Restaurants Hotels	Positive Negative Neutral
SemEval-2016.5	AE (Category) Opinion Target Extraction ASC (Category)	English Arabic Mandarin Dutch French Russian Spanish Turkish	Laptop Computers Restaurants Mobile Telephones Digital Cameras Hotels Museums Telecommunications	Positive Negative Neutral
2014 Twitter	ASC	English	Electronics Celebrities Companies	Positive Negative Neutral
SentiHood	AE (Category) ASC (Category)	English	Neighborhoods	Positive Negative None

MAMS	AE (Term) ASC (Term) AE (Category) ASC (Category)	English	Restaurants	Positive Negative Neutral
Unified	E2E-ABSA (Term)	English	Restaurants	Positive Negative Neutral
Camera	AE (Term) ASC (Term) E2E-ABSA (Term)	Mandarin	Cameras	Positive Negative
Car	AE (Term) ASC (Term) E2E-ABSA (Term)	Mandarin	Cars	Positive Negative
Notebook	AE (Term) ASC (Term) E2E-ABSA (Term)	Mandarin	Notebook Computers	Positive Negative
Phone	AE (Term) ASC (Term) E2E-ABSA (Term)	Mandarin	Mobile Telephones	Positive Negative

Chapter 3

Design

Having studied ABSA approaches in literature, we formulate three ABSA problems (AE, ASC, and E2E-ABSA) and propose a solution using adversarial training with the Albert language model ("Albat").

3.1 Problem Formulation

We first formalize the learning problems for the tasks associated with ABSA (AE, ASC, E2E-ABSA). Following the notation of [13], we have considered an input sentence $\mathbf{x}$ of T words. We have denoted the words associated with opinion targets or their aspects (henceforth referred to as *aspect words*) using the index set $\mathbf{i}$. The index set has cardinality $n < T$, i.e. there are n aspect words in $\mathbf{x}$, $\{x_{i_0}, ..., x_{i_n}\}$. Note that sentences may contain compounds of multiple aspect words (e.g. *battery life*), which we term *aspect phrases*.

AE Given the input sequence $\mathbf{x}$, AE seeks to identify the positions of all aspect phrases. We have used the *OBI* labeling scheme, with O indicating non-aspect words, B single-word aspects and words beginning an aspect phrase, and I words otherwise inside an aspect phrase.

ASC Given the input sequence $\mathbf{x}$ and the aspect words $\{x_{i_0}, ..., x_{i_n}\}$, ASC assigns a sentiment polarity to each target word. The sentiment labels may be positive (**POS**), negative (**NEG**), neutral (**NEU**), or conflicting (**CON**).

End-to-End ABSA We may combine AE and ASC so that the aspect phrase position estimates (O,B,I) may be generated jointly with the sentiment labels (*POS*, *NEG*, *NEU*,

CON). This yields a unified label for each word expressing whether a word belongs to an aspect phrase and if so, the associated sentiment polarity [26].

In implementing E2E-ABSA numerous practical considerations may arise. First, it is important to distinguish between opinion targets (which are entities) and aspects (which are attributes). The most simple sentences describe multiple aspects of a single opinion target, however the opinion target itself may not be mentioned explicitly. More challenging sentences may involve multiple opinion targets and one or more aspects [73]. For soundness, a unified labeling scheme must ensure that the same sentiment label is applied to all words within an aspect phrase, and that no sentiment labels are applied with the label *O*.

3.2 Solution Design

3.2.1 Albat

Inspired by the BERT Adversarial Training model described in [64], we have implemented an Albert-based model ("Albat") that exploits adversarial training examples to improve classification performance. Figure 3.1 presents an overview of the Albat model architecture.

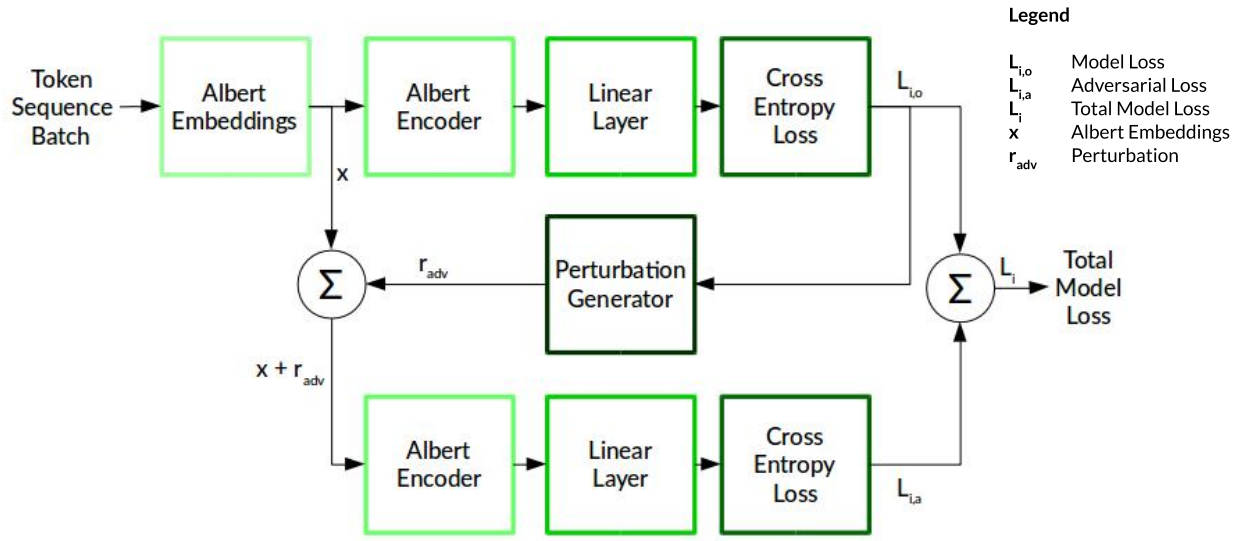


Figure 3.1: Albat Model Overview (with a One-Layer Classifier)

Input Embeddings We first pre-process each input review $\mathbf{x_i}$ into a token sequence $\{[CLS], x_{i_0}, ..., x_{i_n}, [SEP]\}$ of variable length i_n. Note that $[CLS]$ and $[SEP]$ perform the same functions for Albert as for BERT. We pad each token sequence with null tokens to reach a uniform length of $n_s = 100$ tokens and we form training batches of $n_b = 16$ sequences, hence each batch is of size (n_b, n_s). Within each batch, each token is then converted into a numerical representation (known as an *embedding*) comprising three components:

- the token embedding, a numerical vector representation of size $n_e = 128$. The tokens embeddings have been extracted using the SentencePiece software package [90] for English text, and a modified BERT tokenizer [2] for Mandarin text.
- the segment embedding, denoting the token's original sentence;
- the position embedding, indicating the absolute index of the token in the original review text.

The resulting batches of embedding sequences have a uniform size (n_b, n_s, n_e).

Encoder The batches of embedding sequences are fed as inputs to an Albert encoder module. The encoder generates a numerical *hidden representation* of $n_h = 768$ dimensions for each embedding, resulting in an output of size (n_b, n_s, n_h). Whereas in BAT the encoder uses the `bert-base-uncased` transformer architecture for English texts, Albat uses the more recent `albert-base-v2` and `albert_chinese_base` models with 12 attention heads and 12 hidden layers each. The original `albert-base-v1` model achieves a significant reduction in the number of training parameters (and thus memory footprint and training duration) by sharing parameter between layers and factorizing the embedding matrix into smaller sub-matrices [5]. The `albert-base-v2` model is an improved version pre-trained on a larger dataset for more steps and without using dropout, while the `albert_chinese_base` model has been pre-trained further on Mandarin texts.

Classification With $n_l = 9$ possible sequence labels per token, we pass the encoder output to a classification network. In this section we will describe the Albat model variant using a single linear layer of size (n_h, n_l), however the subsequent loss analysis is identical for the two-layer and three-layer model variants (as the final classification layers have the same output dimension n_l). The linear layer generates a logit tensor $\bar{y}_i \in \mathbb{R}^{n_b \times n_s \times n_l}$ containing the unnormalized likelihoods of each label for each token. We then calculate the cross-entropy loss for batch i using the maximum-likelihood label estimates $\hat{y}_i = \arg\max_l \bar{y}_i[\cdot, \cdot, l] \in \mathbb{R}^{n_b \times n_s}$

and ground-truth labels $y_i \in \mathbb{R}^{n_b \times n_s}$ as

$$L_{i,o} = \sum_{b=1}^{n_b} \sum_{s=1}^{n_s} -\ln \left(\frac{\exp\left(\hat{y}[b, s, y[b, s]]\right)}{\sum_{l=1}^{n_l} \exp\left(\hat{y}[b, s, l]\right)} \right) \tag{3.1}$$

$$= \sum_{b=1}^{n_b} \sum_{s=1}^{n_s} -\hat{y}[b, s, y[b, s]] + \ln \sum_{l=1}^{n_l} \exp\left(\hat{y}[b, s, l]\right) \tag{3.2}$$

We can thus calculate the scalar loss value $L_{i,o}$ for each training batch.

Adversarial Attack To increase the robustness of the model to input noise, we may use adversarial perturbations when training the model. As proposed in [64] we employ a "white-box" attack that adds a deterministic perturbation to the original input embedding. In doing so we seek to minimize the model loss for the worst-possible input perturbation, i.e. one which would cause the model to predict an incorrect label. The model can thus learn how to make appropriate predictions given future disturbances via the input.

We first denote the probability of the model generating the label y given the input x and model parameters θ as $p(y|x, \theta)$. We recall from Equation 3.2 that the cross-entropy loss of the model is positively correlated with the log-likelihood of generating an incorrect label $y_{\neg t}$ for an input (since $\sum_{l=1}^{n_l} \exp\left(\hat{y}[\cdot, \cdot, l]\right) \propto p(y_{\neg t}|x, \theta)$). We therefore require the perturbation r_{adv} that satisfies the optimization problem

$$r_{adv} = \underset{r:\|r\| \leq \epsilon}{\arg\min} \ln p(y|x + r, \hat{\theta}) \tag{3.3}$$

We use $\hat{\theta}$ as a constant copy of the model parameters θ to prevent gradient propagation through the adversarial sub-network during training. As described in [91] we may approximate the solution to Equation 3.3 by linearizing $\ln p(y|x + r, \hat{\theta})$ about x. Controlled by the size parameter ϵ, this yields the quantity

$$r_{adv} = -\epsilon \frac{\mathbf{g}}{\|\mathbf{g}\|_2} \tag{3.4}$$

$$\text{where } \mathbf{g} = \nabla_x \ln p(y|x + r, \hat{\theta})$$

which we add to the original embeddings. We then pass the perturbed input through an identical encoder and classification network and calculate the resulting cross-entropy loss $L_{i,a}$. Finally we calculate the total model loss L_i as the sum of the model loss $L_{i,o}$ and the adversarial loss $L_{i,a}$.

3.2.2 Albat-PT

Building on the architecture of Albat, we may modify the underlying language model derived from `albert-base-v2`. Further pre-training the model `bert-base-uncased` using texts from a specific target format (e.g. online reviews) improves performance on downstream ABSA tasks such as AE and ASC [63]. In contrast, the efficacy of further pre-training `albert-base-v2` for downstream tasks has not yet been explored. Doing so may yield similar performance benefits for AE, ASC and E2E-ABSA.

Chapter 4

Results

In this chapter we present details of the implementation of the Albat model architecture and evaluate the performance of Albat variants on ABSA tasks.

4.1 Experimental Setup

4.1.1 Implementation

Code Base We have used the publicly-available code for the BAT model introduced in [64] as a base for developing and fine-tuning Albat models. Due to the retirement of the package `pytorch_pretrained_bert` in favor of the newer `transformers` package, we have needed to rename a number of functions and classes used in the BAT code to avoid execution errors. We have then replaced the tokenizer and encoder associated with `bert-base-uncased` with either those of `albert-base-v2` (for English) or those of `albert_chinese_base` (for Mandarin). For all fine-tuning operations we have used a batch size of 16, a maximum sequence length of 100, and the Adam optimizer algorithm with an initial learning rate of 3×10^{-5} and tuneable weight decay constant λ.

Model Variants and Evaluation Metrics We have introduced three new model classes involving a classifier stage with one, two and three layers respectively. For each model variant we have observed that there is no significant improvement in performance when fine-tuning for more than five epochs. We have therefore initialized five different models, trained each model for five epochs and saved the model parameters with the highest performance for the corresponding validation dataset. When tuning hyperparameters we have reported the average values over five runs for E2E-ABSA (accuracy **ACC** and macro-averaged F_1 score

MF1). When reporting the performance of model variants using optimized hyperparameters we have also reported metrics for ASC (**ACC** and **MF1**) and AE (F_1 score **F1**).

Further Pre-Training To explore the impact of further pre-training on English review texts on the performance of Albat, we have relied on code from the earlier BERT-PT repository [63]. Substituting `albert-base-v2` for `bert-base-uncased`, we have used mixed-precision calculations for further pre-training (specifically 16-bit floating point representation). This has reduced the memory footprint of the model and input data representations, allowing for more training examples per batch and a reduced total run time.

Hardware For all code development and testing we have used the cloud-based Colaboratory development environment, providing access to a remote back-end server with 80 GB of storage space, 12 GB of RAM and a Tesla P100-PCIE-16GB graphical processing unit (GPU). To work around occasional Colaboratory usage restrictions placed on its GPU-enabled servers, we have also had access to a back-end server using a GeForce GTX 1080 Ti GPU provided by Emotech.

4.1.2 Data

Fine-tuning To fine-tune models and evaluate their performance, we have relied primarily on the 2014 SemEval Challenge Task 4 laptop and restaurant datasets (respectively **Lapt14** and **Rest14**) [11]. Since these datasets contain labels only for the AE and ASC tasks, we have used an automated script to generate the corresponding unified labels for E2E-ABSA as described in Section 3.1. These new datasets' properties are shown in Tables 4.1 and 4.2.

Table 4.1: SemEval-2014 Laptop Dataset Properties

Dataset	**Sentences**	**Sentiments**				
		Aspects	**POS**	**NEG**	**NEU**	**CON**
Train	3048	2317	980	841	454	42
Validation	100	49	29	16	4	0
Test	800	650	340	126	168	16
Total	*3948*	*3016*	*1349*	*983*	*626*	*58*

Table 4.2: SemEval-2014 Restaurant Dataset Properties

Dataset	**Sentences**	**Sentiments**				
		Aspects	**POS**	**NEG**	**NEU**	**CON**
Train	3044	3654	2145	789	631	89
Validation	100	96	68	18	10	0
Test	800	1130	725	195	196	14
Total	*3944*	*4880*	*2938*	*1002*	*837*	*103*

After evaluating the impact of various hyperparameter combinations on model performance using the Lapt14 and Rest14 datasets, we have selected the optimal hyperparameter combination for each model architecture (e.g. Albat containing `albert-base-v2` and a one-layer classifier). We have then fine-tuned each model with Lapt14, Rest14 and six additional datasets individually.

Two of the additional six datasets contain English-language data. The **Unified** dataset includes more examples of restaurant domain reviews from the 2014–2016 SemEval Challenges [13], while the harder **MAMS** dataset focuses on reviews involving multiple aspects and multiple sentiments [73]. These datasets' properties are presented in Tables 4.3 and 4.4.

Table 4.3: Unified Dataset Properties

Dataset	**Sentences**	**Sentiments**				
		Aspects	**POS**	**NEG**	**NEU**	**CON**
Train	3490	3834	2306	919	609	0
Validation	387	409	268	93	48	0
Test	2158	2239	1493	489	257	0
Total	*6035*	*6482*	*4067*	*1501*	*914*	*0*

Table 4.4: MAMS Dataset Properties

Dataset	**Sentences**	**Sentiments**				
		Aspects	**POS**	**NEG**	**NEU**	**CON**
Train	4297	11182	3379	2764	5039	0
Validation	500	1332	403	325	604	0
Test	500	1336	400	329	607	0
Total	*5297*	*13850*	*4182*	*3418*	*6250*	*0*

The remaining four datasets contain Mandarin-language data as collated and prepared in [50]. The *Camera*, *Car*, *Notebook*, and *Phone* datasets respectively cover the domains of cameras, cars, notebook computers and mobile telephones. We have evaluated performance on these datasets in order to widen the field of domains and languages under consideration. The properties of these datasets are presented in Tables 4.5–4.8.

Table 4.5: Camera Dataset Properties

Dataset	**Sentences**	**Sentiments**				
		Aspects	**POS**	**NEG**	**NEU**	**CON**
Train	1395	1393	967	426	0	0
Validation	348	345	230	115	0	0
Test	435	434	322	112	0	0
Total	*2178*	*2172*	*1519*	*653*	*0*	*0*

Table 4.6: Car Dataset Properties

Dataset	**Sentences**	**Sentiments**				
		Aspects	**POS**	**NEG**	**NEU**	**CON**
Train	737	738	569	169	0	0
Validation	184	184	140	44	0	0
Test	230	230	164	66	0	0
Total	*1151*	*1152*	*873*	*279*	*0*	*0*

Table 4.7: Notebook Dataset Properties

Dataset	**Sentences**	**Sentiments**				
		Aspects	**POS**	**NEG**	**NEU**	**CON**
Train	398	398	262	136	0	0
Validation	98	98	66	32	0	0
Test	123	123	88	35	0	0
Total	*619*	*619*	*416*	*203*	*0*	*0*

Table 4.8: Phone Dataset Properties

Dataset	**Sentences**	**Sentiments**				
		Aspects	**POS**	**NEG**	**NEU**	**CON**
Train	1592	1587	1053	534	0	0
Validation	396	396	263	133	0	0
Test	497	497	341	156	0	0
Total	*2485*	*2480*	*1657*	*823*	*0*	*0*

Further Pre-Training Using the BERT-PT pre-training script, we have collated reviews from the Amazon [92] and Yelp datasets [63]. Due to memory constraints during pre-training we have needed to use rejection sampling to create two downsampled review corpora: the first contains 1% of review examples, while the second contains 5% of review examples. Even with such adjustments we have observed that pre-training remains a relatively slow and memory-intensive procedure: one epoch of pre-training using 1% of review examples takes approximately two hours using the Tesla P100-PCIE-16GB GPU, while using 5% of review examples takes nearly 12 hours. Due to the shorter time required to complete an epoch of pre-training, we have pre-trained `albert-base-v2` using 1% of the review examples for a maximum of four epochs (as opposed to a maximum of two epochs when pre-training using 5% of the review examples).

4.1.3 Models

We have investigated the performance of three variants of the Albat model architecture. Each variant has an identical structure except for the depth of the classifier stage immediately after the two Albert encoder stages:

- **Albat-1LC** has a single linear layer;
- **Albat-2LC** has two consecutive linear layers, with an optional rectified linear unit (ReLU) between each layer;
- **Albat-3LC** has three consecutive linear layers, with optional ReLU stages between each layer.

We have also sought to compare the performance on English datasets of all three Albat variants using different Albert tokenizer-encoder settings. We have further pre-trained `albert-base-v2` on either 1% of the review corpus for one and four epochs, or 5% for one and two epochs.

4.2 Albat with One-Layer Classifier

Using the default `albert-base-v2` tokenizer and encoder we have first investigated the effects on performance of three hyperparameters: dropout rate δ, perturbation size ϵ, and weight decay constant λ (all are dimensionless). For each hyperparameter we have conducted five runs, in each of which we have fine-tuned the model for five epochs. We have reported the average values over the five runs of accuracy and the macro-averaged F_1 score for both Lapt14 and Rest14.

Using the optimal set of hyperparameters we have then evaluated Albert's performance on the three ABSA tasks (E2E-ABSA, ASC, AE) with the four English datasets (Lapt14, Rest14, Unified, MAMS) and the four Mandarin datasets (Camera, Car, Notebook, Phone).

4.2.1 Hyperparameter Tuning

Dropout Designating the perturbation size $\epsilon = 2$ and the weight decay constant $\lambda = 10^{-1}$, we have evaluated the performance of Albat-1LC with dropout $\delta \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. Results are shown in Table 4.9, with the highest values in boldface.

