National College of Ireland

"Sarcasm Detection using Dilbert and Albert: An In-Depth Comparative Analysis with Bert"

MSc Research Project
Data Analytics

Rohit Gopal Wadhwani
Student ID: 21194645

School of Computing
National College of Ireland

Supervisor: Dr.Muslim Jameel Syed

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Rohit Gopal Wadhwani
Student ID:	21194645
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Dr.Muslim Jameel Syed
Submission Due Date:	14/08/2023
Project Title:	"Sarcasm Detection using Dilbert and Albert: An In-Depth Comparative Analysis with Bert"
Word Count:	7521
Page Count:	26

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL Internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use another author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Rohit Gopal Wadhwani
Date:	28th August 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on a computer.	☐

Assignments that are submitted to the Programme Coordinator's office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

"Sarcasm Detection using Dilbert and Albert: An In-Depth Comparative Analysis with Bert"

Rohit Gopal Wadhwani
21194645

Abstract

In today's communication landscape, sarcasm has become a prevalent way of interacting with one another. However, its written form presents a challenge as the absence of tone makes understanding sarcastic comments difficult. Accurately detecting sarcasm in written text is crucial for grasping the true sentiment behind the words. This ability will not only help people to find the true intention of the headlines but also will aid companies in improving their services for customers and will also help identify business gaps and driving exponential growth. To address this, machine learning models, especially transformers, have been more efficient in sarcasm detection. For this research, we have compiled a large dataset of sarcastic and non-sarcastic news headlines from 2 sources: the onion and huffingtonpost, as headlines are a prime training ground for sarcasm detection. Utilizing transformer-based models like ALBERT and DILBERT, we aim to compare their performance with BERT and give a detailed analysis for the same. While retaining high accuracy of 97%, and 98% and in sarcasm detection, comparing BERT, ALBERT, and DILBERT enables model selection that is optimized based on variables including computing resources, scalability, and training efficiency. I hope to offer fresh approaches to improve sarcasm detection precision and advance NLP techniques for textual content analysis by examining the strengths and shortcomings of each model. Finally, the research findings will contribute to more accurate sarcasm recognition and improved text understanding in a variety of applications.

1 Introduction

Individuals have been tempted to convey their views and feelings on the Internet since the emergence of the Internet in recent years. Exploring users' feelings via social media has been a focus of study for researchers all around the world. Due to the increasing usage of sarcasm in social networks such as Twitter and newspapers, as well as the anti-emotional polarity of sarcasm, sarcasm identification for comment phrases has emerged as an important study subject in machine learning.

Sarcasm has developed into a highly informal means of communication according to linguistic evolution. The use of mocking and feigned politeness is used to supposedly enhance animosity. Sarcasm may be picked up in a face-to-face discussion by paying attention to and studying the speaker's speech patterns, body language, and the context in which they are speaking. Additionally, a context-aware approach is required to get around these challenges because it is impossible to see or hear the speaker's emotion or tone. So, in this research, I have used News Headlines from the Kaggle website to detect

Sarcasm. In our project, distinguishing between general concern and sarcasm becomes particularly challenging since we cannot hear speech or discern emotions. However, the development of BERT has transformed the detection of sarcasm in written text. Before BERT, employing traditional deep learning or machine learning techniques like word2vec or recurrent neural networks (RNNs) proved arduous due to the inherent complexities of language.Words have diverse meanings in different circumstances, making it difficult to recognize sarcasm within a sentence. By capturing the complicated links between words, BERT's contextual embeddings significantly improved sarcasm detection, resulting in a more precise and effective examination of written text for sarcasm. With BERT, we now have a powerful instrument to solve the difficulty of detecting sarcasm in written communication, opening up exciting possibilities for analyzing sentiment in textual data. So, Bert for sarcasm detection for analyzing sentiments has been used in the previous paper by Gosavi (2022) which was highly effective and had shown greater accuracy than other techniques. BERT (Bidirectional Encoder Representations from Transformers) is a game-changing advancement in natural language processing. It is an extension of the basic encoder-decoder transformer architecture that uses a novel technique called self-supervised training. BERT can learn and construct contextual representations of words by performing masked language modeling and next-sentence prediction tasks. By applying self-supervised learning methodologies, BERT gains a more comprehensive understanding of the language, allowing it to grasp the complicated linkages and meanings of words within the context of a phrase. BERT is particularly effective in a number of language-related tasks due to this contextual awareness, making it a key milestone in the development of transformer-based models for NLP.

It uses two processes to create unique models for a range of activities: pre-training and fine-tuning. Bert is a key component of the Transformers. BERT is a transformer-based model that succeeds at capturing the contextual meaning of words in a sentence. When encoding each word, it considers the entire phrase and its context, allowing it to better understand language aspects such as sarcasm. A new model called DistilBERT has been pre-trained on the same dataset as BERT, but it is smaller and processes information more efficiently. DistilBERT is a compact, quick, inexpensive, and light Transformer model that has been instructed by a distilling BERT basis. It has 40% fewer parameters and runs 60% quicker than bert-base-uncased. During its self-supervised training, it learnt from the unannotated text by using the BERT base model as guidance. DistilBERT was able to leverage a large amount of publically available data that was automatically processed to generate inputs and labels using the BERT base model utilizing this method.

Since the introduction of BERT three years ago, natural language research has embraced a new paradigm, leveraging vast quantities of existing text to pre-train a model's parameters via self-supervision without the requirement for data annotation. As a result, rather than developing a machine-learning model for natural language processing (NLP) from scratch, I might start using a model that has already been pre-trained on a language. However, in order to improve this novel approach to NLP, it is necessary to understand precisely what factors influence language-understanding performance, such as the number of layers, the size of the hidden layer representations, learning criteria for self-supervision, or something entirely different. Google Research created the ALBERT language representation model. It is a BERT variant designed to train more rapidly and effectively while delivering at least as excellent performance on language comprehension tasks as BERT. The key innovation of ALBERT is its use of a parameter-sharing method to reduce the number of trainable parameters in the model. Performance is enhanced

despite utilizing fewer resources due to more effective use of training data and faster convergence during training. ALBERT, like BERT, is pre-trained on massive amounts of text data using a self-supervised learning technique to develop rich language representations. These previously trained models can then be upgraded for future tasks such as natural language processing (NLP).

In this project, I will compare Dilbert and Albert with Bert to find the best prediction and most efficient model to detect sarcasm for our news Headline Data set

1.1 Reserach Question

The above research problem motivates the following research question: **Are Dilbert and Albert better than Bert at detecting sarcastic texts in news headlines?**

This study's main objective is to compare the Distilled BERT (DILBERT) and A Lite BERT (ALBERT) language models to the traditional BERT model in order to evaluate their effectiveness and efficiency. The objective is to evaluate if these modern variations exceed BERT for sarcasm detection within our chosen dataset in terms of accuracy and processing speed. We want to identify the best model for boosting sarcasm detection through thorough investigation.

Our research will begin with a thorough literature review in which we will carefully assess earlier technology and datasets. This thorough evaluation not only provides an informative recap of earlier developments but also acts as a guide to ensure that our research trajectory is on course.

The next stage includes the precise use of the technique, which includes a sequence of planned research steps designed to achieve the goals of our study. This section will explain how to use the DILBERT and ALBERT models in a methodical manner before outlining a comparison with the widely used BERT model.

2 Related Work

Sarcasm is a type of communication that is often used in everyday life to favorably express negative information. Experts in languages, psychology, and cognitive science have investigated their ability for detecting sarcasm. In this paper, I will provide an overview of previous research on this topic.

2.1 Origin of Sarcasm Detection in Computing

Detecting sarcasm is one of the most challenging tasks in computing because it involves written text rather than verbal communication. Unlike humans, who can discern sarcasm through tone, NLP faces difficulties in this area. Before the advent of deep learning techniques, achieving accurate sarcasm detection seemed almost impossible. In the research paper by Davidov et al. (2010), the authors proposed a method that involved training a classifier (KNN) on a small labeled dataset of sarcastic and non-sarcastic words and then applying it to a larger unlabeled dataset. The classifier used distinct features such as syntax, lexicon, and semantics to identify sarcastic comments in both datasets. While the approach showed impressive results (75.6% accuracy) on the Amazon dataset, it struggled to provide remarkable results (43.6% accuracy) on the Twitter dataset due to more noise and non-standard language.This paper highlighted the differences in sarcasm

across languages and focused light on the difficulty of detecting sarcasm in a variety of linguistic conditions.

The following paper titled "ICWSM - A Great Catchy Name: Semi-Supervised Recognition of Sarcastic Sentences in Online Product Reviews" presents the SASI (Semi-supervised Algorithm for Sarcasm Identification) for detecting sarcastic sentences in online product reviews. The authors Tsur et al. (2010) conducted experiments using a dataset of Amazon reviews.There are two steps to the algorithm: semi-supervised pattern collection and sarcasm classification. It makes use of pattern-based features and punctuation-based features to classify sentences as either sarcastic or not. The SASI algorithm's performance was evaluated using the KNN machine learning model, and it was compared to a baseline heuristic.SASI achieved much higher precision in identifying sarcastic statements, according to the data. The authors also investigated the most effective features for detecting sarcasm and discovered that a mixture of subtle features gave the greatest results. They were astonished by the impressive 76.66% precision achieved by the algorithm for this dataset. The authors admited the algorithm's shortcomings, particularly in detecting subtle sarcasm. They also investigated the motivations for using sarcasm in online groups and social networks. Finally, the SASI system performed well in detecting sarcastic words in online product evaluations. Future studies should look into incorporating sarcasm recognition into review summaries and ranking systems, according to the authors.