Table 4.9: Influence of Dropout Rate on Albat-1LC Performance

δ	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
0	**95.09**	52.00	94.96	59.94
0.1	95.07	51.92	94.93	60.04
0.2	95.06	**52.02**	**95.02**	60.22
0.3	95.08	51.90	**95.02**	**60.85**
0.4	95.06	51.65	94.95	60.38
0.5	95.03	51.53	94.87	60.13

Perturbation Size When setting the dropout rate $\delta = 0$ and the weight decay constant $\lambda = 10^{-1}$, we have evaluated the performance of Albat-1LC for perturbation sizes $\epsilon \in \{0, 0.5, 1, 2, 3, 5\}$. Table 4.10 contains the results, with the highest values in boldface.

Table 4.10: Influence of Perturbation Size on Albat-1LC Performance

ϵ	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
0	94.97	51.26	94.38	60.11
0.5	95.07	51.60	94.79	**61.77**
1	95.05	51.65	94.72	60.08
2	**95.09**	**52.00**	**94.96**	59.94
3	**95.09**	51.91	**94.96**	59.86
5	94.83	50.97	**94.96**	60.42

Weight Decay Constant Assigning the dropout rate $\delta = 0$ and the perturbation size $\epsilon = 2$, we have evaluated the performance of Albat-1LC for weight decay constants $\lambda \in \{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. Table 4.11 has the results, with the highest values in boldface.

Table 4.11: Influence of Weight Decay Constant on Albat-1LC Performance

λ	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
0	**95.09**	**52.05**	94.96	60.07
10^{-4}	**95.09**	**52.05**	94.96	60.07
10^{-3}	**95.09**	**52.05**	94.96	60.07
10^{-2}	**95.09**	51.99	94.96	**60.18**
10^{-1}	94.73	49.91	94.90	60.04
1	95.07	51.86	**94.99**	60.14

4.2.2 Optimized Model Results

Despite small improvements in performance for some metrics with non-zero dropout rates in Table 4.9, we have decided to use zero dropout. This is supported by the observation in [5] that non-zero dropout rates do not necessarily improve model robustness for the Albert family of language models. Table 4.10 demonstrates that a perturbation size of 2 yields the highest value for three of the four metrics. In Table 4.11 a number of weight decay constant values are associated with comparable performance on the Lapt14 metrics but the value 10^{-2} yields better performance on the Rest14 macro-averaged F_1 score.

With $\delta = \mathbf{0}$, $\epsilon = \mathbf{2}$ and $\lambda = \mathbf{10^{-2}}$, Table 4.12 presents the performance of Albat-1LC for E2E-ABSA using all datasets and additional metrics for AE and ASC. Note that we have used `albert-base-v2` for English-language datasets and `albert_chinese_base` for Mandarin-language datasets.

Table 4.12: Performance of Albat-1LC using Optimized Hyperparameters

Dataset	**E2E-ABSA**		**ASC**		**AE**
	ACC	**MF1**	**ACC**	**MF1**	**F1**
Lapt14	95.09	51.99	79.87	77.39	87.45
Rest14	94.96	60.18	85.02	77.57	80.51
Unified	95.11	63.91	88.31	76.42	83.95
MAMS	92.24	66.32	84.30	83.60	78.77
Camera	88.90	80.45	94.10	92.38	88.45
Car	85.21	47.14	94.00	92.48	88.77
Notebook	69.05	30.59	92.68	91.56	87.60
Phone	90.20	85.58	96.22	95.61	89.99

4.3 Albat with Two-Layer Classifier

Using the default `albert-base-v2` tokenizer and encoder we have investigated the effects on performance of perturbation size ϵ, the weight decay constant λ, and the first classification layer's output dimension. For each hyperparameter we have conducted five runs, in each of which we have fine-tuned the model for five epochs. We report the average values over the five runs of accuracy and the macro-averaged F_1 score for both Lapt14 and Rest14.

Using the optimal set of hyperparameters we have then evaluated the model's performance on the three ABSA tasks (E2E-ABSA, ASC, AE) with the eight datasets (Lapt14, Rest14, Unified, MAMS, Camera, Car, Notebook, Phone).

4.3.1 Hyperparameter Tuning

Perturbation Size Selecting the dropout rate $\delta = 0$, the weight decay constant $\lambda = 10^{-1}$, and the first classification layer's output dimension to 1152 we have evaluated the performance of Albat-2LC for perturbation sizes $\epsilon \in \{0, 0.5, 1, 2, 3, 5\}$ either with or without a ReLU stage between the two classification layers. Tables 4.13 and 4.14 show the respective results.

Table 4.13: Influence of Perturbation Size on Albat-2LC Performance (without ReLU)

ϵ	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
0	94.57	48.85	94.97	60.36
0.5	95.03	51.34	**95.08**	**61.30**
1	95.00	51.29	94.17	53.90
2	95.07	51.76	94.99	60.39
3	**95.09**	**51.91**	94.96	59.86
5	94.97	51.52	94.75	59.12

Table 4.14: Influence of Perturbation Size on Albat-2LC Performance (using ReLU)

ϵ	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
0	94.57	48.85	94.97	60.36
0.5	95.03	51.34	**95.08**	**61.30**
1	95.00	51.29	94.17	53.90
2	95.07	51.76	94.99	60.39
3	**95.09**	**51.91**	94.96	59.86
5	94.97	51.52	94.75	59.12

Weight Decay Constant Designating the dropout rate $\delta = 0$, the perturbation size $\epsilon = 2$, and the first classification layer's output dimension to 1152 we have evaluated the performance of Albat-2LC for weight decay constants $\lambda \in \{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$ with or without a ReLU stage between the classification layers. We present the results in Tables 4.15 and 4.16.

Table 4.15: Influence of Weight Decay Constant on Albat-2LC Performance (without ReLU)

λ	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
0	95.05	**51.73**	94.83	59.95
10^{-4}	95.05	**51.73**	94.83	59.95
10^{-3}	95.05	**51.73**	94.83	59.95
10^{-2}	94.96	51.31	**95.03**	**60.26**
10^{-1}	**95.07**	**51.73**	94.83	59.95
1	95.02	51.50	94.98	60.25

Table 4.16: Influence of Weight Decay Constant on Albat-2LC Performance (using ReLU)

λ	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
0	95.05	51.73	94.83	59.95
10^{-4}	95.05	51.73	94.83	59.95
10^{-3}	95.05	51.73	94.83	59.95
10^{-2}	94.96	51.31	**95.03**	60.26
10^{-1}	**95.07**	**51.76**	94.99	**60.39**
1	95.02	51.50	94.98	60.25

First Classification Layer Output Dimension Setting the dropout rate $\delta = 0$, the perturbation size $\epsilon = 2$, and the weight decay constant $\lambda = 10^{-2}$ we have evaluated the performance of Albat-2LC for the first classification layer output dimension selected from $\{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\} \times 768$. Results for the model with an intermediate ReLU stage and without one are contained in Tables 4.17 and 4.18 respectively.

Table 4.17: Influence of First Classification Layer Output Dimension on Albat-2LC Performance (without ReLU)

First Output Dimension	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
384	94.26	45.78	94.65	56.89
768	95.08	51.68	94.98	59.57
1152	95.07	51.76	94.99	60.39
1536	95.06	51.56	**95.04**	60.46
1920	**95.16**	**51.89**	94.95	60.90
2304	94.40	45.92	95.02	**61.69**

Table 4.18: Influence of First Classification Layer Output Dimension on Albat-2LC Performance (using ReLU)

First Output Dimension	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
384	94.26	45.78	94.65	56.89
768	95.08	51.68	94.98	59.57
1152	95.07	51.76	94.99	60.39
1536	95.06	51.56	**95.04**	60.46
1920	**95.16**	**51.89**	94.95	60.90
2304	94.40	45.92	95.02	**61.69**

4.3.2 Optimized Model Results

Tables 4.13 and 4.14 indicate that regardless of whether a ReLU stage is present, using $\epsilon = 3$ yields the highest performance on Lapt14, while using $\epsilon = 0.5$ yields the highest performance on Rest14. A suitable compromise is $\epsilon = 2$, for which we trade a decrease in performance on Lapt14 for better performance on Rest14. In Table 4.15 a number of weight decay constant values perform comparably on Lapt14, however in Table 4.16 the value $\lambda = 10^{-1}$ yields better performance on both Lapt14 and Rest14 when a ReLU stage is used. Setting the first classification layer output dimension to $2.5 \times 768 = 1920$ yields the highest performance on Lapt14 and second-highest performance on the Rest14 macro-averaged F_1 score regardless of whether a ReLU stage is present.

Table 4.19 displays the performance of Albat-2LC with $\delta = \mathbf{0}$, $\epsilon = \mathbf{2}$, $\lambda = \mathbf{10^{-1}}$, and a ReLU stage in place.

Table 4.19: Performance of Albat-2LC using Optimized Hyperparameters

Dataset	**E2E-ABSA**		**ASC**		**AE**
	ACC	**MF1**	**ACC**	**MF1**	**F1**
Lapt14	95.16	51.89	78.71	76.48	87.71
Rest14	94.95	60.90	84.96	77.32	81.51
Unified	95.23	65.25	87.73	75.71	83.69
MAMS	92.23	66.25	76.63	71.23	78.20
Camera	89.97	82.35	94.06	92.30	88.17
Car	86.61	48.82	93.83	92.26	88.40
Notebook	76.66	44.61	92.03	90.84	88.68
Phone	90.66	86.28	96.22	95.59	89.64

4.4 Albat with Three-Layer Classifier

We have investigated the effects on performance of four hyperparameters (perturbation size ϵ, the weight decay constant λ, the first and the second classification layers' output dimensions) by utilizing the default `albert-base-v2` tokenizer and encoder. For each hyperparameter we have conducted five runs, in each of which we have fine-tuned the model for five epochs. The average values of accuracy and the macro-averaged F_1 score over the five runs for both Lapt14 and Rest14 have been reported.

We have then evaluated performance on the three ABSA tasks (E2E-ABSA, ASC, AE) with the eight datasets and the optimal hyperparameters.

4.4.1 Hyperparameter Tuning

Perturbation Size Setting the dropout rate $\delta = 0$, the weight decay constant $\lambda = 10^{-1}$, the first and second classification layers' output dimensions to 1152 and 1536 respectively, we have evaluated the performance of Albat-3LC for perturbation sizes $\epsilon \in \{0, 0.5, 1, 2, 3, 5\}$ either with or without ReLU stages between each of the three classification layers. See Tables 4.20 and 4.21 respectively.

Table 4.20: Influence of Perturbation Size on Albat-3LC Performance (without ReLU)

ϵ	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
0	94.59	48.50	94.69	59.48
0.5	94.90	50.43	94.83	59.67
1	95.04	51.37	94.96	60.36
2	**95.14**	**51.91**	**95.03**	**60.84**
3	95.08	51.78	94.87	59.86
5	94.98	51.26	94.87	59.75

Table 4.21: Influence of Perturbation Size on Albat-3LC Performance (using ReLU)

ϵ	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
0	94.59	48.50	94.69	59.48
0.5	94.90	50.43	94.83	59.67
1	95.04	51.37	94.96	60.36
2	**95.14**	**51.91**	**95.03**	**60.84**
3	95.08	51.78	94.87	59.75
5	94.98	51.26	94.87	59.75

Weight Decay Constant Applying the dropout rate $\delta = 0$, the perturbation size $\epsilon = 2$, the first and second classification layers' output dimensions to 1152 and 1536 respectively, we have evaluated the performance of Albat-3LC for weight decay constants $\lambda \in \{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$ either with or without ReLU stages between each of the three classification layers. See Tables 4.22 and 4.23 respectively.

Table 4.22: Influence of Weight Decay Constant on Albat-3LC Performance (without ReLU)

λ	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
0	95.07	51.47	95.03	60.28
10^{-4}	95.07	51.47	95.03	60.28
10^{-3}	94.87	50.26	94.92	60.52
10^{-2}	**95.17**	**52.08**	**95.13**	**64.00**
10^{-1}	95.12	51.93	94.92	61.98
1	95.07	51.84	**95.13**	62.01

Table 4.23: Influence of Weight Decay Constant on Albat-3LC Performance (using ReLU)

λ	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
0	95.07	51.47	**95.03**	60.28
10^{-4}	95.07	51.47	**95.03**	60.28
10^{-3}	95.07	51.47	**95.03**	60.28
10^{-2}	95.02	51.12	94.77	58.55
10^{-1}	**95.14**	**51.91**	**95.03**	**60.84**
1	95.06	51.50	94.95	59.91

First Classification Layer Output Dimension Designating the dropout rate $\delta = 0$, the perturbation size $\epsilon = 2$, the weight decay constant $\lambda = 10^{-1}$, and the second classification layer's output dimension to 1536, we have evaluated the performance of Albat-3LC for the first classification layer output dimensions selected from $\{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\} \times 768$, either with or without ReLU stages between each of the three classification layers. See Tables 4.24 and 4.25 respectively.

Table 4.24: Influence of First Classification Layer Output Dimension on Albat-3LC Performance (without ReLU)

First Output Dimension	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
384	95.01	51.32	94.87	60.04
768	94.94	50.63	95.07	60.86
1152	95.12	51.93	94.92	61.98
1536	95.05	51.98	95.04	62.58
1920	95.10	51.90	95.06	62.64
2304	**95.17**	**52.29**	**95.08**	**64.35**

Table 4.25: Influence of First Classification Layer Output Dimension on Albat-3LC Performance (using ReLU)

First Output Dimension	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
384	94.21	47.09	94.47	52.98
768	94.72	49.45	94.90	60.08
1152	95.14	51.91	95.03	60.84
1536	95.05	51.46	94.91	59.94
1920	95.05	51.46	94.97	59.91
2304	**95.15**	**52.01**	**95.10**	**61.02**

Second Classification Layer Output Dimension Employing the dropout rate $\delta = 0$, the perturbation size $\epsilon = 2$, the weight decay constant $\lambda = 10^{-1}$, and the first classification layer's output dimension to 1152, we have evaluated the performance of Albat-3LC for the second classification layer output dimensions selected from $\{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\} \times 768$, either with or without ReLU stages between each of the three classification layers. See Tables 4.26 and 4.27 respectively.

Table 4.26: Influence of Second Classification Layer Output Dimension on Albat-3LC Performance (without ReLU)

Second Output Dimension	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
384	95.03	51.34	94.99	60.49
768	**95.17**	52.09	94.98	59.59
1152	94.98	51.13	95.00	61.87
1536	95.12	51.93	94.92	61.98
1920	95.03	51.74	95.04	63.07
2304	**95.17**	**52.19**	**95.05**	**63.23**

Table 4.27: Influence of Second Classification Layer Output Dimension on Albat-3LC Performance (using ReLU)

Second Output Dimension	**Lapt14**		**Rest14**	
	ACC	**MF1**	**ACC**	**MF1**
384	93.65	39.97	94.54	54.67
768	94.83	50.05	94.65	56.61
1152	94.89	50.63	94.71	57.91
1536	**95.14**	**51.91**	95.03	**60.84**
1920	95.12	**51.91**	**95.15**	60.77
2304	95.02	51.46	94.96	60.17

4.4.2 Optimized Model Results

Tables 4.20 and 4.21 demonstrate that whether or not a ReLU stage is present, using $\epsilon = 2$ yields the highest performance on both Lapt14 and Rest14. Comparing Tables 4.22 and 4.23, the use of $\lambda = 10^{-2}$ without ReLU stages outperforms the use of $\lambda = 10^{-1}$ with ReLU stages on both Lapt14 and Rest14. Finally, with all other variables held constant in Tables 4.24 – 4.27, the combination of setting the first and second classification layer output dimensions to 2304 and 1536 respectively without ReLU stages performs best on Lapt14 and Rest14 (see Table 4.24).

Table 4.28 presents the performance of Albat-3LC with $\delta = \mathbf{0}$, $\epsilon = \mathbf{2}$, $\lambda = \mathbf{10^{-2}}$, the first and second classification layer output dimensions set to 2304 and 1536 respectively, and no ReLU stages.

Table 4.28: Performance of Albat-3LC using Optimized Hyperparameters

Dataset	**E2E-ABSA**		**ASC**		**AE**
	ACC	**MF1**	**ACC**	**MF1**	**F1**
Lapt14	95.17	52.16	78.43	75.87	88.06
Rest14	95.10	63.27	78.43	75.87	80.69
Unified	95.08	64.27	87.25	72.33	83.38
MAMS	92.04	65.13	84.37	83.76	78.59
Camera	88.35	76.12	93.64	91.73	87.97
Car	86.73	53.19	90.52	83.68	88.52
Notebook	77.81	56.80	90.89	89.37	88.86
Phone	90.27	85.89	96.62	96.06	89.91

4.5 Albat with Further Pre-training

Having established the combinations of hyperparameters that yield optimal performance for Albat-1LC, Albat-2LC, and Albat-3LC, we have studied the impact of further pre-training the Albert tokenizer-encoder pair on the performance of the whole network. For each of the three Albat model variants we have compared the performance of the default `albert-base-v2` tokenizer-encoder settings with those of four further pre-trained model settings, trained with 1% of the review corpus for 1 epoch and 4 epochs, and with 5% of the review corpus for 1 epoch and 2 epochs.[1] Due to memory limitations we were unable to conduct further pre-training using 10% of the corpus.

4.5.1 Albat-1LC

Comparing the results of the `albert-base-v2` model in Table 4.12 with those of the further pre-trained models in Tables 4.29–4.32, we have observed that further pre-training was detrimental to performance on ABSA tasks. In particular, a marked reduction in the E2E-ABSA macro-averaged F_1 score for each dataset has suggested that further pre-training Albert has caused "catastrophic forgetting" of existing knowledge as described in [93].[2]

Table 4.29: Performance of Albat-1LC (PT using 1% Data for 1 Epoch)

Dataset	**E2E-ABSA**		**ASC**	
	ACC	**MF1**	**ACC**	**MF1**
Lapt14	89.97	10.52	46.77	20.80
Rest14	84.65	11.82	65.00	26.26

Table 4.30: Performance of Albat-1LC (PT using 1% Data for 4 Epochs)

Dataset	**E2E-ABSA**		**ASC**	
	ACC	**MF1**	**ACC**	**MF1**
Lapt14	89.97	10.52	53.45	23.22
Rest14	84.79	12.76	65.00	26.26

[1] While in [63] numerous pre-training methods are proposed, we have only found publicly released code for the BERT-DK method.

[2] Notably this phenomenon has been observed for other text classification tasks using BERT models [94].

Table 4.31: Performance of Albat-1LC (PT using 5% Data for 1 Epoch)

Dataset	E2E-ABSA		ASC	
	ACC	MF1	ACC	MF1
Lapt14	89.97	10.52	58.37	42.70
Rest14	84.71	11.47	72.30	45.03

Table 4.32: Performance of Albat-1LC (PT using 5% Data for 2 Epochs)

Dataset	E2E-ABSA		ASC	
	ACC	MF1	ACC	MF1
Lapt14	89.97	10.52	55.86	36.07
Rest14	84.71	11.47	66.71	30.80

We have observed different patterns in performance for each of the ABSA tasks:

- **E2E-ABSA**: there was no change in performance when further pre-training Albert for more than one epoch.
- **AE**: it was not possible to generate an F_1 score for any of the further pre-trained models.
- **ASC**: when pre-training using 1% of the review corpus beyond one epoch, there was a slight improvement in Lapt14 metrics but no improvement in Rest14 metrics. When pre-training using 5% of the review corpus beyond one epoch, both domains' metrics deteriorated.

It is likely that further pre-training Albert on a larger subset of the review corpus may not improve performance on any of the three ABSA tasks.

4.5.2 Albat-2LC

In comparing the results of the `albert-base-v2` model in Table 4.19 with those of the further pre-trained models in Tables 4.33–4.36, we have observed that further pre-training was detrimental to performance on ABSA tasks:

- **E2E-ABSA**: there was no change in performance when further pre-training Albert for more than one epoch.
- **AE**: it was not possible to generate an F_1 score for any of the further pre-trained models.