The next paper, titled "Identifying Sarcasm in Twitter Messages" by (González-Ibánez et al.; 2011), tackles the complex task of detecting sarcasm in Twitter posts. The author employs Support Vector Machine (SVM) with sequential minimal optimization (SMO) and Logistic Regression on extensive Twitter datasets, achieving a commendable 71% accuracy. The study's show that machine learning approaches examine a number of variables, such as the use of emoticons, negation, and the frequency of remarks that are sarcastic. Notably, the study only considers English, showing the potential advantages of using deep learning methods to comprehend sarcasm in several languages. The article comes to the conclusion that deep learning shows promising gains in sarcasm detection, even if machine learning approaches have some constraints that prevented obtaining better accuracy.

The work "Sarcasm as a Contrast between a Positive Sentiment and a Negative Situation" focuses on detecting sarcasm in tweets and presents a boosting method to automatically learn positive sentiment phrases, negative situation phrases, and positive predictive words from Twitter data. The authors (Riloff et al.; 2013) evaluate their method on a manually annotated dataset and compare it to several baselines. For training and evaluation, the authors utilized a dataset of harsh tweets. The paper explains the iterative method used by the bootstrapping technique to grow the sets of positive sentiment and negative circumstance words. The authors also compare their method to an SVM classifier. The experimental results show that the suggested bootstrapping algorithm outperforms the baseline approaches in precision, recall, and F-score. Interestingly, the bootstrapped lists of phrases can identify sarcastic tweets that the SVM classifier misses.In addition, the authors discover that integrating two strategies in a hybrid way increases recall. Finally, the paper compares a promising strategy to identifying sarcasm in tweets using bootstrapping to other baselines and an SVM classifier. The study demonstrates that using several models and combining their outputs leads to better results, providing vital insights into the efficacy of the suggested methodology for detecting sarcasm in social media texts.

Rajadesingan established from the previously mentioned works that sarcasm detection extends beyond language analysis and can also be determined from user behavior. In the study by Rajadesingan et al. (2015), the authors introduced a novel approach that relies on behavioral modeling to analyze patterns in Twitter users' activities. This included examining factors such as the number of tweets, retweets, followers, and followings, as well as the length of their tweets. The focus of this approach was on a specific event, the 2014 World Cup. The authors used a SCUBA model (Sarcasm Classification Using a Behavioral Modeling Approach) with L1 regularized logistic regression. The SCUBA model achieved an impressive accuracy of 86.1%. However, the approach had limitations as it was specifically tailored to capture sarcasm based on the behavior of users during the World Cup event and may not be directly applicable to other datasets, such as news headlines or Amazon reviews.

Towards the conclusion of the origin of sarcasm detection, the paper titled "Who cares about sarcastic tweets? Investigating the impact of sarcasm on sentiment analysis" looks at how sarcasm affects sentiment analysis in tweets. The authors Maynard and Greenwood (2014)explore hashtag tokenization and sentiment analysis evaluations, while also devising rules to enhance sentiment analysis in the presence of sarcasm. The experiments are aimed at identifying sarcasm in tweets. The data show that correctly identifying sarcasm can significantly improve sentiment recognition; nonetheless, perfect accuracy remains a difficulty. In summary, the paper identifies sarcasm as a complex aspect of sentiment analysis that presents challenges for computer systems to accurately identify.

2.2 Deep learning techniques for sarcasm detection

In the year 2016, numerous authors began exploring the application of deep learning techniques for predicting sarcasm based on context. The motivation behind sarcasm detection was twofold: to comprehend a user's intended message more accurately and to enable companies/organizations to gauge the sentiment of their users. This vital knowledge could assist businesses in addressing any concerns or discontent with their services or products, allowing them to make required modifications and better meet the demands of their customers.

In 2016, Ghosh presented a remarkable paper (Ghosh and Veale; 2016) where they explored the detection of sarcasm in Twitter datasets using SVM and CNN-DNN-LSTM techniques. They emphasized the shortcomings of typical feature-based techniques in dealing with the complexities and variety of sarcasm. To address this, they opted for a large dataset and devised a model that combines convolutional and recurrent layers to extract features from the input text and capture its temporal and spatial dependencies. The authors achieved impressive results with high precision, recall, and F-score (91.9%, 92.3%, and 91.2%, respectively). This accomplishment might be credited to the use of several pre-processing techniques, such as spell-checking, tokenization, and stemming, that decreased noise and increased data quality. It is important to highlight, however, that their approach is built particularly for Twitter data and the English language, restricting its direct application to other datasets and languages.