Table 4.33: Performance of Albat-2LC (PT using 1% Data for 1 Epoch)

Dataset	E2E-ABSA		ASC	
	ACC	MF1	ACC	MF1
Lapt14	89.97	10.52	40.09	18.39
Rest14	84.71	11.47	65.00	26.26

Table 4.34: Performance of Albat-2LC (PT using 1% Data for 4 Epochs)

Dataset	E2E-ABSA		ASC	
	ACC	MF1	ACC	MF1
Lapt14	89.97	10.52	40.09	18.39
Rest14	84.71	11.47	65.00	26.26

Table 4.35: Performance of Albat-2LC (PT using 5% Data for 1 Epoch)

Dataset	E2E-ABSA		ASC	
	ACC	MF1	ACC	MF1
Lapt14	89.97	10.52	40.09	18.39
Rest14	84.71	11.47	65.00	26.26

Table 4.36: Performance of Albat-2LC (PT using 5% Data for 2 Epochs)

Dataset	E2E-ABSA		ASC	
	ACC	MF1	ACC	MF1
Lapt14	89.97	10.52	49.37	36.49
Rest14	84.71	11.47	67.64	35.88

- **ASC**: there was no change in performance when pre-training using 1% of the review corpus beyond one epoch. When pre-training using 5% of the review corpus beyond one epoch, both domains' metrics have improved.

Comparing the results for pre-training using 1% and 5% of the review corpus, we may speculate that further pre-training Albert on a larger subset of the review corpus may improve the performance of Albat-2LC on ASC but not E2E-ABSA or AE.

4.5.3 Albat-3LC

Comparing the results of the `albert-base-v2` model in Table 4.28 with those of the further pre-trained models in Tables 4.37–4.40, we have again observed that further pre-training was detrimental to performance on ABSA tasks:

Table 4.37: Performance of Albat-3LC (PT using 1% Data for 1 Epoch)

Dataset	**E2E-ABSA**		**ASC**	
	ACC	**MF1**	**ACC**	**MF1**
Lapt14	89.97	10.52	20.06	11.14
Rest14	84.71	11.47	65.00	26.26

Table 4.38: Performance of Albat-3LC (PT using 1% Data for 4 Epochs)

Dataset	**E2E-ABSA**		**ASC**	
	ACC	**MF1**	**ACC**	**MF1**
Lapt14	89.97	10.52	20.06	11.14
Rest14	84.71	11.47	65.00	26.26

Table 4.39: Performance of Albat-3LC (PT using 5% Data for 1 Epoch)

Dataset	**E2E-ABSA**		**ASC**	
	ACC	**MF1**	**ACC**	**MF1**
Lapt14	89.97	10.52	50.94	34.12
Rest14	84.71	11.47	65.00	26.26

Table 4.40: Performance of Albat-3LC (PT using 5% Data for 2 Epochs)

Dataset	**E2E-ABSA**		**ASC**	
	ACC	**MF1**	**ACC**	**MF1**
Lapt14	89.97	10.52	52.10	40.57
Rest14	84.71	11.47	68.86	38.84

- **E2E-ABSA**: there was no change in performance when further pre-training Albert for more than one epoch.
- **AE**: it was not possible to generate an F_1 score for any of the further pre-trained models.
- **ASC**: there was no change in performance when pre-training using 1% of the review corpus beyond one epoch. When pre-training using 5% of the review corpus beyond one epoch, both domains' metrics have improved slightly.

Comparing the results for pre-training using 1% and 5% of the review corpus, we may speculate that further pre-training Albert on a larger subset of the review corpus would improve the performance of Albat-3LC on ASC but not E2E-ABSA or AE.

Chapter 5

Discussion

Experimental outcomes within the context of results found in literature are discussed, a case study is employed to demonstrate our methodology, confusion matrices are generated to study patterns in model error types and frequencies, and we analyze the various Albat models' resource usage.

5.1 Model Performance

5.1.1 Metrics

Eight datasets have been used to evaluate Albat model performance:

- English-language datasets:
  - **Lapt14** is a standard dataset for evaluating performance on AE and ASC for the laptop computer domain;
  - **Rest14** is a standard dataset for evaluating performance on AE and ASC for the restaurant domain;
  - **Unified** is a standard dataset for evaluating performance on E2E-ABSA, containing a larger sample of restaurant-domain reviews;
  - **MAMS** contains more complex restaurant-domain reviews, with multiple entities, aspects or sentiments per review.

- Mandarin-language datasets:
  - **Camera** is a standard dataset for evaluating performance on AE and ASC for the camera domain;
  - **Car** is a standard dataset for evaluating performance on AE and ASC for the car domain;
  - **Notebook** is a standard dataset for evaluating performance on AE and ASC for the notebook computer domain;
  - **Phone** is a standard dataset for evaluating performance on AE and ASC for the mobile telephone domain.

Standardized metrics associated with each of the ABSA tasks (E2E-ABSA, AE and ASC) have been employed. We will discuss the rationale behind the choice of metrics for each task below.

E2E-ABSA The evaluation of Albat model variants on E2E-ABSA involving all datasets has been one of our primary objectives. E2E-ABSA is a single-label multi-class classification task, so an average of each class's F_1 score has been an appropriate metric for performance evaluation [26]. We have reported both the micro-averaged F_1 score (here equivalent to accuracy[1]) and the macro-averaged F_1 score for each dataset, corresponding to a total of sixteen evaluation metrics:

- Lapt14 accuracy and macro-averaged F_1 score,
- Rest14 accuracy and macro-averaged F_1 score,
- Unified accuracy and macro-averaged F_1 score,
- MAMS accuracy and macro-averaged F_1 score,
- Camera accuracy and macro-averaged F_1 score,
- Car accuracy and macro-averaged F_1 score,
- Notebook accuracy and macro-averaged F_1 score,
- Phone accuracy and macro-averaged F_1 score.

[1]See Appendix C for an explanation of the different averaging methods used with the F_1 score.

AE and ASC We have also sought to evaluate the performance of the Albat model variants on each dataset. AE is a binary classification task (a token is either part of an aspect or not), with the F_1 score for aspect words as the literature's standard metric. This has generated eight evaluation metrics:

- Lapt14 F_1 score,
- Rest14 F_1 score,
- Unified F_1 score,
- MAMS F_1 score,
- Camera F_1 score,
- Car F_1 score,
- Notebook F_1 score,
- Phone F_1 score.

Similar to E2E-ABSA, ASC is a single-label multi-class classification task with accuracy and the macro-averaged F_1 score as the standard metrics reported in literature. We have reported sixteen evaluation metrics:

- Lapt14 accuracy and macro-averaged F_1 score,
- Rest14 accuracy and macro-averaged F_1 score,
- Unified accuracy and macro-averaged F_1 score,
- MAMS accuracy and macro-averaged F_1 score,
- Camera accuracy and macro-averaged F_1 score,
- Car accuracy and macro-averaged F_1 score,
- Notebook accuracy and macro-averaged F_1 score,
- Phone accuracy and macro-averaged F_1 score.

Literature Results We have been able to find results in literature for twenty-one of these forty metrics:

- E2E-ABSA accuracy for Lapt14 and Unified;
- AE F_1 score for Lapt14, Rest14, Camera, Car, Notebook, and Phone;
- ASC accuracy and macro-averaged F_1 score for Lapt14, Rest14, Camera, Car, Notebook, and Phone;
- ASC accuracy for MAMS.

The optimized Albat model variants exceed state of the art results in five of the twenty-one metrics for which literature values have been found (both E2E-ABSA metrics, one of the thirteen ASC metrics and two of the six AE metrics). Note that since all Albat model variants have used hyperparameters tuned with Lapt14 and Rest14, it is possible that the performance of Albat model variants fine-tuned with the Mandarin datasets has been sub-optimal. Additional performance gains may be achieved by tuning these models' hyperparameters using the Mandarin datasets instead of Lapt14 and Rest14.

Model Identification For greater clarity, in Figures 5.1–5.12 we have used a numerical scheme supplemented with a color scheme to identify each model uniquely:

1. Albat-1LC
2. Albat-2LC
3. Albat-3LC
4. BAT [64]
5. DomBERT [14]
6. BERT + SAN [13]
7. CapsNet-BERT [73]
8. XLNetCN-AS-AP-add [95]
9. LCF-ATEPC-CDW [50]
10. LCF-ATEPC-CDM [50]
11. LCF-ATEPC-Fusion [50]

5.1.2 E2E-ABSA

In the E2E-ABSA literature we have observed that model performance in [26] and [13] has been reported using micro-averaged F_1 scores for each domain (cf. Appendix B). Since these results have been included in [14] under the same performance metric, it is reasonable to assume that [14] has also reported micro-averaged F_1 scores (equivalent to accuracy, see Appendix C). Figures 5.1 and 5.2 have been collated under that assumption.

In Figure 5.1, Albat-3LC has achieved state of the art accuracy for Lapt14, while in Figure 5.2 Albat-2LC has achieved state of the art accuracy for Unified.

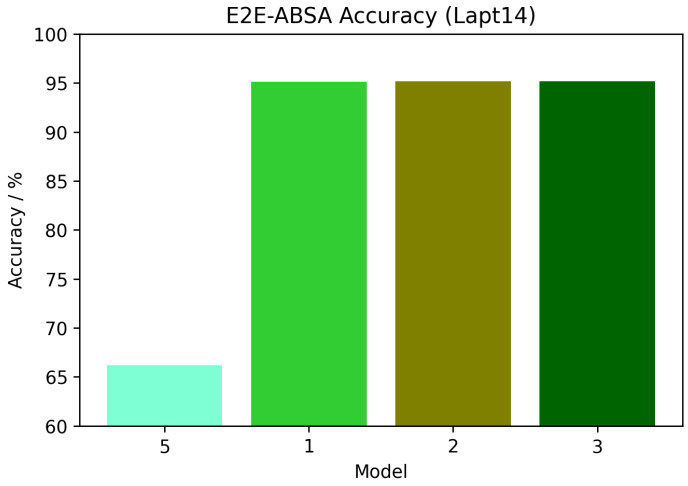


Figure 5.1: Performance Comparison with State of the Art for E2E-ABSA on Lapt14 (Accuracy)

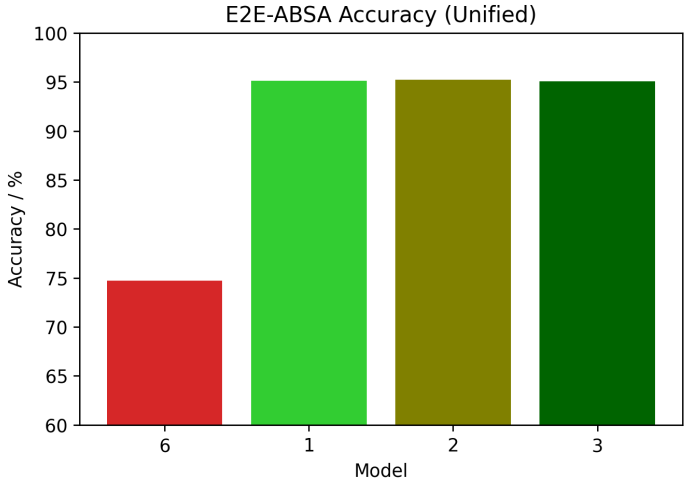


Figure 5.2: Performance Comparison with State of the Art for E2E-ABSA on Unified (Accuracy)

5.1.3 AE

Figures 5.3 and 5.4 present AE results for Lapt14 and Rest14 respectively. Albat-1LC achieves the state of the art AE F_1 score for Lapt14, while no Albat model variant achieves state of the art performance for Rest14.

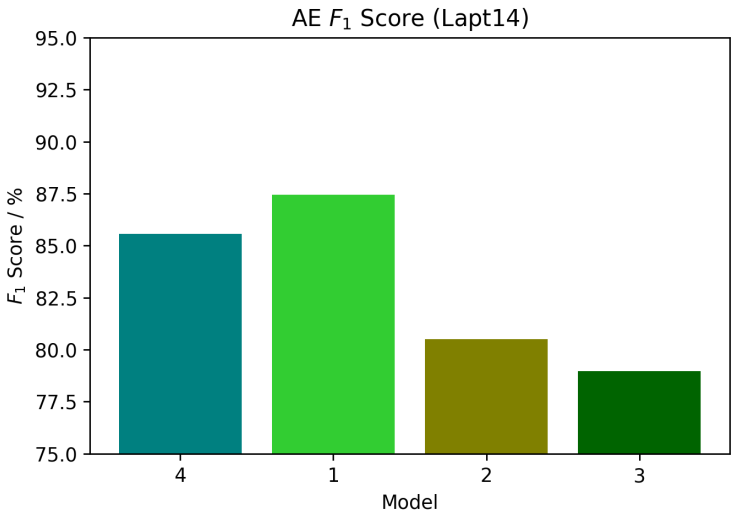


Figure 5.3: Performance Comparison with State of the Art for AE on Lapt14 (F_1 Score)

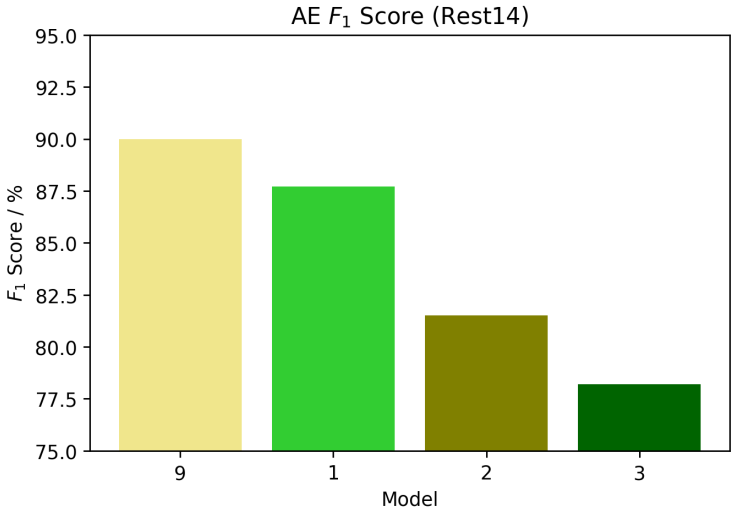


Figure 5.4: Performance Comparison with State of the Art for AE on Rest14 (F_1 Score)

Figures 5.5 and 5.6 present AE results for Camera and Car respectively. The Albat model variants achieve competitive performance on the state of the art AE F_1 score for Camera. In addition, Albat-1LC achieves the state of the art AE F_1 score for Car.

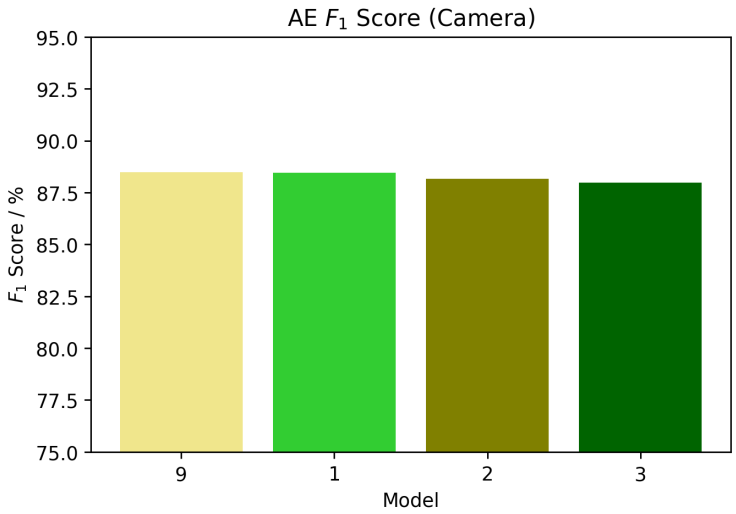


Figure 5.5: Performance Comparison with State of the Art for AE on Camera (F_1 Score)

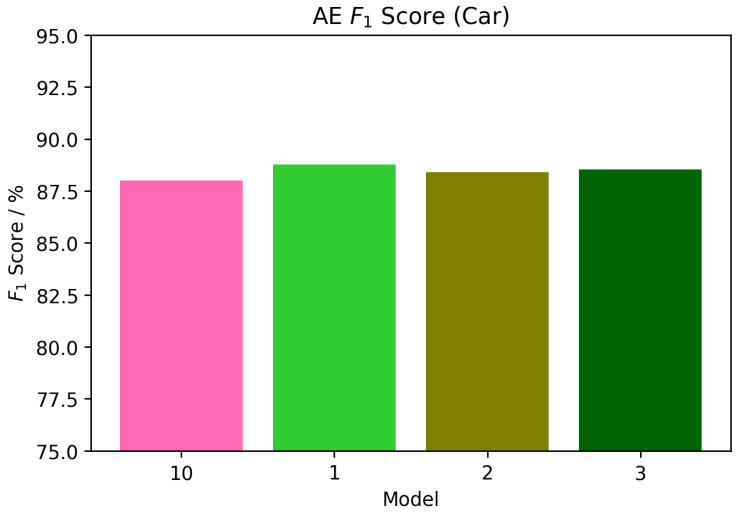


Figure 5.6: Performance Comparison with State of the Art for AE on Car (F_1 Score)

The AE results for Notebook and Phone appear respectively in Figures 5.7 and 5.8. For both domains the Albat model variants achieve competitive performance with the corresponding state of the art model.

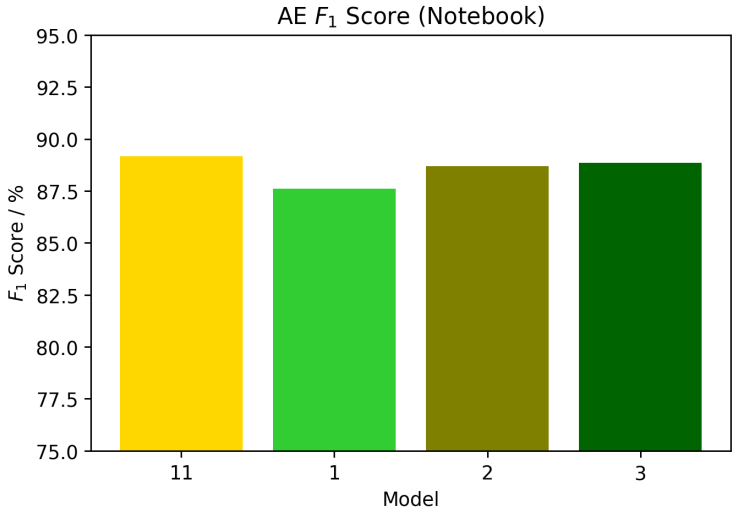


Figure 5.7: Performance Comparison with State of the Art for AE on Notebook (F_1 Score)

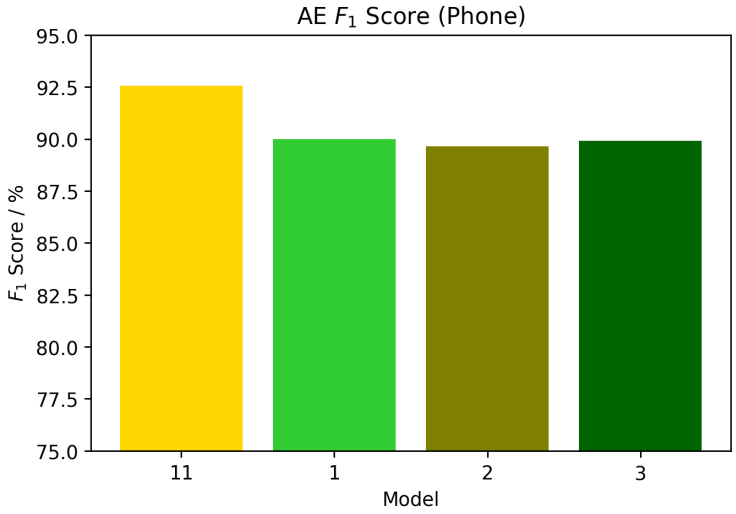


Figure 5.8: Performance Comparison with State of the Art for AE on Phone (F_1 Score)

5.1.4 ASC

Figures 5.9 and 5.10 indicate that Albat-1LC has not achieved the state of the art ASC accuracy and macro-averaged F_1 score for Lapt14.