In their study titled "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion, and sarcasm," Felbo et al. (2017) explored a unique approach to sentiment analysis using emojis. Given the popularity of emojis during that time, the researchers aimed to leverage them to predict both sarcasm and sentiments effectively. They utilized LSTM (Long Short-Term Memory) to process

extensive Twitter data enriched with emojis, seeking to understand the semantic and emotional meanings behind each emoji's usage in sentences. Remarkably, their approach surpassed traditional sentiment analysis methods, yielding impressive results with precision, accuracy, F score, and AUC reaching 85.5. These findings were among the most optimistic forecasts made for Twitter datasets. However, one significant shortcoming of this strategy was that it was only applicable to datasets including emojis, such as Twitter data, limiting its relevance to larger datasets lacking emoji annotations.

The research paper titled "Sentiments and Emotions Evoked by News Headlines of Coronavirus Disease (COVID-19) Outbreak" utilized a vast dataset comprising 141,208 news headlines related to COVID-19. These headlines were sourced from reputable news outlets like Reuters, BBC, and Yahoo News, among others, and collected through the COVID-19 repository at John Hopkins University. To analyze the sentiments conveyed by these headlines, the authors Aslam et al. (2020) employed sentiment analysis, a branch of natural language processing (NLP) that categorizes sentiments in opinions and reviews. Specifically, they used the Lexicon method recommended by Saif, which associates words with emotions and sentiments. The main objective was to determine the polarity of words, phrases, or entire documents as either positive or negative.Before performing the sentiment analysis, the authors completed data processing procedures such as converting headlines to text files, developing a news headline corpus, eliminating digits and excess whitespace, stemming words, and creating a Document Term Matrix. These methods guaranteed that the data was properly prepared for sentiment analysis, yielding useful insights into the emotional and sentimental components of COVID-19 news headlines.

In the paper "Twitter sentiment analysis with a deep neural network: An enhanced approach using user behavioral information" by Alharbi and de Doncker (2019), two Twitter datasets, SemEval-2016_1 and SemEval-2016_2, were utilized, containing 3694 and 1122 manually annotated tweets, respectively, with positive and negative labels. To label additional tweets, the SentiStrength algorithm was employed. The authors used 10-fold cross-validation to train their model, splitting the dataset into 10 subsets and performing classification on one while validating the others. Long Short-Term Memory (LSTM) models, a hybrid visual-textual model, and classic classifiers like Naive Bayes (NB) and Support Vector Machines (SVM) were all investigated in the study for sentiment analysis. LSTM models, which are known for their ability to handle long-term dependencies, performed brilliantly in sentiment categorization tasks. The joint visual-textual model combined a convolutional neural network for image sentiment analysis with a paragraph vector model for textual sentiment analysis. The experimental findings continuously proved the supremacy of deep learning classifiers, particularly CNN and LSTM, with CNN obtaining outstanding accuracy of 88.71%.

In the research paper titled "Sarcasm Detection Using MHA-BiLSTM" authored by Kumar et al. (2020) a novel deep neural network approach is proposed for sarcasm detection. The study used two datasets: one balanced with 5,000 sarcastic and 5,000 non-sarcastic comments, and one imbalanced with 5,000 sarcastic and 10,000 non-sarcastic comments. The authors compare their MHA-BiLSTM model to a support vector machine (SVM) model that detects sarcasm using locally produced features. Interestingly, their feature-rich SVM model achieves superior performance compared to previous works. The MHA-BiLSTM model consists of a word encoder layer and a sentence-level multi-head attention layer. Within a comment, the former captures contextual information from both directions, whilst the later concentrates on distinct parts of the statement to understand its underlying semantics. Notably, the authors show that the MHA-BiLSTM

model outperforms other models, including the SVM technique, in detecting sarcasm in comments.

In the paper by Pan et al. (2020), a novel method for detecting sarcasm in multi-modal data is proposed. The writers consider the contrast between various modes of communication, such as text and image, as well as the literal and figurative meanings of words within the same mode. To achieve this, they propose a two-stage model, first extracting features separately from each mode of communication and then combining them using a multi-modal fusion layer. The algorithm was trained on a huge dataset of sarcastic tweets with text and visual sarcasm annotations. Extensive testing on benchmark datasets demonstrated that our strategy outperforms previous state-of-the-art methods, resulting in extraordinary accuracy rates of more than 80%. Furthermore, the authors conducted research to investigate various components of their model, which revealed that intra- and inter-modal differences are crucial in detecting sarcasm.Overall, this study suggests a potential technique for detecting sarcasm in multimodal data that employs Multimodal ANN.

2.3 Transformer-based Sarcasm Detection

A transformer represents a neural network architecture specifically created for handling sequential data, including natural language text and time series data. It incorporates self-attention, a mechanism that intelligently assigns varying importance to different parts of the input data, enabling it to grasp context and meaning by capturing relationships within sequential data.Transformers are used in a variety of domains such as natural language processing (NLP) and computer vision (CV) researches. The model uses an advanced mathematical technique known as attention or self-attention to find the interdependencies and impacts between distant items in the data. To summarize, transformers use self-attention to successfully handle sequential information, making them useful tools for a wide range of data analysis tasks.

In their paper titled "Sarcasm Detection using Hybrid Neural Network," Misra and Arora (2019) present a novel dataset specifically designed for sarcasm detection. This dataset comprises news headlines sourced from both a sarcastic news website and a genuine news website, addressing previous limitations related to labeling and language found in Twitter-based datasets. Based on a modified version of a prior model, the authors specify a comprehensible Hybrid Neural Network architecture to address the sarcasm detection issue. The model incorporates pre-trained word embeddings from the word2vec model and utilizes both an LSTM module and an Attention module. The LSTM module helps encode the contextual information of words in a sentence, while the Attention module reweighs this encoded context. The model's training objective involves minimizing the cross-entropy error between predicted and actual labels.The results suggest that the proposed model beats a strong baseline in terms of classification accuracy by roughly 5%, demonstrating its usefulness in sarcasm detection.

In the study conducted by Potamias et al. (2020), remarkable results were achieved in sarcasm detection using Recurrent CNN (convolutional neural networks) RoBERTa. The authors selected a Twitter dataset and combined CNN with RoBERTa, comparing it with various other transformers and deep learning models like ELMo, USE, NBSVM, XLNet, BERT, UPF, ClaC, and DESC. RoBERTa demonstrated the highest accuracy and the least mean squared error (MSE) compared to the other models. This study demonstrates the usefulness of transformers in detecting sarcasm and offers the possibility of applying

similar algorithms to diverse datasets, such as news articles and customer evaluations.

The research conducted by Lee et al. (2020) introduces a novel method to enhance sarcasm detection models by incorporating unlabeled conversation contexts. Their approach involves using Bert-large+BiLSTM+NextVLAD, which employs BERT large, a powerful model with 24 layers and 336M parameters that enables deep comprehension of input data. Although BERT is large requires significant processing time and is not the most efficient option, it delivers exceptional accuracy. The authors achieved precision and accuracy levels over 90% in the Twitter dataset and results exceeding 80% in the Reddit dataset, which was obtained through a competition. Given BERT's amazing performance as one of the best Transformers, we intend to use its skills for sarcasm detection in our work.

In their recent paper, Shrivastava and Kumar Shrivastava and Kumar (2021) introduced an innovative approach to detect sarcasm in a social media text. Their hybrid model uses both semantic and syntactic aspects of the text to improve sarcasm detection accuracy. They compared the model's performance to that of other regularly used models, such as Support Vector Machine, Logistic Regression, Long Short-Term Memory, Convolutional Neural Networks, BiLSTM, and attention-based models, to assess its performance. Using a data file, the researchers fine-tuned the BERT model, resulting in an advanced classification module that efficiently distinguishes between sarcastic and non-sarcastic data. The process involves two parts: a BERT tokenizer and a BERT pre-trained model. Notably, their research outperformed all other deep learning models with an outstanding accuracy of 70%. While specifics about their study procedures were not revealed, their findings show the model's ability in detecting sarcasm, particularly when conducted on high-performance systems such as Google Colab, which provides large GPU and RAM resources for efficient execution.

Finally, multiple research papers studied the use of Twitter datasets for sentiment and sarcasm identification, but their results were not consistently better compared to other types of datasets. Recently, a transformative study by (Gosavi; 2022) emerged, focusing on leveraging transformer-based approaches for predicting sarcasm and sentiment in text. The researchers used BERT, a pre-trained transformer model, to encode input text from publicly available datasets, subsequently training and evaluating their algorithm.The results showed that their approach outperformed other cutting-edge models in terms of accuracy and F1 scores. Notably, the model performed exceptionally well at detecting positive and negative feelings in textual data. However, the algorithm struggled to recognize sarcasm in neutral statements, most likely due to the lack of asymmetry in such scenarios, making sarcasm detection more challenging.

2.4 Conclusion

As a result, this work highlights the utility of transformer-based models in detecting sentiment and sarcasm in textual data. Furthermore, the results beat all previous methodologies. Finally, analyzing its performance revealed a number of issues that needed to be addressed. The research's real-world consequences include the development of more accurate sentiment and sarcasm recognition systems based on natural language processing (NLP). These algorithms have the potential to be useful in evaluating social media discussion and customer feedback ratings in a variety of contexts.

3 Methodology

In the literature review of previous papers, a prominent trend is the extensive use of Twitter datasets for sarcasm detection. This decision was influenced by the dataset's public availability and ease of use for academics and students. The primary motivation behind focusing on Twitter data was to unravel the hidden sentiment behind sarcasm. Sarcasm often masks genuine emotions, and users use it to express their true feelings indirectly. For instance, when someone sarcastically remarks, "Oh, fantastic! My flight has been postponed," it actually reflects their frustration or disappointment with the flight delay. The sarcastic expression serves as a coping mechanism, allowing them to humorously convey their annoyance at the inconvenience caused by the disrupted travel plans.