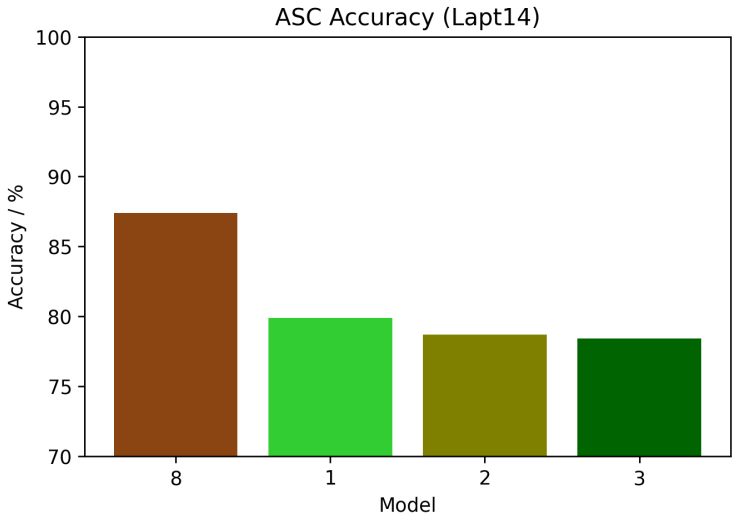


Figure 5.9: Performance Comparison with State of the Art for ASC on Lapt14 (Accuracy)

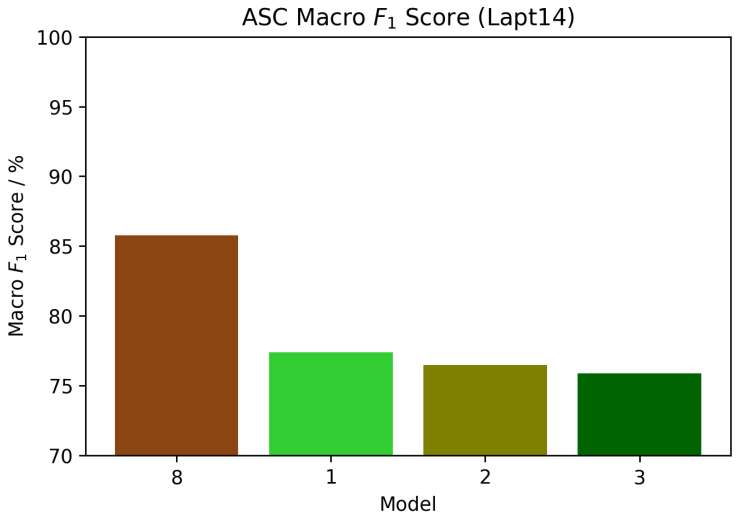


Figure 5.10: Performance Comparison with State of the Art for ASC on Lapt14 (Macro-averaged F_1 Score)

Similarly from Figures 5.11–5.12 we may observe that the Albat model variants have not achieved the state of the art ASC results for Rest14.

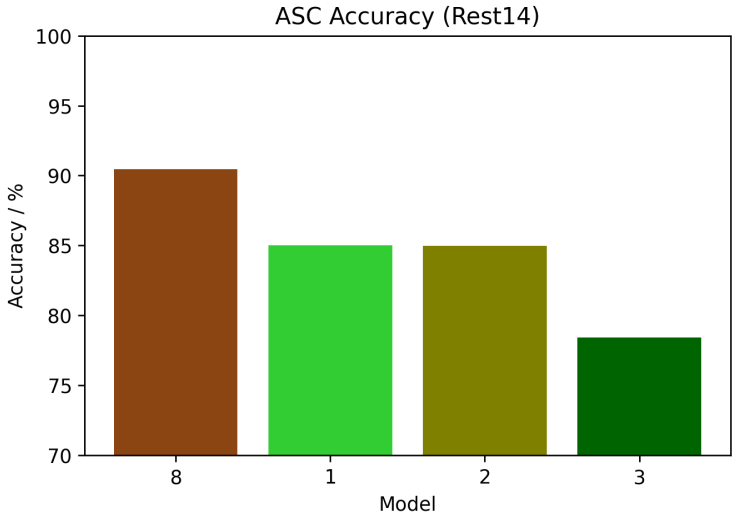


Figure 5.11: Performance Comparison with State of the Art for ASC on Rest14 (Accuracy)

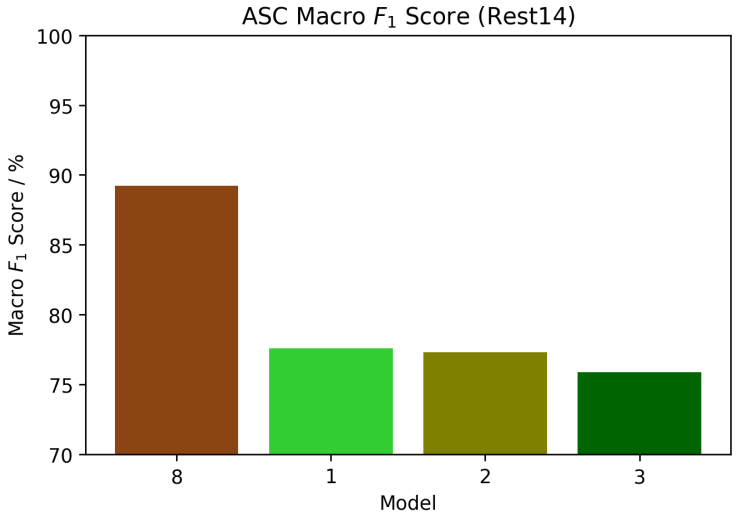


Figure 5.12: Performance Comparison with State of the Art for ASC on Rest14 (Macro-averaged F_1 Score)

Figures 5.13–5.20 demonstrate that the Albat model variants have not achieved state of the art ASC results for the four Mandarin datasets (Camera, Car, Notebook, Phone). Nevertheless, Albat model variants perform much better on the Mandarin datasets than on the English datasets when comparing accuracy and macro-averaged F_1 scores.

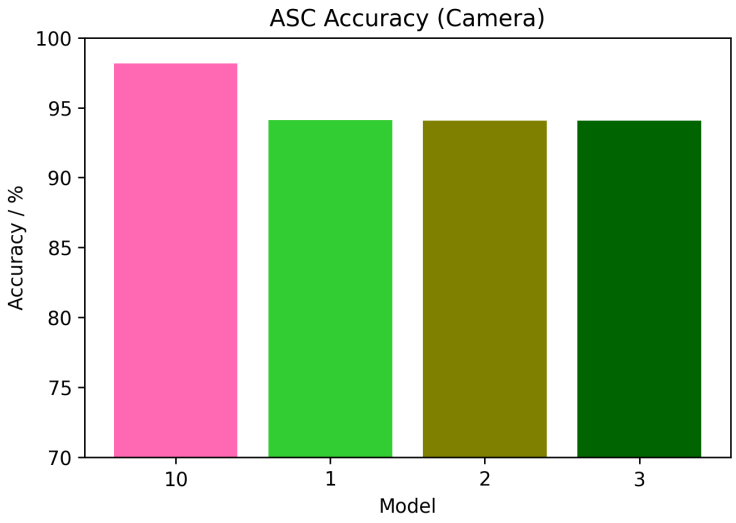


Figure 5.13: Performance Comparison with State of the Art for ASC on Camera (Accuracy)

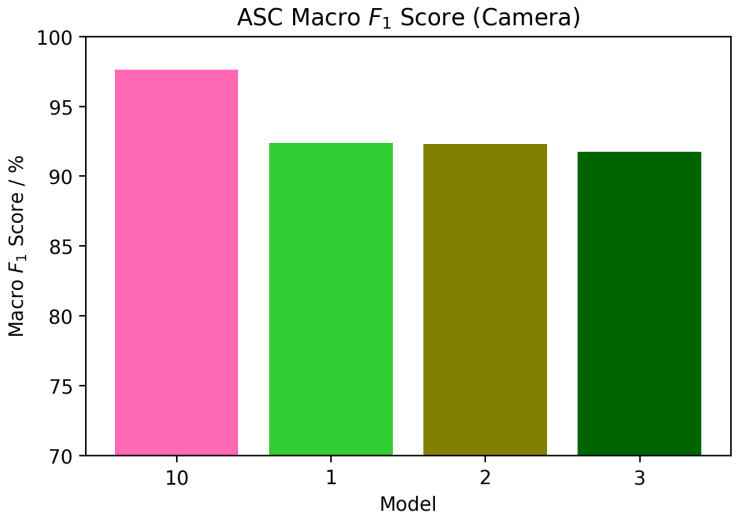


Figure 5.14: Performance Comparison with State of the Art for ASC on Camera (Macro-averaged F_1 Score)

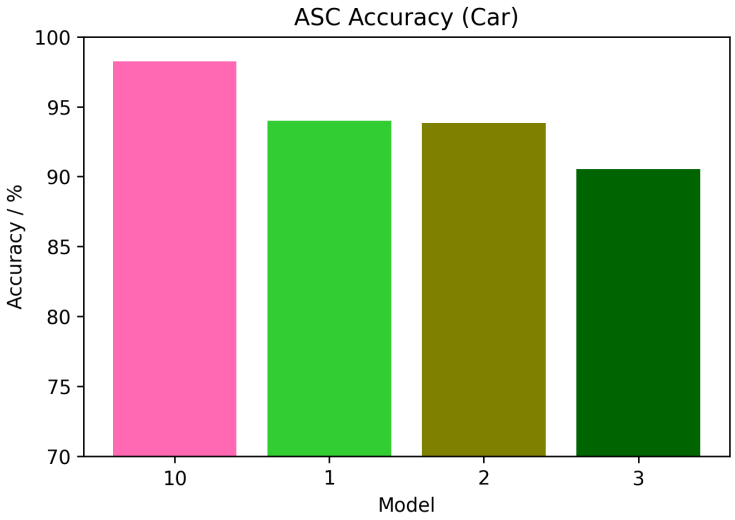


Figure 5.15: Performance Comparison with State of the Art for ASC on Car (Accuracy)

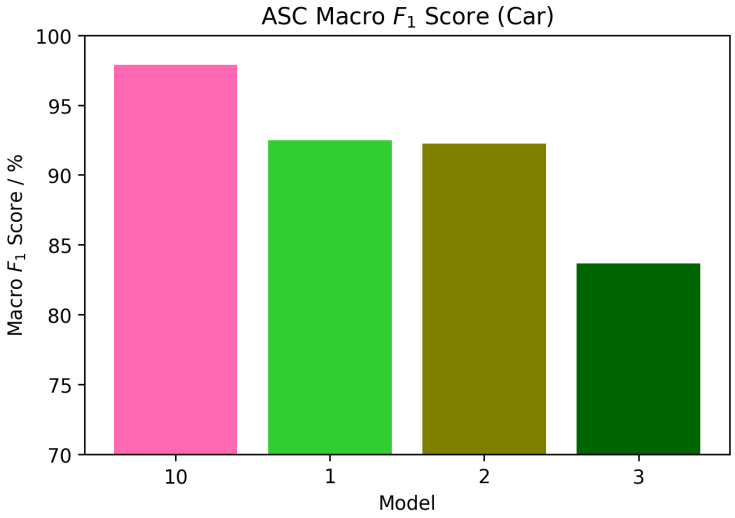


Figure 5.16: Performance Comparison with State of the Art for ASC on Car (Macro-averaged F_1 Score)

Figures 5.17–5.20 indicate that the Albat model variants' metrics for the Notebook and Phone datasets in particular are competitive with the corresponding state of the art models.

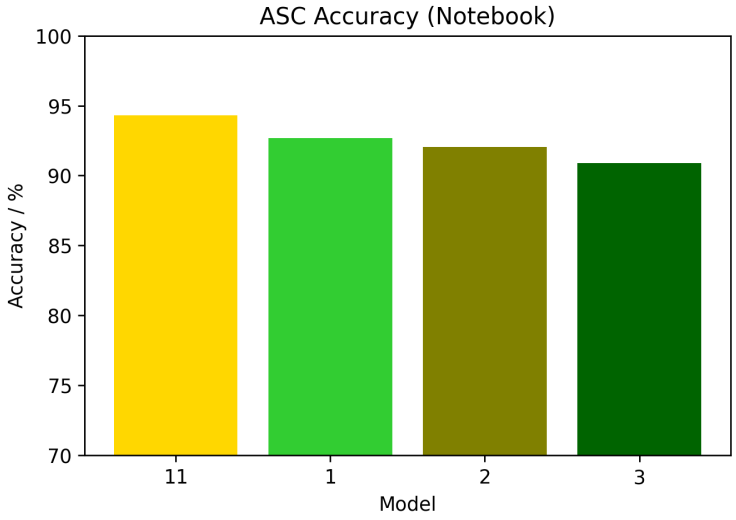


Figure 5.17: Performance Comparison with State of the Art for ASC on Notebook (Accuracy)

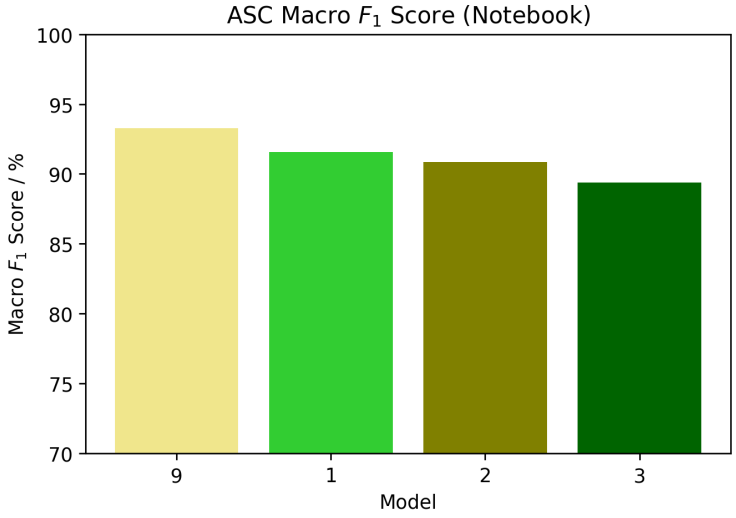


Figure 5.18: Performance Comparison with State of the Art for ASC on Notebook (Macro-averaged F_1 Score)

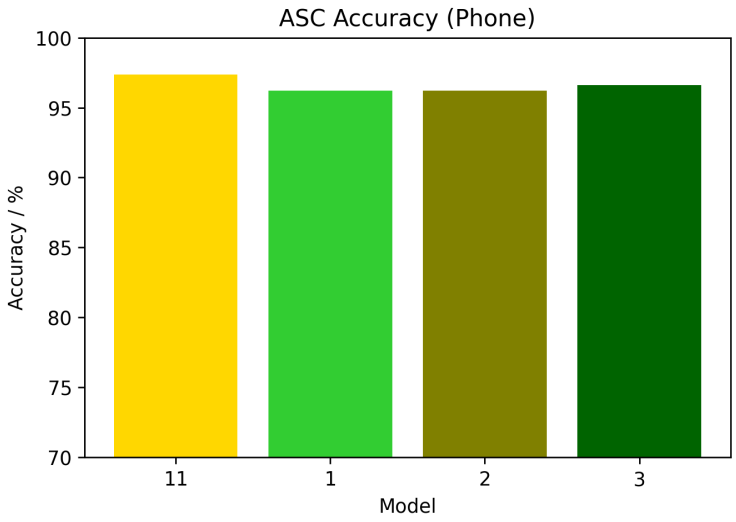


Figure 5.19: Performance Comparison with State of the Art for ASC on Phone (Accuracy)

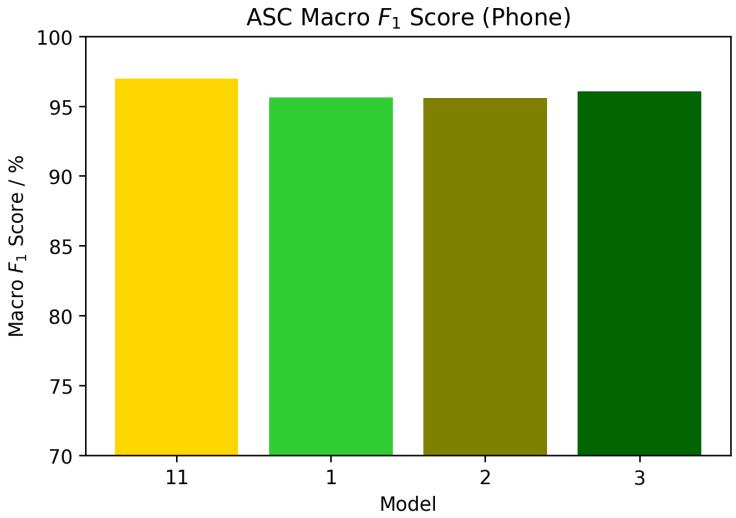


Figure 5.20: Performance Comparison with State of the Art for ASC on Phone (Macro-averaged F_1 Score)

Finally, in Figure 5.21 the optimized Albat-3LC model demonstrates the state of the art ASC accuracy for the MAMS dataset.

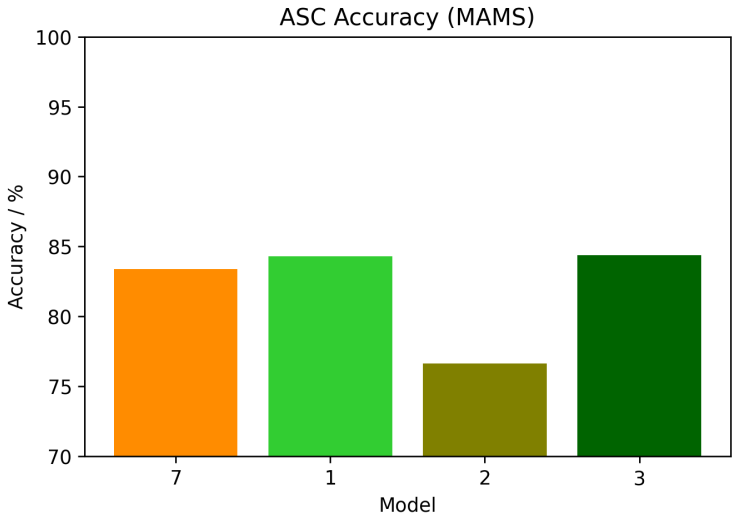


Figure 5.21: Performance Comparison with State of the Art for ASC on MAMS (Accuracy)

5.1.5 Case Study

To better understand the operation of the Albat model variants we may consider a sample English-language review and the labels predicted by each model.

Ground Truth The following text is the first review in the MAMS test subset, containing multiple aspects and sentiments. In the examples that we have provided below we have assigned a simple colour scheme in order to highlight the annotated ground truth aspect labels to the reader: green for positive aspects, silver for neutral aspects and pink for negative aspects. We note that there are three such aspects: two positive aspects (*served* and *appetizers*) and one neutral aspect (*food*).

> "The food was served promptly but the meal wasn't rushed - we had plenty of time to enjoy the appetizers and our entrees as well as sit and chat while finishing up our drinks even after we paid."

Albat-1LC The sentiment labels predicted by Albat-1LC are as follows:

> "The food was served promptly but the meal wasn't rushed - we had plenty of time to enjoy the appetizers and our entrees as well as sit and chat while finishing up our drinks even after we paid."

While *appetizers* is correctly labeled, the model tends to identify most nouns in the text as aspects. The grammatical structure of the fragment *"The food was served promptly but the ..."* would imply a contrast of sentiments. Since the first clause contains the adverb *promptly*, the subject *food* is associated with positive sentiment. For contrast the subsequent noun *meal* is associated with negative sentiment, which demonstrates that the model has not detected the use of litotes in the second clause to express positive sentiment. The author's use of the verb *enjoy* with the noun phrase *"the appetizers and our entrees"* implies favorable sentiment towards both *appetizers* and *entrees*, explaining their positive sentiment labels. The noun *drinks* is associated with neutral sentiment as the author expresses no opinion towards it. Finally, since *served* is not a noun but a past participle it is not detected as an aspect concerning the restaurant's service.

Albat-2LC The model has annotated fewer labels than both the ground truth and Albat-1LC:

> "The food was served promptly but the meal wasn't rushed - we had plenty of time to enjoy the appetizers and our entrees as well as sit and chat while finishing up our drinks even after we paid."

Neither *food* nor *served* are detected as aspects, however *appetizers* is assigned positive sentiment correctly. As with Albat-1LC, the noun *drinks* is incorrectly labeled as neutral.

Albat-3LC The model annotates the review text similarly to Albat-1LC:

> "The food was served promptly but the meal wasn't rushed - we had plenty of time to enjoy the appetizers and our entrees as well as sit and chat while finishing up our drinks even after we paid."

Unlike Albat-1LC however, the model does not predict that *entrees* is an aspect with positive sentiment.