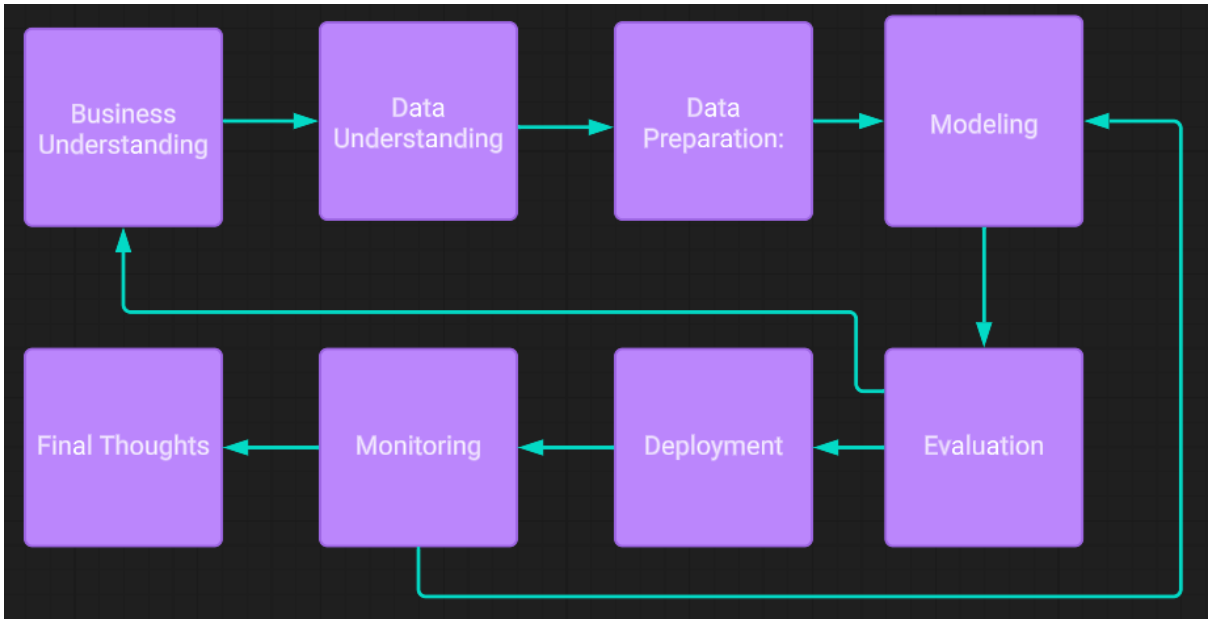

Figure 1: CRISP-DM Methodology

The implementation of "Sarcasm Detection using DiLiBERT and ALBERT: An In-Depth Comparative Analysis with BERT" followed the well-structured and widely recognized CRISP-DM methodology. This method is well-known for its planned and methodical approach to planning data mining initiatives. It ensures a successful and efficient implementation by outlining each step in the process.

Imagine a step-by-step approach like a well-organized roadmap which is also shown in figure 1. Firstly, I will clearly define the goal of the project, which was to detect sarcasm in text using the DiLiBERT and ALBERT models, and then compare their performance with the BERT model.

Next, For this research project, I chose a dataset called the "News Headline dataset" compiled by Rishabh Mishra, and it was obtained from the Kaggle website. Initially, the dataset was in a zip file, and after extracting it in my local computer , there were 2 files with json file which were then uploaded in Google Colab with the help of the pandas library.

The original format of the data was JSON, a structured data format often used for storing and exchanging data. The data was converted to a Data frame format using the

Pandas package to make it easier to work with and understand. The Data frame format arranges the data in a tabular form, making it more accessible for analysis and processing.

The dataset consists of three columns: "is_sarcastic," "Headline," and "article link." However, for this research, only two columns were primarily relevant: "headline" and "is_sarcastic." In the dataset, "headline" contains the news headlines in the form of text strings, and "is_sarcastic" is a binary attribute. In this context, the value 1 in the "is_sarcastic" column represents a sarcastic news headline, while the value 0 denotes a non-sarcastic news headline.

The study focuses on using this dataset to train and test models for detecting sarcasm. I hope to construct models that can accurately recognize and discriminate sarcastic news headlines from non-sarcastic ones by working with the "headline" and "is_sarcastic" columns, so giving useful insights to the field of natural language understanding and sentiment analysis.

Once they had the data ready, I Preprocessed the text data, including lowercasing, tokenization, and padding/truncation to ensure a consistent format for model input. Also, I created labels (target variable) that correspond to each headline's sarcasm status. Then I Split the data into training, validation, and testing sets to evaluate the model's performance.