Common Error Types Having observed the labels predicted for the review by each Albat model variant, we are able to use confusion matrices to summarize the types of errors experienced by each model. To demonstrate this visually, a square of brighter color indicates a higher number of tokens affected by a given error in the series of graphs provided below.

In Figure 5.22 the most brightly-colored square in the confusion matrix corresponds to Albat-1LC predicting positive sentiment for a non-aspect word (*entrees* is counted as two tokens by `albert-base-v2`, hence the corresponding error event frequency is higher).

The teal-colored squares correspond to an instance of:

- predicting negative sentiment for a non-aspect word (*meal*);
- predicting neutral sentiment for a non-aspect word (*drinks*);
- predicting no sentiment for an aspect with positive sentiment (*served*);
- predicting positive sentiment for an aspect with neutral sentiment (*food*).

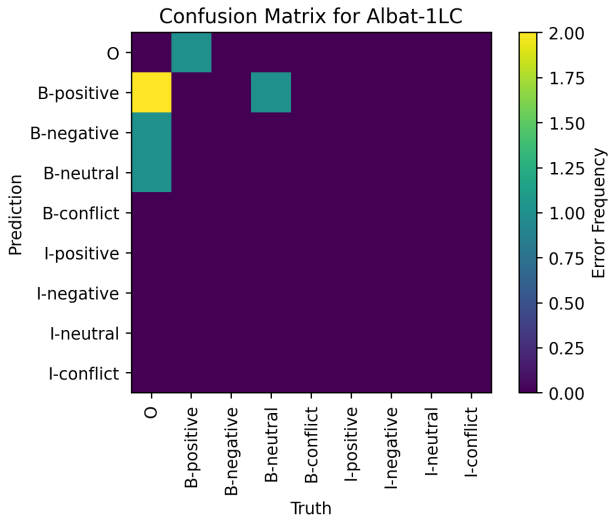


Figure 5.22: Confusion Matrix for Sample Review (Albat-1LC)

With Albat-2LC in Figure 5.23 we see fewer types of mistakes with equal frequencies: predicting no sentiment for aspects with positive (*served*) and neutral sentiments (*food*), and predicting neutral sentiment for a non-aspect word (*drinks*).

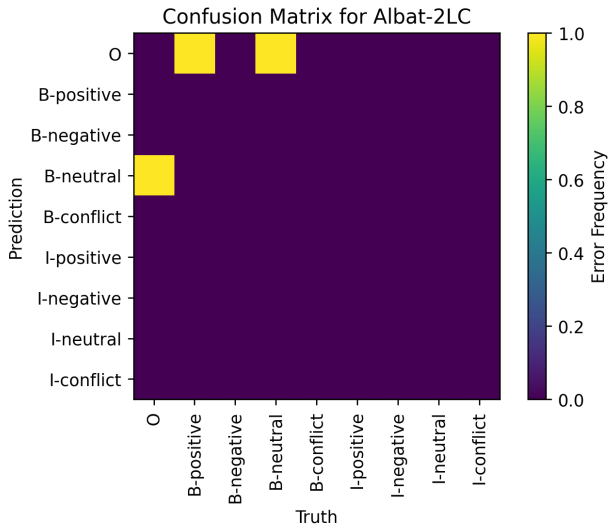


Figure 5.23: Confusion Matrix for Sample Review (Albat-2LC)

With Albat-3LC, in Figure 5.24 we see similar mistakes to those of Albat-1LC without predicting positive sentiment for a non-aspect word (*entrees*).

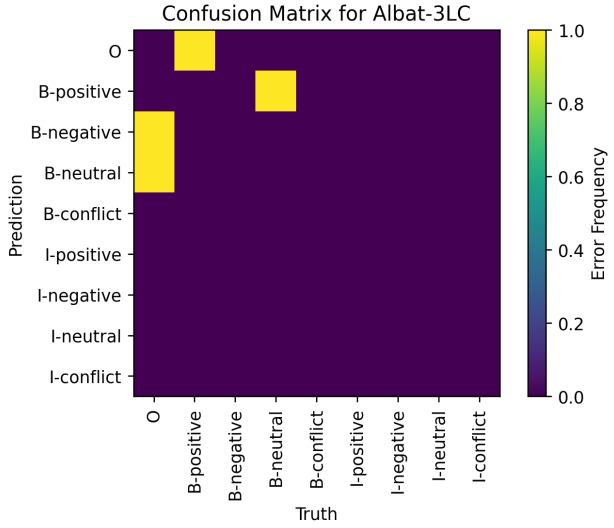


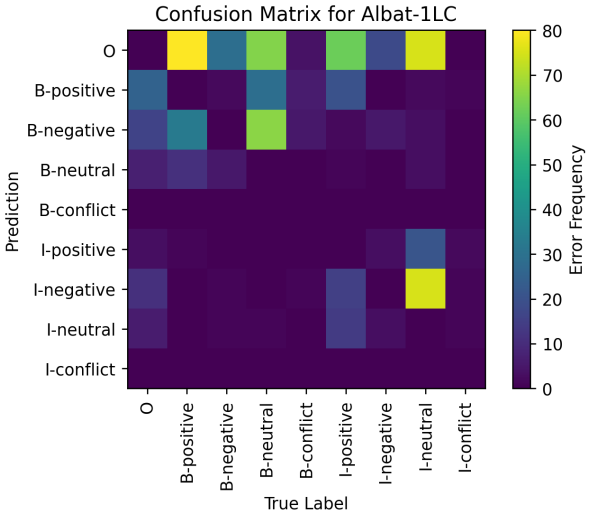
Figure 5.24: Confusion Matrix for Sample Review (Albat-3LC)

From this example, we can hypothesize that Albat model variants tend to predict sentiments for non-aspect words or vice versa, and that it is less common to confuse an aspect word's sentiment for another sentiment. Using these observations, we may infer that the ground truth labels have not considered words which may be legitimate aspect phrases.

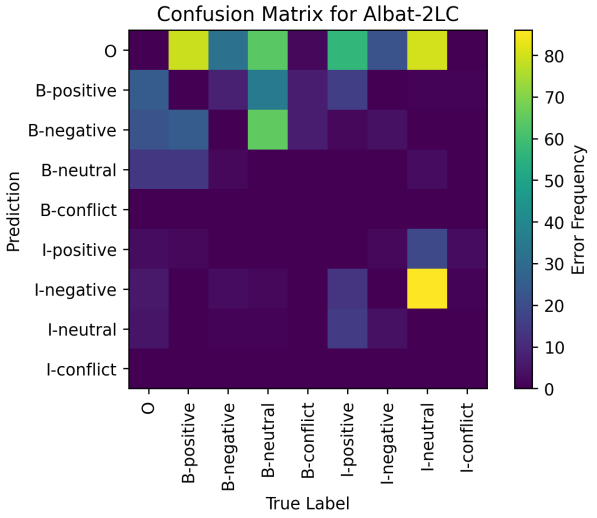
5.1.6 Confusion Matrices

We next consider the types of errors for each dataset and model for the task of E2E-ABSA.

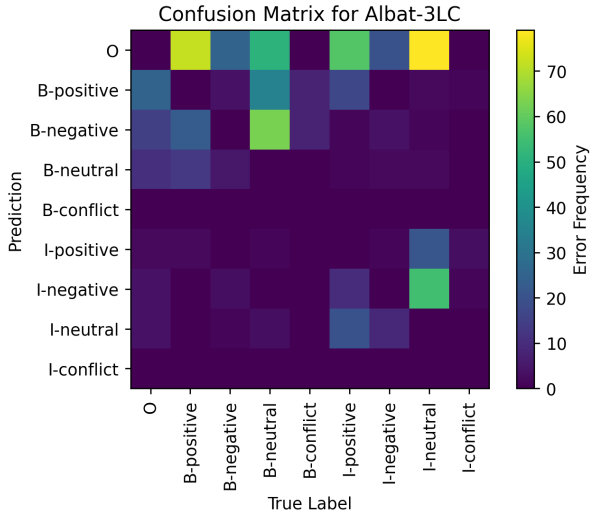
Lapt14 Figures 5.25a–5.25c display similar patterns to each other. The most frequent error is predicting no sentiment for a true aspect word (portrayed by the top row of the confusion matrix), followed by predicting negative sentiment instead of neutral sentiment. Confusion between sentiments is less common, particularly involving the conflict sentiment.



(a) Albat-1LC



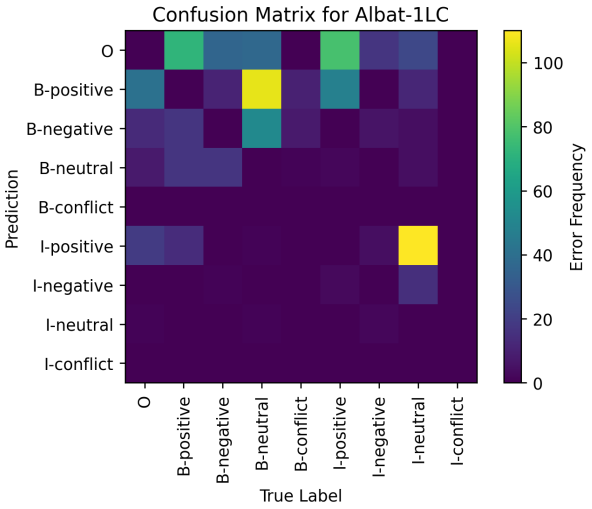
(b) Albat-2LC



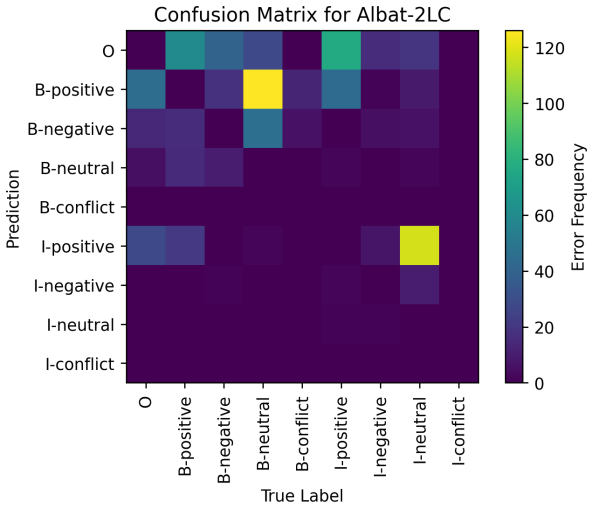
(c) Albat-3LC

Figure 5.25: Confusion Matrices for Lapt14

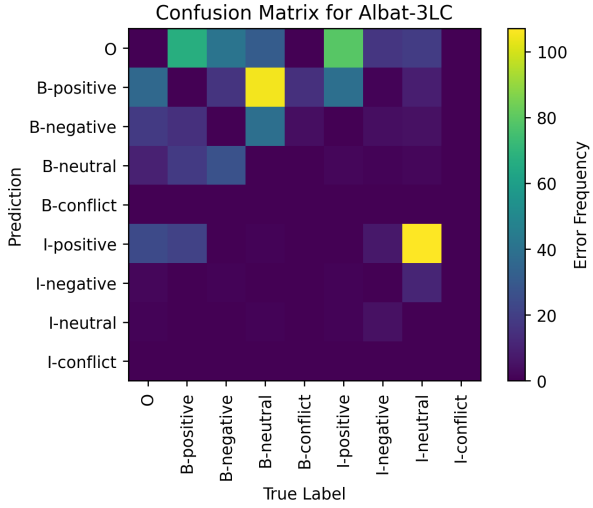
Rest14 Figures 5.26a–5.26c display similar patterns to each other. The most frequent errors involve predicting positive sentiment instead of neutral sentiment, followed by predicting no sentiment for a true aspect word. As for Lapt14, confusion between sentiments is less common, as is predicting a sentiment for a non-aspect word.



(a) Albat-1LC



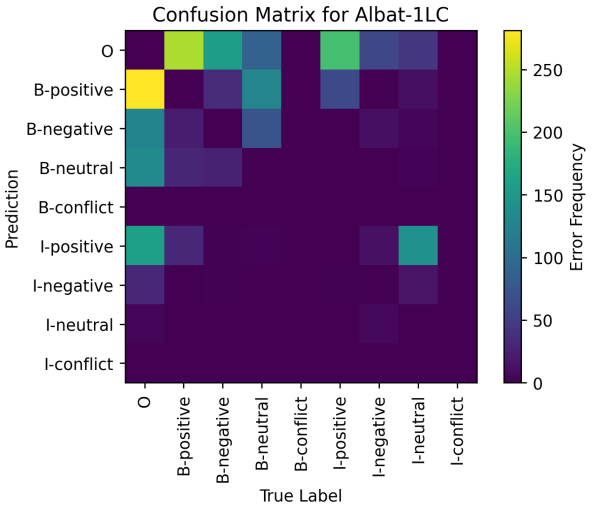
(b) Albat-2LC



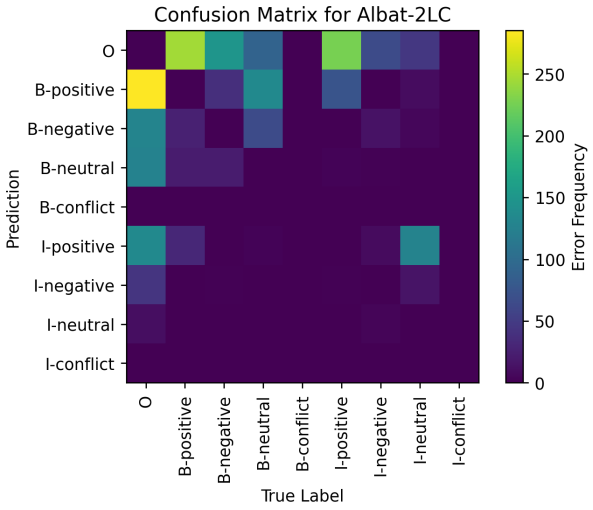
(c) Albat-3LC

Figure 5.26: Confusion Matrices for Rest14

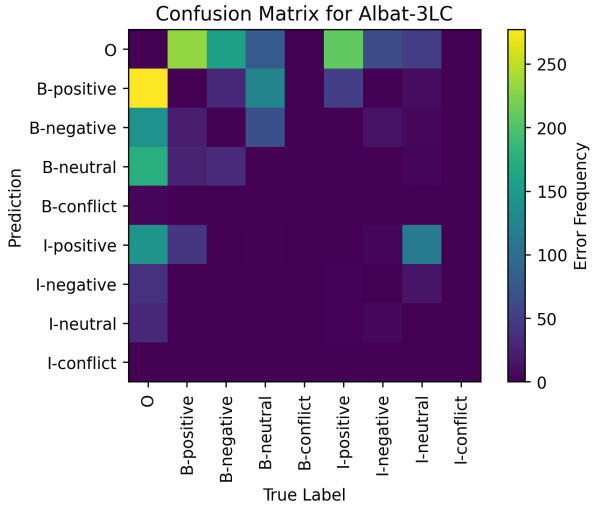
Unified Figures 5.27a–5.27c display similar patterns to each other. Unlike Lapt14 and Rest14, the most frequent errors involve predicting a sentiment for non-aspect words (chiefly positive sentiment), followed by predicting no sentiment for true aspect words (primarily with positive sentiments), and predicting positive sentiment instead of neutral sentiment. Other types of confusions between sentiments are less common. Note that Unified and MAMS do not contain conflict-labeled aspects.



(a) Albat-1LC



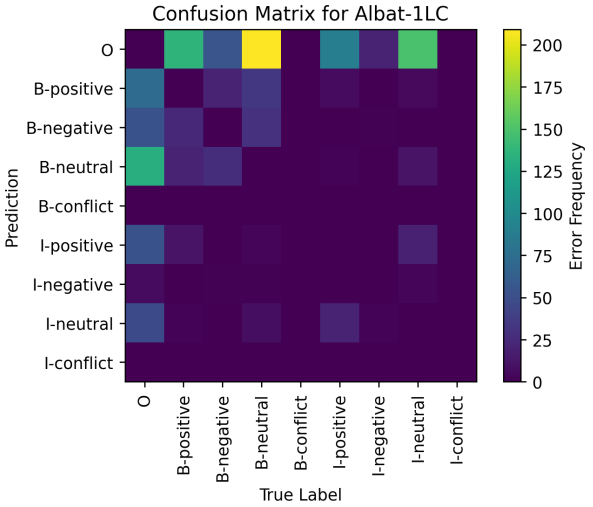
(b) Albat-2LC



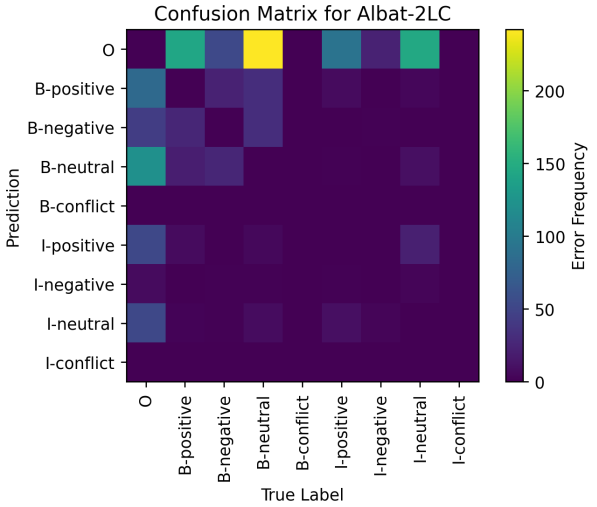
(c) Albat-3LC

Figure 5.27: Confusion Matrices for Unified

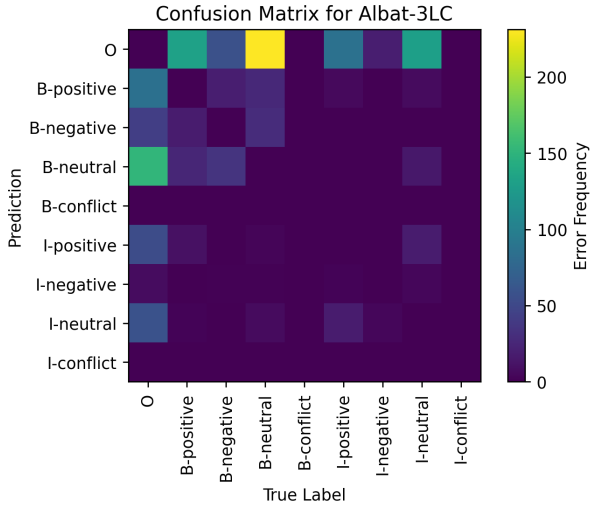
MAMS Figures 5.28a–5.28c display similar patterns to each other. It is most common to predict no sentiment for true aspect words (particularly those with positive sentiment), followed by predicting neutral sentiment for non-aspect words. It is less common to predict other sentiments for non-aspect words, or to confuse positive, neutral, and negative sentiments with each other.