The next crucial step was training the models. They selected DiLiBERT, ALBERT, and BERT as the architectures for sarcasm detection and fine-tuned them on the prepared data. This procedure involved enhancing the models' pre-trained information for the sarcasm detection task.

As these models were overfitting, so I used a method called early stopping which will help our model to do overfitting and will give us accurate results.

After training, I will then evaluate all the models on the validation set that was split earlier to choose the best-performing model based on appropriate evaluation metrics like accuracy, precision, recall, and ROC-AUC. This enabled me to choose the best-performing model based on a variety of metrics, ensuring that it was capable of recognizing sarcasm.

I analyzed the model's performance throughout the process to guarantee it remains accurate and successful in real-world settings as well.

I was able to conduct a well-organized and extensive analysis of sarcasm detection utilizing DiLiBERT, ALBERT, and BERT models using the CRISP-DM approach, offering useful insights and results for future research and practical applications.

3.1 Data set selection

I will focus on sarcasm detection using news headlines as the primary dataset in this research. While some previous studies utilized Twitter datasets with hashtag-based labels, these datasets can be noisy and limited in terms of language variety. Furthermore, tweets often include replies, requiring contextual information for accurate sarcasm recognition. To address these limitations, I will utilize the News Headlines dataset for Sarcasm Detection, sourced from two news websites. The dataset includes headlines from TheOnion, which produces sarcastic versions of current events, and non-sarcastic news headlines from HuffPost.I intend to use this dataset to investigate the efficacy of several models in identifying sarcasm in news headlines. I may also consider incorporating more datasets to test the models' robustness across different data sources. Kaggle's popularity and ease of use makes it an one of the excellent platforms for gathering and analyzing data.

3.1.1 This news dataset has the following advantages over the existing Twitter datasets:

In news headlines, spelling errors or informal language are rare as they are crafted by professionals with great care. This increases the availability of pre-trained embeddings and makes language analysis easier. Additionally, compared to Twitter datasets, the labels we obtain from The Onion, a satirical news outlet, are of higher quality with minimal noise. The news headlines collected are independent statements, unlike tweets that may be responses to others, simplifying the process of identifying genuine sarcastic elements. These factors contribute to the dataset's reliability and make it conducive for accurate sarcasm detection research.

3.1.2 Actual Content of Data set

In Figure 2 we will display small amount of data

	article_link	headline	is_sarcastic
0	https://www.huffingtonpost.com/entry/versace-b...	former versace store clerk sues over secret 'b...	0
1	https://www.huffingtonpost.com/entry/roseanne-...	the 'roseanne' revival catches up to our thorn...	0
2	https://local.theonion.com/mom-starting-to-fea...	mom starting to fear son's web series closest ...	1
3	https://politics.theonion.com/boehner-just-wan...	boehner just wants wife to listen, not come up...	1
4	https://www.huffingtonpost.com/entry/jk-rowlin...	j.k. rowling wishes snape happy birthday in th...	0

Figure 2: Snippet of the data

Each record consists of three attributes:
is_sarcastic: 1 if the record is sarcastic otherwise 0
headline: the headline of the news article
article_link: link to the original news article. Useful in collecting supplementary data

The dataset consists of 26710 total observations as well as the three Features Headline, Sarcastic (a binary feature), and Link (the news link). After that, the dataset will be used for additional analysis. This news dataset was chosen due to the text's authenticity, which included both sarcastic and non-sarcastic content. [1]

3.2 Data Pre-processing

Once the dataset was imported into the Google colab notebook, we proceeded with data pre-processing pipelines. The essential libraries employed were pandas, numpy, and NTLK and deep learning frameworks like TensorFlow, PyTorch, and Keras. Additionally, I have installed some additional packages like transformers. The data, in JSON format, was read using the pandas library. During the initial analysis, we determined the total word count and obtained value counts for both sarcastic and non-sarcastic labels. Subsequently, we conducted a basic exploratory analysis. To prepare the data for further analysis, we utilized NLP pipelines to remove stop words, punctuation, and other

[1] Data Collection: `https://www.kaggle.com/datasets/rmisra/news-headlines-dataset-for-sarcasm-detectio`

unnecessary elements. The cleaned data was then divided into separate data frames for sarcastic and non-sarcastic instances to gain a better understanding of the dataset.