(a) Albat-1LC



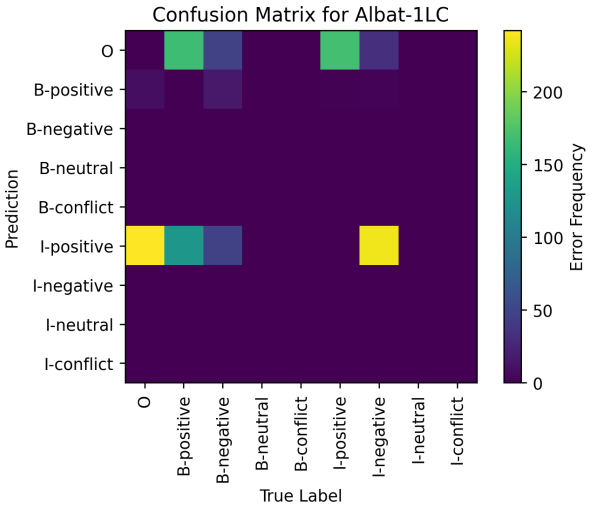
(b) Albat-2LC



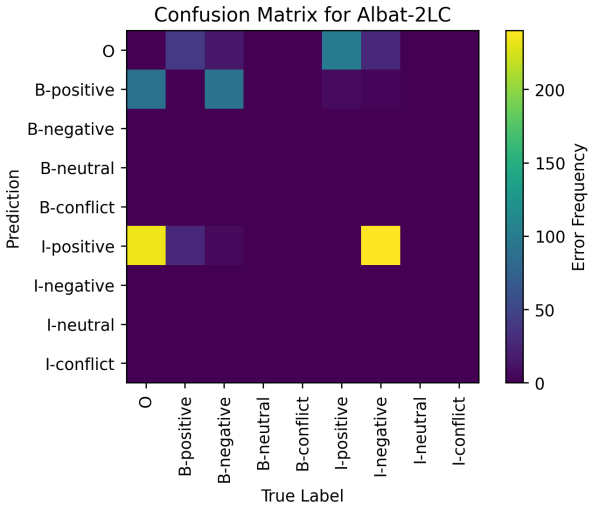
(c) Albat-3LC

Figure 5.28: Confusion Matrices for MAMS

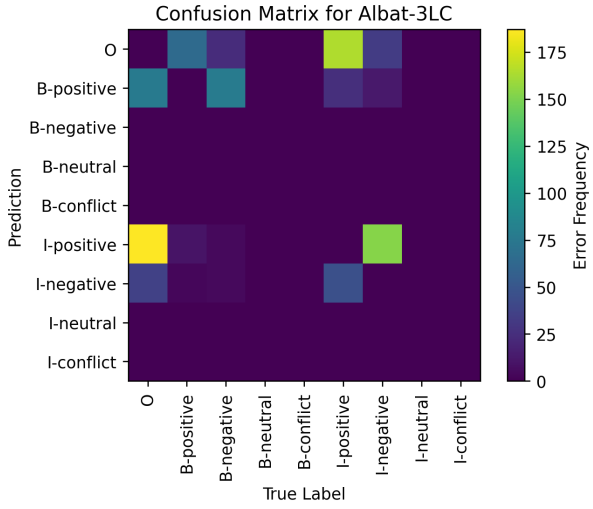
Camera Figures 5.29a–5.29c display similar error distributions to each other. All three Albat models frequently assign positive sentiment to negative aspects and non-aspect words, while assigning no sentiment to positive aspects is less common. This suggests a tendency towards over-assigning positive sentiments, perhaps due to the imbalance of positive and negative aspects in the dataset. Note that Camera and the other Mandarin datasets do not contain neutral- or conflict-labeled aspects.



(a) Albat-1LC



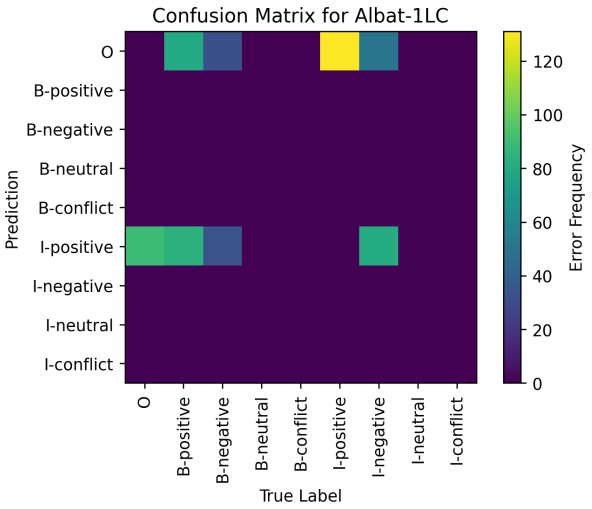
(b) Albat-2LC



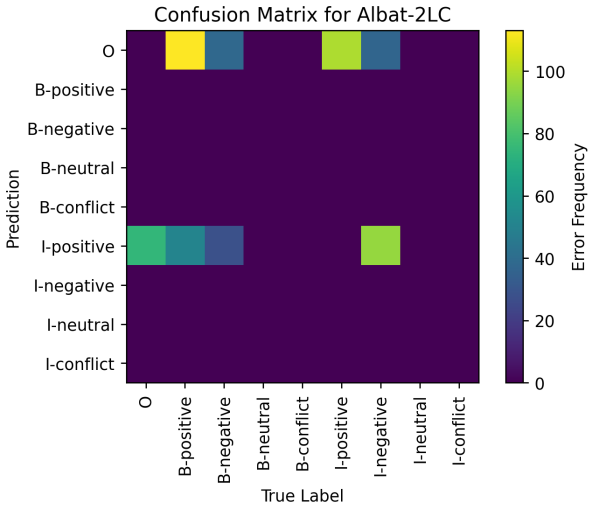
(c) Albat-3LC

Figure 5.29: Confusion Matrices for Camera

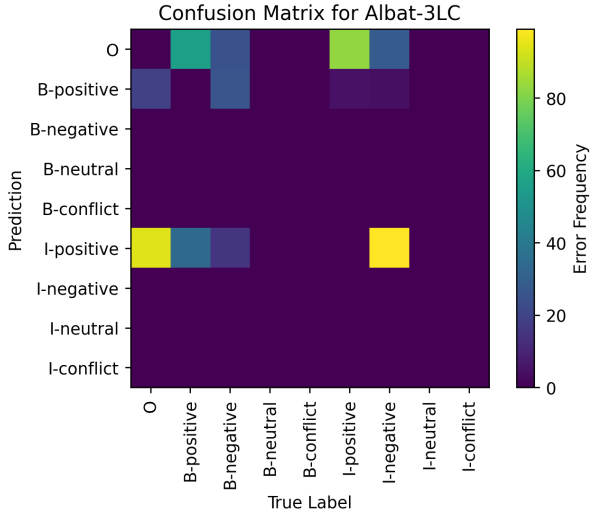
Car In Figures 5.30a and 5.30b, the most common confusions involve assigning no sentiment to positive aspects. It is less common for these models to assign positive sentiment to non-aspect words and negative aspects, however these are the most frequent error types for Albat-3LC in Figure 5.30c. As seen with Camera, these results suggest an imbalance of training examples containing positive and negative aspects.



(a) Albat-1LC



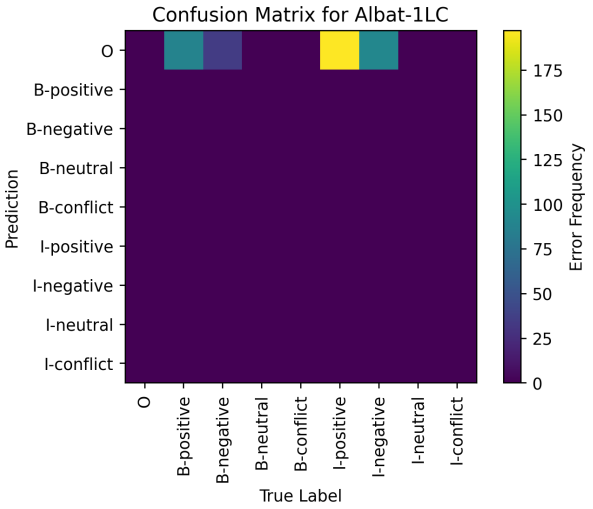
(b) Albat-2LC



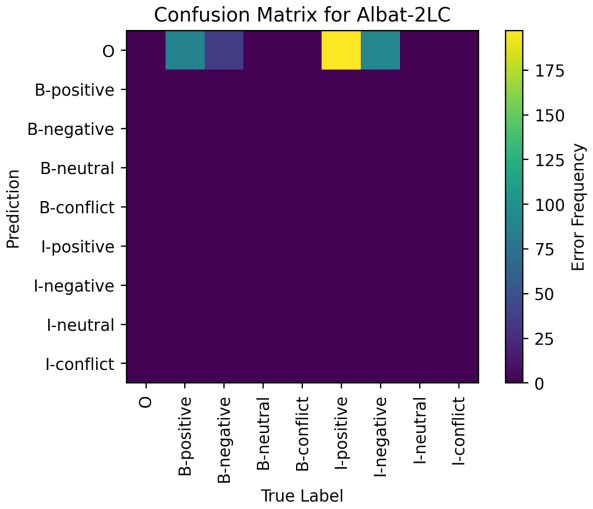
(c) Albat-3LC

Figure 5.30: Confusion Matrices for Car

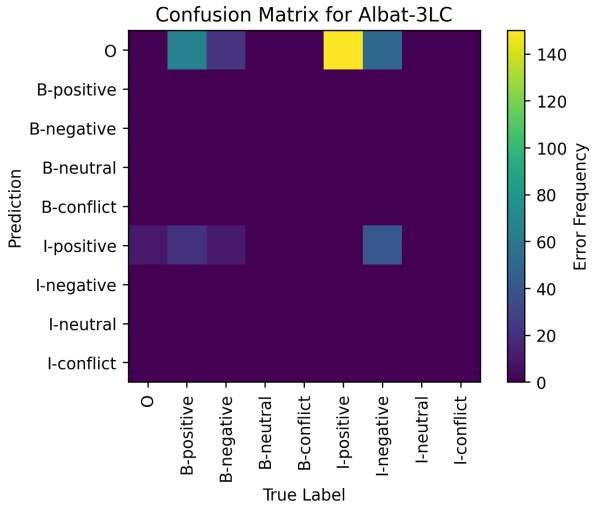
Notebook In Figures 5.31a and 5.31b the only errors arise when the model assigns no sentiment to positive and negative aspects. In addition, we may observe an additional group of errors in Figure 5.30c, with Albat-3LC assigning positive sentiment to negative aspects and non-aspect words less frequently.



(a) Albat-1LC



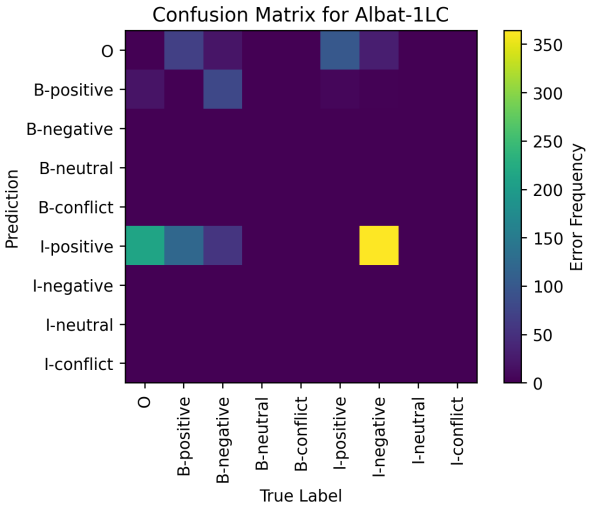
(b) Albat-2LC



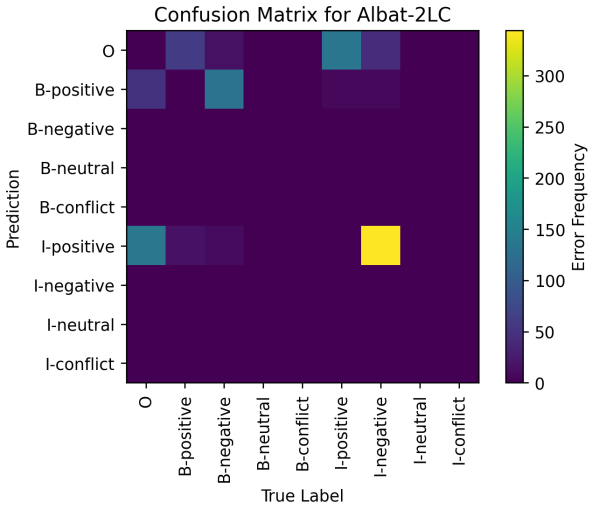
(c) Albat-3LC

Figure 5.31: Confusion Matrices for Notebook

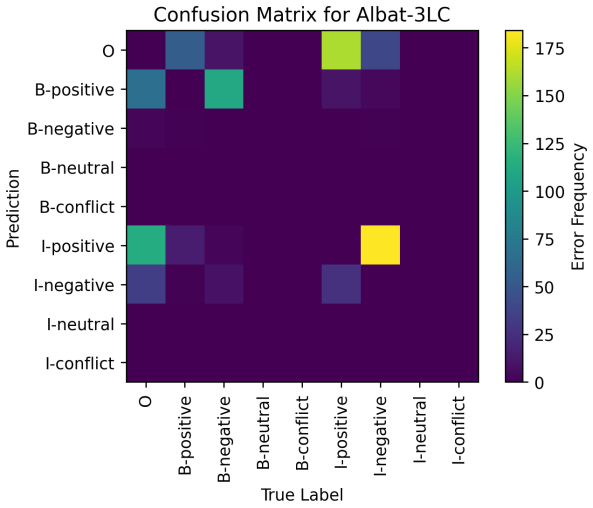
Phone Figures 5.32a and 5.32b display similar patterns to each other, with the most frequent errors occurring when the model assigns positive sentiment to negative aspects and non-aspect words. In Figure 5.32c there is a higher frequency of errors assigning no sentiment to positive aspects, followed by assigning negative sentiment to positive aspects and non-aspect words. As seen with the other Mandarin datasets, the majority of aspects examples in Phone have positive sentiment, and this is reflected by the models' tendency to over-assign positive sentiment.



(a) Albat-1LC



(b) Albat-2LC



(c) Albat-3LC

Figure 5.32: Confusion Matrices for Phone

5.1.7 Further Pre-Training

We now turn to evaluating the methodology for pre-training Albert, considering the performance observed with pre-training `albert-base-v2` on 1% and 5% of the review corpus in Section 4.5.

Methodology To implement further pre-training we have used a script contained within the code repository for [63]. The original paper proposed numerous methods for further pre-training BERT models:

- **BERT-DK**, using unsupervised in-domain data;
- **BERT-MRC**, using supervised data for reading comprehension tasks;
- **BERT-PT**, interleaving training examples from in-domain review data and out-of-domain reading comprehension data.

While the paper's results have demonstrated superior performance on ABSA tasks when using the BERT-PT method, the code repository contains an implementation of the BERT-DK method. It is therefore plausible that the poor performance observed with the further pre-trained Albert models has arisen due to two connected factors:

- First, using the BERT-DK method may have caused catastrophic forgetting of task-related knowledge contained within `albert-base-v2`; and
- Secondly, using smaller subsets of the review corpus may have exacerbated this effect.

Data Quantity and BERT-PT As observed in Section 4.5, using a larger review corpus for further pre-training may not necessarily improve the performance of Albat model variants on ABSA tasks. We speculate that using the BERT-PT method instead of the BERT-DK method may yield superior performance when pre-training Albert models further.

5.2 Resource Usage

The primary motivation behind the development of Albert was to reduce the number of trainable parameters used in BERT [5]. We also anticipated a decrease in the memory usage and in the time required to train Albert on a given dataset with respect to BERT.

To verify whether such gains from resource conservation have been achieved with Albat, we have collected run-time statistics from BAT and the three optimized Albat model variants. We have fine-tuned each model (BAT, Albat-1LC, Albat-2LC, Albat-3LC) for three epochs with different task-dataset combinations:

- AE and ASC using Lapt14 and Rest14 (BAT and Albat models);
- AE and ASC using the remaining six datasets (Albat models);
- E2E-ABSA using all datasets (Albat models).

For each fine-tuned model, we have recorded the number of trainable parameters, the size of the model in memory, and the duration of training. We have profiled the resource usage of BAT fine-tuned with the datasets used in [64] (Lapt14 and Rest14), while we have provided full results across all three tasks and eight datasets for the Albat model variants. Of particular importance is the comparison between BAT and Albat-1LC, which differ only in the use of BERT or Albert.

We must recall that the underlying Albert encoders change depending on whether English text is used (i.e. `albert-base-v2`) or Mandarin text is used (i.e. `albert_chinese_base`). We have therefore collated duplicate results for datasets using the same underlying model when considering the number of trainable parameters and model size: **English** datasets refer to Lapt14, Rest14, MAMS, and Unified, while **Mandarin** datasets refer to Camera, Car, Notebook, and Phone.

Trainable Parameters In Figure 5.33 we have observed an 87.6% reduction in the number of parameters between BAT and Albat-1LC for the English AE and ASC tasks (1.09×10^8 versus 1.35×10^7). While Albat-2LC and Albat-3LC have slightly more trainable parameters for the same task-dataset combination (1.69×10^7 and 1.70×10^7 respectively), these quantities are still on an order of magnitude less than that of BAT.

For all task-dataset combinations Albat-1LC has fewer parameters than Albat-2LC and Albat-3LC. Furthermore, the three Albat model variants using `albert_chinese_base` have fewer trainable parameters (1.06×10^7, 1.20×10^7, and 1.59×10^7) than the equivalent English model variants (1.35×10^7, 1.69×10^7, and 1.70×10^7). In addition the proportional increase in trainable parameters between Albat-2LC and Albat-3LC is greater when using `albert_chinese_base` (+32.5%) than when using `albert-base-v2` (+6.25%).

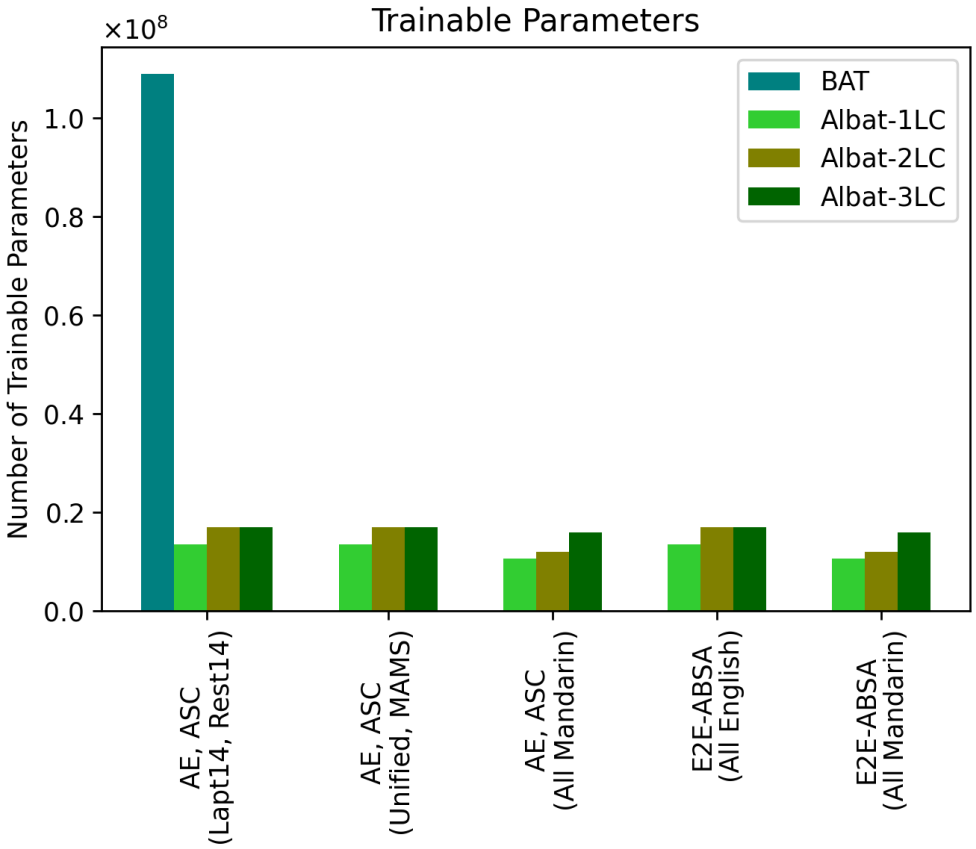


Figure 5.33: Trainable Parameters for BAT and Albat

Model Size We have observed a similar trend in the memory size of models. In Figure 5.34 the BAT models fine-tuned on the Lapt14 and Rest14 AE and ASC tasks occupy 417.7 MB of memory, while the corresponding Albat-1LC models occupy 51.4 MB of memory (a reduction of 87.7%). For the same task-dataset combinations Albat-2LC and Albat-3LC occupy more memory (64.3 MB and 64.9 MB) than Albat-1LC, a trend which is repeated across all AE, ASC, and E2E-ABSA tasks.

As with the number of trainable parameters, the Mandarin Albat model variants require less memory (40.3 MB, 46.0 MB, and 60.6 MB) than their English counterparts (51.4 MB, 64.3 MB, and 64.9 MB). Similarly, the proportional increase in model size between Albat-2LC and Albat-3LC is greater for `albert_chinese_base` (+31.7%) than for `albert-base-v2` (+0.9%).

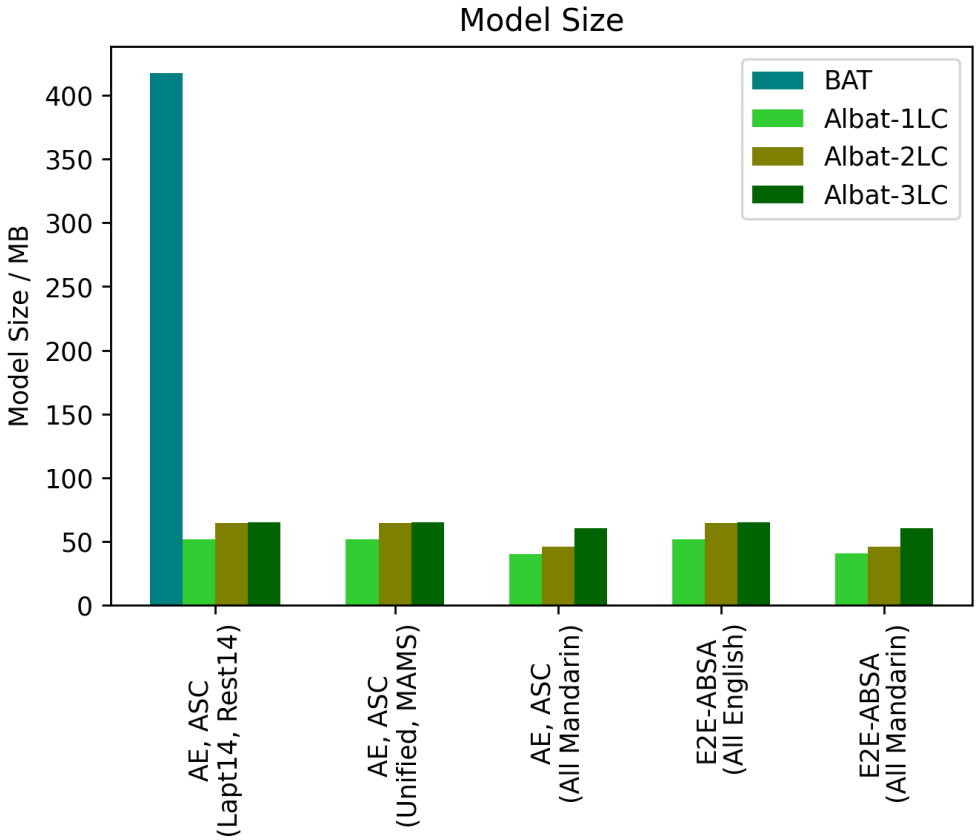


Figure 5.34: Size of Trained Models for BAT and Albat

Training Duration For a finer understanding of the time required to fine-tune models with each of the eight datasets, we have considered the training duration for the three ABSA tasks individually.

Figure 5.35 demonstrates a reduction in training duration for AE with Lapt14 and Rest14 when using Albat-1LC instead of BAT, by 62 seconds and 44 seconds respectively. For all datasets except for MAMS the training duration increases between Albat-1LC and Albat-2LC, while the training duration increases between Albat-2LC and Albat-3LC for all datasets.

It is worth comparing the training durations between datasets: Lapt14 and MAMS require longer training, followed by Unified, Rest14, Phone, Camera, Car and Notebook. Due to the greater number of sentences in the Lapt14 and MAMS datasets, each epoch of fine-tuning considers a larger number of examples, so training requires more time. The observation that dataset size correlates with training duration also explains the shorter training durations for Notebook and Car.

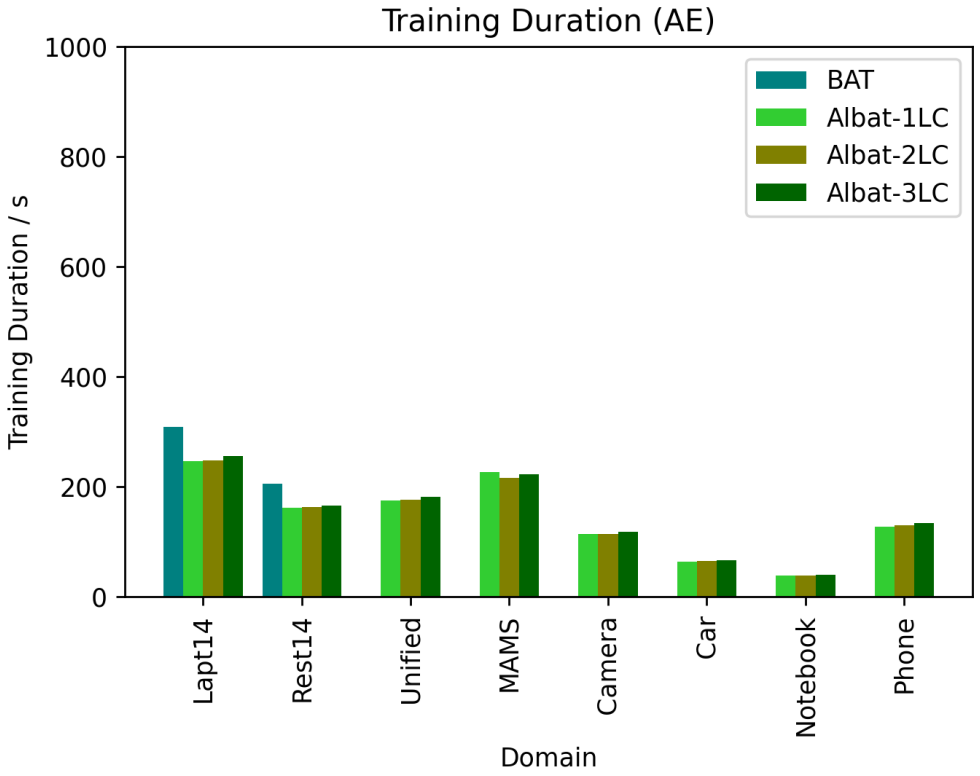


Figure 5.35: Training Duration for BAT and Albat Models (AE)

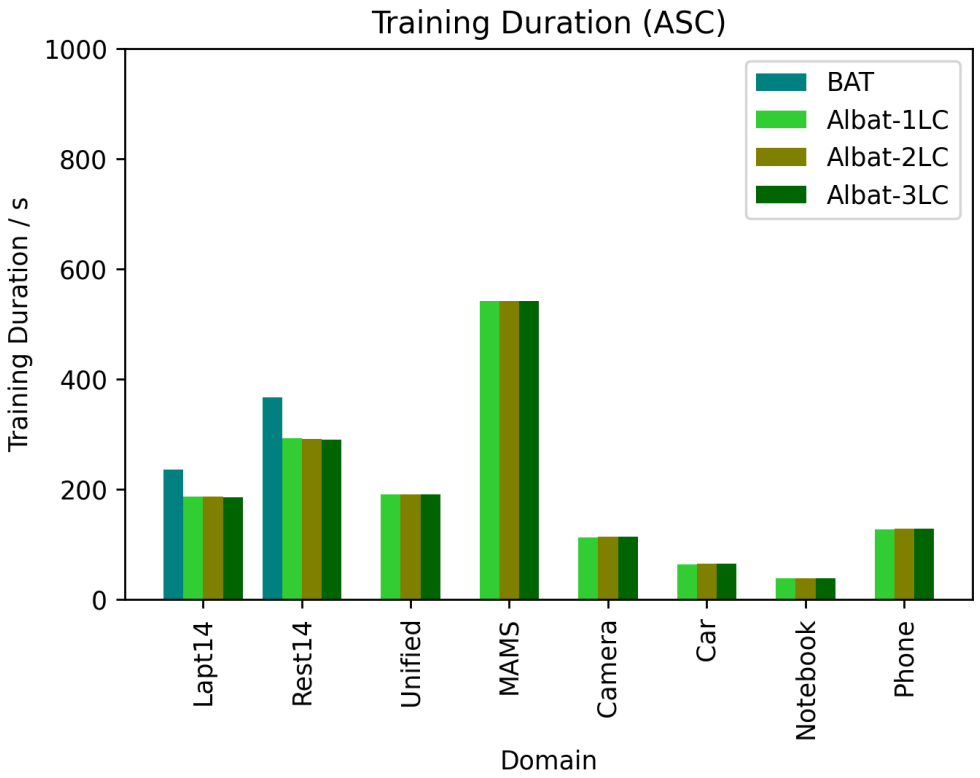


Figure 5.36: Training Duration for BAT and Albat Models (ASC)

As with AE, Figure 5.36 demonstrates a reduction in training duration for ASC with Lapt14 and Rest14 when using Albat-1LC instead of BAT, by 48 seconds and 73 seconds respectively. For the English datasets the training duration decreases between Albat-1LC and Albat-2LC, while for the Mandarin datasets the training duration increases. The training duration increases between Albat-2LC and Albat-3LC for all datasets except Lapt14 and Rest14. MAMS in particular has a much longer training duration than the other datasets, followed by Rest14, Unified, Lapt14, Phone, Camera, Car, and Notebook.

In Figure 5.37 we have observed that when fine-tuning on E2E-ABSA with all datasets except Car, the training duration increases between Albat-1LC and Albat-2LC. In addition the training duration increases between Albat-2LC and Albat-3LC for all datasets. Unified and MAMS require longer training, followed by Lapt14, Rest14, Phone, Camera, Car and Notebook.

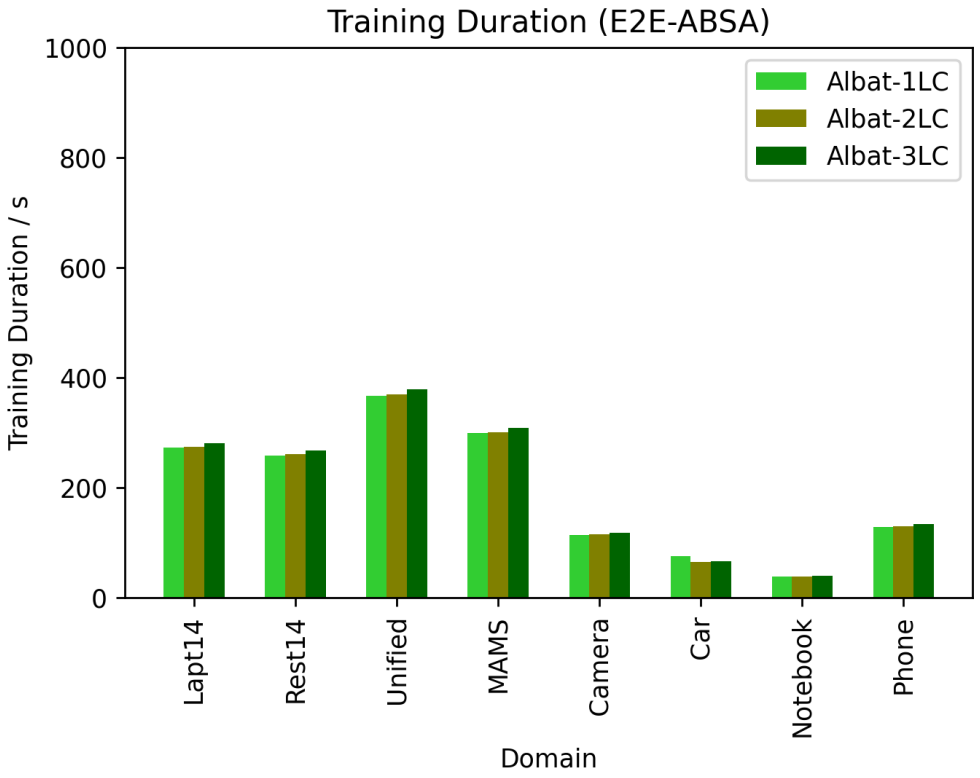


Figure 5.37: Training Duration for BAT and Albat Models (E2E-ABSA)

Chapter 6

Conclusions

6.1 Summary

Our project has focused on applying the Albert language model to the problem of ABSA. Our primary goal has been to integrate an adversarial training method with Albert in order to perform E2E-ABSA and other ABSA tasks.

Literature Review At the beginning of the project we surveyed literature relating to ABSA, considering both information retrieval systems and machine learning techniques. Among the deep neural network architectures discussed, we examined the use of the BERT language model in ABSA applications. We then collated and summarized information about various datasets commonly used to train and evaluate performance on ABSA tasks.

Design After conducting the review of ABSA literature, we formalized the three ABSA tasks (AE, ASC, and E2E-ABSA) as learning problems. Inspired by the BERT Adversarial Training model proposed in [64], we incorporated the lighter Albert language model into an adversarial training scheme, thus creating the Albat model architecture. We also briefly discussed the rationale behind further pre-training Albert on a generic review corpus.

Results We presented details of our implementation of the Albat model architecture before performing a comprehensive optimization of hyperparameter settings using the Lapt14 and Rest14 datasets. Using the optimized hyperparameters we evaluated the performance of three Albat variants on the three ABSA tasks. We also performed further pre-training with Albert on subsets of a review corpus comprising online e-commerce and restaurant reviews, noting a deterioration in the resulting Albat models' performance on ABSA tasks.

Discussion Regarding the experimental outcomes, we compared and discussed our results with those found in the literature. Focusing on a single review, we contrasted the differences between the three Albat models' label predictions and speculated on the underlying reasons. We generated confusion matrices for each dataset and Albat model variant, extracting patterns in error type and frequency. Finally we analyzed each Albat model's resource usage.

We conclude with a reflection on the contributions of this investigation to the literature and discuss a number of project extensions for future research.

6.2 Contributions

By means of this investigation we consider that we have advanced the literature in four ways.

First, we have adapted an adversarial training framework for ABSA tasks, substituting the default Albert model for BERT models further pre-trained on domain-specific review corpora. This decision was motivated by a desire for a reduced model memory footprint and training duration, and has been justified by analysis in Section 5.2.

We have evaluated the efficacy of the adversarial training in addressing E2E-ABSA. While adversarial training was successful with AE and ASC in [64], its use with E2E-ABSA had not been attempted previously. Our use of adversarial training with E2E-ABSA has yielded successful results.

To evaluate model performance on E2E-ABSA we have adapted existing datasets to use a consistent and unified tagging scheme. This has enabled the generation of performance benchmarks for E2E-ABSA with eight datasets (Lapt14, Rest14, Unified, MAMS, Camera, Car, Notebook, and Phone) in two languages (English and Mandarin), widening the scope for future research into E2E-ABSA systems' performance.

Finally, using optimized Albat model variants we have achieved state of the art performance on all E2E-ABSA metrics, two AE metrics, and one ASC metric. The optimized Albat model variants demonstrate competitive performance regarding the state of the art for four AE metrics and four ASC metrics. This provides a foundation for Albat model performance and justifies further improvements to the Albat model architecture.

We refer readers interested in inspecting our code to the Albat online repository.

6.3 Future Work

Given Albat's performance, we outline a number of promising directions for future research.

Building on Section 4, additional tests of Albat model's performance may be pursued. While we have only considered English-language reviews from the laptop computer and restaurant domains and Mandarin-language reviews from the notebook computer, camera, car, and mobile telephone domains, several datasets for ABSA have been released for other languages and domains (notably SentiHood [87], SemEval-2016.5 [85], and the 2014 Twitter dataset [86]). It would be valuable to observe how well Albat models perform on ABSA tasks after fine-tuning using these additional datasets. We also anticipate that the research community will release pre-trained Albert models in more languages, allowing more datasets and domains to be investigated.

In parallel with exploring additional datasets, the operation of the Albat model could be probed further. Analysis of the parameter weights within the fine-tuned Albat models' layers and attention head modules could be conducted as per [96]. Doing this would clarify the relative importance of these components for the label prediction process and would permit a more efficient restructuring of the Albat model (for example, tuning each classification layer's learning rates individually). Employing alternative Albert models such as `albert-large-v2` could yield superior performance on ABSA tasks (as observed with other NLP tasks [5]).

Architectural changes to the Albat structure may improve the handling of subtle, more complex sentence structures. The inclusion of an auxiliary sentence as an input appears to improve performance in BERT-based models for ABSA [9, 61, 65], so it is conceivable that a similar approach could be incorporated into Albat. Although the use of embeddings that include additional information (such aspect target phrase tokens) has also boosted the performance of the BERT-based model on ASC in [65, 73], it is unclear whether it would be feasible or desirable to apply them to other ABSA tasks. This remains to be examined.

The procedures for pre-training and fine-tuning Albat may be adapted as new strategies for BERT-based models emerge. While we have conducted further pre-training following the BERT-DK method proposed in [63] and observed catastrophic forgetting, the alternative BERT-PT method proposed in the same paper could yield a more robust Albert model. A method focusing on extracting common linguistic phenomena between different domains before domain-specialized pre-training is proposed in [77]. Results suggest that this method could reduce the duration of further pre-training significantly. Transferring knowledge during further pre-training from domains and languages with more resources to those with fewer resources is explored in [14], potentially widening the applicability of existing ABSA models. Finally, incorporating self-distillation with self-ensembling into the fine-tuning process as described in [76] may improve the efficacy of Albert when adapting to new domains.

There are several pathways by which this area of research may be fruitfully advanced.

Appendix A

Abbreviations Glossary

The following is a list of abbreviations frequently used in this report. Note that we do not include model names found in literature except Albert and BERT.

ABSA: Aspect-Based Sentiment Analysis

ACC: Accuracy

AE: Aspect Extraction

Albat: Albert Adversarially Trained

Albat-1LC: Albat with a One-layer Classifier

Albat-2LC: Albat with a Two-layer Classifier

Albat-3LC: Albat with a Three-layer Classifier

Albat PT: Albat Further Pre-trained

Albert: A Lite BERT

ASC: Aspect Sentiment Classification

ASE: Aspect Sentiment Evolution

BAT: BERT Adversarially Trained

BERT: Bi-directional Encoder Representations from Transformers

BERT-DK: BERT Further Pre-training using Domain Knowledge

BERT-MRC: BERT Further Pre-training using Machine Reading Comprehension

BERT-PT: BERT Post-Training

CLS: Classification Place-holder Token

CNN: Convolutional Neural Network

CON: Conflict

E2E-ABSA: End-to-End Aspect-Based Sentiment Analysis

FN: False Negative

FP: False Positive

GLUE: General Language Understanding Evaluation benchmark

GPU: Graphical Processing Unit

GRU: Gated Recurrent Unit

Lapt14: SemEval-2014 Challenge Task 4 Laptop domain dataset

LSTM: Long Short-Term Memory network

MAMS: Multi-Aspect Multi-Sentiment dataset

MF1: Macro-averaged F_1 Score

MHSA: Multi-Head Self-Attention

MLM: Masked Language Modeling

NEG: Negative

NEU: Neutral

NLP: Natural Language Processing

NSP: Next Sentence Prediction

POS: Positive

RACE: Large-scale Reading Comprehension Dataset From Examinations

RAM: Random Access Memory

ReLU: Rectified Linear Unit

Rest14: SemEval-2014 Challenge Task 4 Restaurant domain dataset

RNN: Recurrent Neural Network

RvNN: Recursive Neural Network

SEP: Segment Separator Place-holder Token

SQuAD: Stanford Question Answering Dataset

TN: True Negative

TP: True Positive

Unified: Restaurant domain reviews from [26]

Appendix B

E2E-ABSA Literature: F_1 Score Averaging Method

In Figure B.1 we have included a brief clarification of the averaging method used with the F_1 scores reported in [13], of which Mr. Xin Li was a co-author. The paper [13] compared its results with those of an earlier paper [26], so it is reasonable to assume that [26] also reported micro-averaged F_1 scores.

Re: Exploiting BERT for End-to-End Aspect-based Sentiment Analysis

lixin@se.cuhk.edu.hk <lixin@se.cuhk.edu.hk>
Mon 7/13/2020 11:21 AM
To: Williams, Daniel <daniel.williams.19@ucl.ac.uk>
Dear Daniel,

Thank you for your attention. The results reported in this paper are micro-averaged F1 scores.

Best,

Xin Li

Figure B.1: Email Correspondence with Mr. Xin Li

Appendix C

Classification Performance Metrics

To evaluate the performance of classification predictions for AE, ASC, and E2E-ABSA, we have used a number of metrics. Since AE is a binary classification task, we have reported the binary F_1 score. In contrast, ASC and E2E-ABSA involve multi-class classification predictions so we have reported accuracy and the macro-averaged F_1 score. In this appendix we explain how accuracy is equivalent to the micro-averaged F_1 score for multi-class classification.

C.1 Binary Classification Metrics

We first consider a binary classification task involving n examples. The classifier's decisions can be described as one of the following cases:

- true positive (TP), classified as positive and actually positive;
- true negative (TN), classified as negative and actually negative;
- false positive (FP), classified as positive but actually negative;
- false negative (FN), classified as negative but actually positive.

The following metrics evaluate the performance of the classifier [97, 98]:

Precision Of the examples classified as positive, how many are actually positive?

$$P = \frac{TP}{TP + FP}$$

Recall Of the actually positive examples, how many are classified as such?

$$R = \frac{TP}{TP + FN}$$

F_1 Score The harmonic mean of precision and recall:

$$F_1 = \frac{2}{P^{-1} + R^{-1}}$$

Accuracy Of all the examples, how many are classified correctly?

$$\alpha = \frac{TP + TN}{TP + TN + FP + FN}$$

C.2 Multi-class Classification Metrics

In multi-class classification we consider n examples labeled using one of c classes. This requires the use of unique case counters $\{TP_i, TN_i, FP_i, FN_i\}$ for each class i. Note that one example may belong to different case counters for different labels: given $c = 3$ classes, an example with true label 2 and predicted label 3 would contribute to TN_1, FN_2 and FP_3 simultaneously.

If we consider one class against all other classes, we may evaluate a binary classification metric for each of the c classes. For convenience, we instead report averages of the binary classification metrics. Several averaging methods exist, including macro-averaging and micro-averaging.

Macro-averaged Metrics Macro-averaging calculates the "one-versus-rest" binary classification metric for each class and returns their unweighted arithmetic mean [98].

Micro-averaged Metrics Micro-averaging considers the sum of $\{TP_i, TN_i, FP_i, FN_i\}$ across all classes i as inputs for a binary classification metric [98]. We will now show that micro-averaged precision, recall, F_1 score and accuracy are equivalent.

Recall that $P = \frac{TP}{TP+FP}$. Micro-averaged precision is given by

$$\hat{P} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i}.$$

The numerator $\sum_i TP_i$ equals the number of correct predictions across all classes. The denominator is the total number of true positives and false positives for each class; this is equal to the total number of examples n (since every example has been assigned a label regardless of its truthfulness). Therefore $\hat{P}$ is the ratio of the number of correctly-classified examples to the total number of examples, equal to micro-averaged accuracy $\hat{\alpha}$.

Similarly note that $R = \frac{TP}{TP+FN}$. Micro-averaged recall is given by

$$\hat{R} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i}.$$

The numerator $\sum_i TP_i$ equals the number of correct predictions across all classes. The denominator is the total number of true positives and false negatives for each class; this is equal to the total number of examples n (since every example is assigned a label). Therefore $\hat{R}$ is the ratio of the number of correctly-classified examples to the total number of examples, equal to micro-averaged accuracy $\hat{\alpha}$.

Since micro-averaged precision and recall are identical to micro-averaged accuracy $\hat{\alpha}$, the harmonic mean

$$\hat{F}_1 = \frac{2}{\hat{P}^{-1} + \hat{R}^{-1}} = \frac{2}{2\hat{P}^{-1}} = \frac{2}{2\hat{R}^{-1}}$$

is equal to both micro-averaged precision and recall. Hence the micro-averaged F_1 score is equal to micro-averaged accuracy.

Bibliography

[1] X. Wu, S. Lv, L. Zang, J. Han, and S. Hu, "Conditional bert contextual augmentation," in *International Conference on Computational Science*. Springer, 2019, pp. 84–95.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[3] J. Alammar. (2018) The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning). Accessed: 2020-09-09. [Online]. Available: https://jalammar.github.io/illustrated-bert/

[4] A. Chaudhary. (2020) Visual Paper Summary: ALBERT (A Lite BERT). Accessed: 2020-09-09. [Online]. Available: https://amitness.com/2020/02/albert-visual-summary/

[5] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," in *International Conference on Learning Representations*, 2019.

[6] A. Yadav and D. K. Vishwakarma, "Sentiment analysis using deep learning architectures: a review," *Artificial Intelligence Review*, pp. 1–51, 2019.

[7] A. Nazir, Y. Rao, L. Wu, and L. Sun, "Issues and challenges of aspect-based sentiment analysis: A comprehensive survey," *IEEE Transactions on Affective Computing*, 2020.

[8] H. H. Do, P. Prasad, A. Maag, and A. Alsadoon, "Deep learning for aspect-based sentiment analysis: a comparative review," *Expert Systems with Applications*, vol. 118, pp. 272–299, 2019.

[9] H. Wan, Y. Yang, J. Du, Y. Liu, K. Qi, and J. Z. Pan, "Target-Aspect-Sentiment Joint Detection for Aspect-Based Sentiment Analysis," in *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[10] J. Zhou, J. X. Huang, Q. Chen, Q. V. Hu, T. Wang, and L. He, "Deep learning for aspect-level sentiment classification: Survey, vision, and challenges," *IEEE Access*, vol. 7, pp. 78 454–78 483, 2019.

[11] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. 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. [Online]. Available: https://www.aclweb.org/anthology/S14-2004

[12] Y. Wang, A. Sun, M. Huang, and X. Zhu, "Aspect-level sentiment analysis using as-capsules," in *The World Wide Web Conference*, 2019, pp. 2033–2044.

[13] X. Li, L. Bing, W. Zhang, and W. Lam, "Exploiting BERT for End-to-End Aspect-based Sentiment Analysis," in *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, 2019, pp. 34–41.

[14] H. Xu, B. Liu, L. Shu, and P. S. Yu, "DomBERT: Domain-oriented language model for aspect-based sentiment analysis," *arXiv preprint arXiv:2004.13816*, 2020.

[15] R. He, W. S. Lee, H. T. Ng, and D. Dahlmeier, "An Interactive Multi-Task Learning Network for End-to-End Aspect-Based Sentiment Analysis," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 504–515.

[16] A. Alghunaim, M. Mohtarami, S. Cyphers, and J. Glass, "A Vector Space Approach for Aspect Based Sentiment Analysis," in *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, 2015, pp. 116–122.

[17] Y. Wang, Q. Chen, M. Ahmed, Z. Li, W. Pan, and H. Liu, "Joint inference for aspect-level sentiment analysis by deep neural networks and linguistic hints," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[18] D. M. Koupaei, T. Song, K. S. Cetin, and J. Im, "An assessment of opinions and perceptions of smart thermostats using aspect-based sentiment analysis of online reviews," *Building and Environment*, vol. 170, p. 106603, 2020.

[19] G. S. Chauhan and Y. K. Meena, "Domsent: Domain-specific aspect term extraction in aspect-based sentiment analysis," in *Smart Systems and IoT: Innovations in Computing*. Springer, 2020, pp. 103–109.

[20] M. E. Mowlaei, M. S. Abadeh, and H. Keshavarz, "Aspect-based sentiment analysis using adaptive aspect-based lexicons," *Expert Systems with Applications*, vol. 148, p. 113234, 2020.

[21] N. Nikolić, O. Grljević, and A. Kovačević, "Aspect-based sentiment analysis of reviews in the domain of higher education," *The Electronic Library*, 2020.

[22] M. Shams, N. Khoshavi, and A. Baraani-Dastjerdi, "Lisa: Language-independent method for aspect-based sentiment analysis," *IEEE Access*, vol. 8, pp. 31 034–31 044, 2020.

[23] C. R. Aydin and T. Güngör, "Combination of recursive and recurrent neural networks for aspect-based sentiment analysis using inter-aspect relations," *IEEE Access*, vol. 8, pp. 77 820–77 832, 2020.

[24] C. Ji and H. Wu, "Cascade architecture with rhetoric long short-term memory for complex sentence sentiment analysis," *Neurocomputing*, 2020.

[25] M. Al-Smadi, O. Qawasmeh, M. Al-Ayyoub, Y. Jararweh, and B. Gupta, "Deep recurrent neural network vs. support vector machine for aspect-based sentiment analysis of arabic hotels' reviews," *Journal of Computational Science*, vol. 27, pp. 386–393, 2018.

[26] X. Li, L. Bing, P. Li, and W. Lam, "A unified model for opinion target extraction and target sentiment prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6714–6721.

[27] K. Shuang, Q. Yang, J. Loo, R. Li, and M. Gu, "Feature distillation network for aspect-based sentiment analysis," *Information Fusion*, 2020.

[28] N. Liu and B. Shen, "Aspect-based sentiment analysis with gated alternate neural network," *Knowledge-Based Systems*, vol. 188, p. 105010, 2020.

[29] H. Xu, B. Liu, L. Shu, and S. Y. Philip, "Double Embeddings and CNN-based Sequence Labeling for Aspect Extraction," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018, pp. 592–598.

[30] L. Shu, H. Xu, and B. Liu, "Controlled cnn-based sequence labeling for aspect extraction," *arXiv preprint arXiv:1905.06407*, 2019.

[31] B. Wang and M. Liu, "Deep learning for aspect-based sentiment analysis," Stanford University, report, 2015.

[32] W. Xue and T. Li, "Aspect Based Sentiment Analysis with Gated Convolutional Networks," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 2514–2523.

[33] X. Hou, J. Huang, G. Wang, K. Huang, X. He, and B. Zhou, "Selective attention based graph convolutional networks for aspect-level sentiment classification," *arXiv preprint arXiv:1910.10857*, 2019.

[34] X. Gu, Y. Gu, and H. Wu, "Cascaded convolutional neural networks for aspect-based opinion summary," *Neural Processing Letters*, vol. 46, no. 2, pp. 581–594, 2017.

[35] Y. Liang, F. Meng, J. Zhang, J. Xu, Y. Chen, and J. Zhou, "An iterative knowledge transfer network with routing for aspect-based sentiment analysis," *arXiv preprint arXiv:2004.01935*, 2020.

[36] ——, "A dependency syntactic knowledge augmented interactive architecture for end-to-end aspect-based sentiment analysis," *arXiv preprint arXiv:2004.01951*, 2020.

[37] F. Ren, L. Feng, D. Xiao, M. Cai, and S. Cheng, "Dnet: A lightweight and efficient model for aspect based sentiment analysis," *Expert Systems with Applications*, p. 113393, 2020.

[38] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning based text classification: A comprehensive review," *arXiv preprint arXiv:2004.03705*, 2020.

[39] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, 2017, pp. 3856–3866.

[40] W. Zhao, H. Peng, S. Eger, E. Cambria, and M. Yang, "Towards Scalable and Reliable Capsule Networks for Challenging NLP Applications," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 1549–1559.

[41] Z. Chen and T. Qian, "Transfer capsule network for aspect level sentiment classification," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 547–556.

[42] C. Du, H. Sun, J. Wang, Q. Qi, J. Liao, T. Xu, and M. Liu, "Capsule network with interactive attention for aspect-level sentiment classification," 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)*, 2019, pp. 5492–5501.

[43] D. Tang, B. Qin, and T. Liu, "Aspect Level Sentiment Classification with Deep Memory Network," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 214–224.

[44] N. Liu and B. Shen, "Rememnn: A novel memory neural network for powerful interaction in aspect-based sentiment analysis," *Neurocomputing*, 2020.

[45] Y. Ma, H. Peng, and E. Cambria, "Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[46] Z. Li, Y. Wei, Y. Zhang, X. Zhang, and X. Li, "Exploiting coarse-to-fine task transfer for aspect-level sentiment classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4253–4260.

[47] Y. Qiang, X. Li, and D. Zhu, "Toward tag-free aspect based sentiment analysis: A multiple attention network approach," *arXiv preprint arXiv:2003.09986*, 2020.

[48] D. Zhang, Z. Zhu, Q. Lu, H. Pei, W. Wu, and Q. Guo, "Multiple interactive attention networks for aspect-based sentiment classification," *Applied Sciences*, vol. 10, no. 6, p. 2052, 2020.

[49] Y. Song, J. Wang, T. Jiang, Z. Liu, and Y. Rao, "Attentional encoder network for targeted sentiment classification," *arXiv preprint arXiv:1902.09314*, 2019.

[50] H. Yang, B. Zeng, J. Yang, Y. Song, and R. Xu, "A Multi-task Learning Model for Chinese-oriented Aspect Polarity Classification and Aspect Term Extraction," *arXiv preprint arXiv:1912.07976*, 2019.

[51] B. Zeng, H. Yang, R. Xu, W. Zhou, and X. Han, "Lcf: A local context focus mechanism for aspect-based sentiment classification," *Applied Sciences*, vol. 9, no. 16, p. 3389, 2019.

[52] D. Meškelė and F. Frasincar, "Aldonar: A hybrid solution for sentence-level aspect-based sentiment analysis using a lexicalized domain ontology and a regularized neural

attention model," *Information Processing & Management*, vol. 57, no. 3, p. 102211, 2020.

[53] Q. Lu, Z. Zhu, D. Zhang, W. Wu, and Q. Guo, "Interactive rule attention network for aspect-level sentiment analysis," *IEEE Access*, vol. 8, pp. 52 505–52 516, 2020.

[54] J. Zhou, Q. Chen, J. X. Huang, Q. V. Hu, and L. He, "Position-aware hierarchical transfer model for aspect-level sentiment classification," *Information Sciences*, vol. 513, pp. 1–16, 2020.

[55] S. Tulkens and A. van Cranenburgh, "Embarrassingly simple unsupervised aspect extraction," *arXiv preprint arXiv:2004.13580*, 2020.

[56] Z. Wu, Y. Li, J. Liao, D. Li, X. Li, and S. Wang, "Aspect-context interactive attention representation for aspect-level sentiment classification," *IEEE Access*, vol. 8, pp. 29 238–29 248, 2020.

[57] B. Huang and K. M. Carley, "Syntax-Aware Aspect Level Sentiment Classification with Graph Attention Networks," 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)*, 2019, pp. 5472–5480.

[58] X. Bai, P. Liu, and Y. Zhang, "Exploiting typed syntactic dependencies for targeted sentiment classification using graph attention neural network," *arXiv preprint arXiv:2002.09685*, 2020.

[59] K. Wang, W. Shen, Y. Yang, X. Quan, and R. Wang, "Relational graph attention network for aspect-based sentiment analysis," *arXiv preprint arXiv:2004.12362*, 2020.

[60] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI, blog, 2019.

[61] Z. Gao, A. Feng, X. Song, and X. Wu, "Target-dependent sentiment classification with BERT," *IEEE Access*, vol. 7, pp. 154 290–154 299, 2019.

[62] C. Sun, L. Huang, and X. Qiu, "Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 380–385.

[63] H. Xu, B. Liu, L. Shu, and S. Y. Philip, "BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 2324–2335.

[64] A. Karimi, L. Rossi, A. Prati, and K. Full, "Adversarial training for aspect-based sentiment analysis with BERT," *arXiv preprint arXiv:2001.11316*, 2020.

[65] X. Li, X. Fu, G. Xu, Y. Yang, J. Wang, L. Jin, Q. Liu, and T. Xiang, "Enhancing BERT representation with context-aware embedding for aspect-based sentiment analysis," *IEEE Access*, vol. 8, pp. 46 868–46 876, 2020.

[66] X. Wang, H. Xu, X. Sun, and G. Tao, "Combining Fine-Tuning with a Feature-Based Approach for Aspect Extraction on Reviews (Student Abstract)," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 10, 2020, pp. 13 951–13 952.

[67] M. R. Yanuar and S. Shiramatsu, "Aspect Extraction for Tourist Spot Review in Indonesian Language using BERT," in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE, 2020, pp. 298–302.

[68] M. Hu, S. Zhao, H. Guo, R. Cheng, and Z. Su, "Learning to Detect Opinion Snippet for Aspect-Based Sentiment Analysis," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, 2019, pp. 970–979.

[69] J. Yu and J. Jiang, "Adapting BERT for target-oriented multimodal sentiment classification," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 5408–5414.

[70] C. A. Putri, "Analisis sentimen review film berbahasa inggris dengan pendekatan bidirectional encoder representations from transformers," *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, vol. 6, no. 2, pp. 181–193, 2020.

[71] Y. Song, J. Wang, Z. Liang, Z. Liu, and T. Jiang, "Utilizing BERT intermediate layers for aspect based sentiment analysis and natural language inference," *arXiv preprint arXiv:2002.04815*, 2020.

[72] A. Rietzler, S. Stabinger, P. Opitz, and S. Engl, "Adapt or get left behind: Domain adaptation through BERT language model finetuning for aspect-target sentiment classification," *arXiv preprint arXiv:1908.11860*, 2019.

[73] Q. Jiang, L. Chen, R. Xu, X. Ao, and M. 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)*, 2019, pp. 6281–6286.

[74] A. Rogers, O. Kovaleva, and A. Rumshisky, "A primer in BERTology: What we know about how BERT works," *arXiv preprint arXiv:2002.12327*, 2020.

[75] R. Wang, H. Su, C. Wang, K. Ji, and J. Ding, "To tune or not to tune? how about the best of both worlds?" *arXiv preprint arXiv:1907.05338*, 2019.

[76] Y. Xu, X. Qiu, L. Zhou, and X. Huang, "Improving BERT fine-tuning via self-ensemble and self-distillation," *arXiv preprint arXiv:2002.10345*, 2020.

[77] C. Wang, M. Qiu, J. Huang, and X. He, "Meta fine-tuning neural language models for multi-domain text mining," *arXiv preprint arXiv:2003.13003*, 2020.

[78] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: Adapt language models to domains and tasks," *arXiv preprint arXiv:2004.10964*, 2020.

[79] C. M. Yeung, "Effects of inserting domain vocabulary and fine-tuning BERT for german legal language," Master's thesis, University of Twente, 2019.

[80] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[81] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.

[82] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2383–2392.

[83] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, "RACE: Large-scale ReAding Comprehension Dataset From Examinations," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 785–794.

[84] M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, and I. Androutsopoulos, "SemEval-2015 task 12: Aspect based sentiment analysis," in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, 2015, pp. 486–495.

[85] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, M. Al-Smadi, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq *et al.*, "SemEval-2016 task 5: Aspect based sentiment analysis," in *10th International Workshop on Semantic Evaluation (SemEval 2016)*, 2016.

[86] L. Dong, F. Wei, C. Tan, D. Tang, M. Zhou, and K. Xu, "Adaptive recursive neural network for target-dependent twitter sentiment classification," in *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers)*, 2014, pp. 49–54.

[87] M. Saeidi, G. Bouchard, M. Liakata, and S. Riedel, "SentiHood: Targeted Aspect Based Sentiment Analysis Dataset for Urban Neighbourhoods," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 1546–1556.

[88] D. Ekawati and M. L. Khodra, "Aspect-based sentiment analysis for Indonesian restaurant reviews," in *2017 International Conference on Advanced Informatics, Concepts, Theory, and Applications (ICAICTA)*. IEEE, 2017, pp. 1–6.

[89] R. Y. Reddy, R. R. R. Gangula, and R. Mamidi, "Dataset Creation and Evaluation of Aspect Based Sentiment Analysis in Telugu, a Low Resource Language," in *Proceedings of The 12th Language Resources and Evaluation Conference*, 2020, pp. 5017–5024.

[90] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2018, pp. 66–71.

[91] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," *arXiv preprint arXiv:1605.07725*, 2016.

[92] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proceedings of the* 25^{th} *International Conference on World Wide Web*, 2016, pp. 507–517.

[93] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[94] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune BERT for text classification?" in *China National Conference on Chinese Computational Linguistics*. Springer, 2019, pp. 194–206.

[95] J. Su, S. Yu, and D. Luo, "Enhancing aspect-based sentiment analysis with capsule network," *IEEE Access*, vol. 8, pp. 100 551–100 561, 2020.

[96] S. Yoosuf and Y. Yang, "Fine-Grained Propaganda Detection with Fine-Tuned BERT," in *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, 2019, pp. 87–91.

[97] L. Derczynski, "Complementarity, F-score, and NLP Evaluation," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 2016, pp. 261–266.

[98] Scikit-learn developers. (2019) 3.3 Metrics and scoring: quantifying the quality of predictions. Accessed: 2020-07-13. [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#multiclass-and-multilabel-classification