In Figure 3 and Figure 4 we will display the split of data

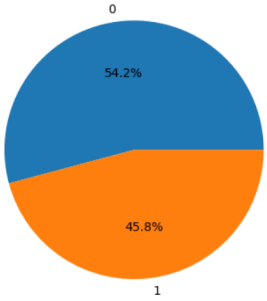

Figure 3: Data Split %

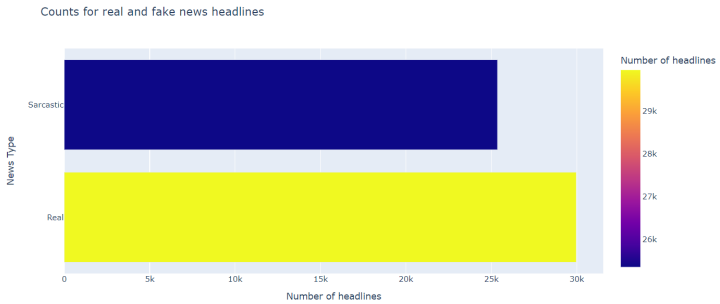

Figure 4: Total news headlines

3.3 Data Analyzing

We have converted the data from JSON to data frame as mentioned before, as we were analyzing the data we can see that words in the news headlines plays a vital role in the detection of sarcasm and so for this we used word cloud to get an idea of the words which were used the most and for the words which were used in headlines were sarcastic or not.

3.3.1 Word Cloud

From the figures Figure 5 and Figure 6 we can understand that trump, new, say, and etc. were used in such kind of headlines where it was not meant to be a sarcastic headline. But on the other hand, we can see new, man, nation and etc were used for sarcastic headlines. If we can understand that new words were used for both sarcastic and non-sarcastic headlines. So it is not easy to understand sarcasm just by words.

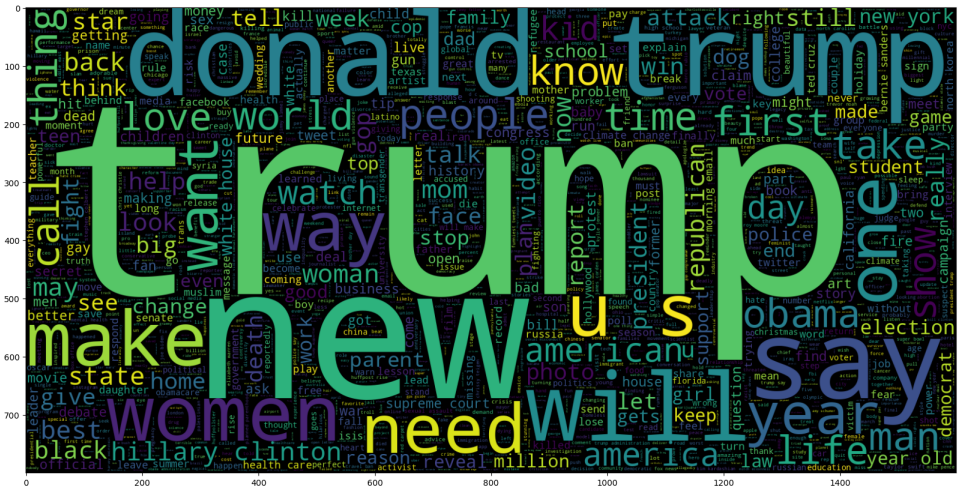

Figure 5: Words used which are not Sarcastic

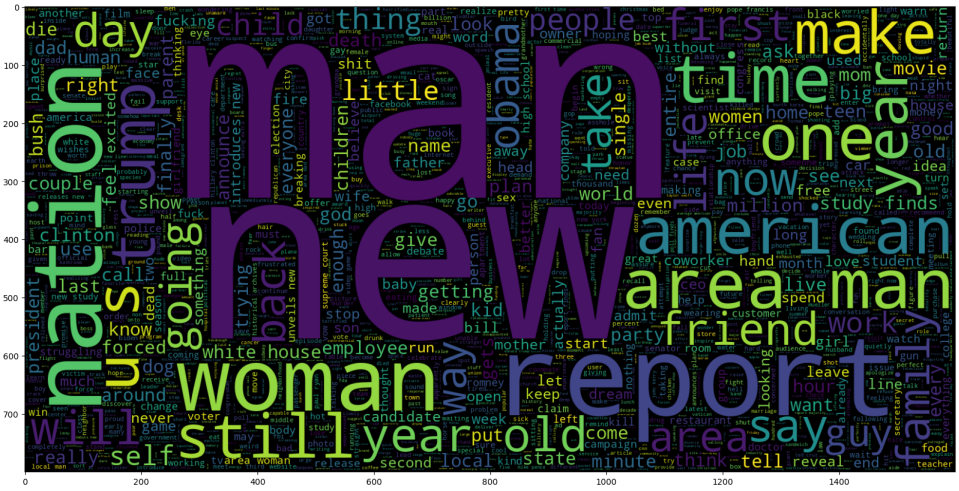

Figure 6: Words which are used in Sarcastic headlines

3.3.2 Average word length in a text

I have analyzed the text length of the news headlines and wanted to check if there is any differentiation between the average word length of the sarcastic headlines and non-sarcastic headlines, but from the figure Figure 7, we can justify that the sarcastic headlines text length was almost the same as non-sarcastic word length

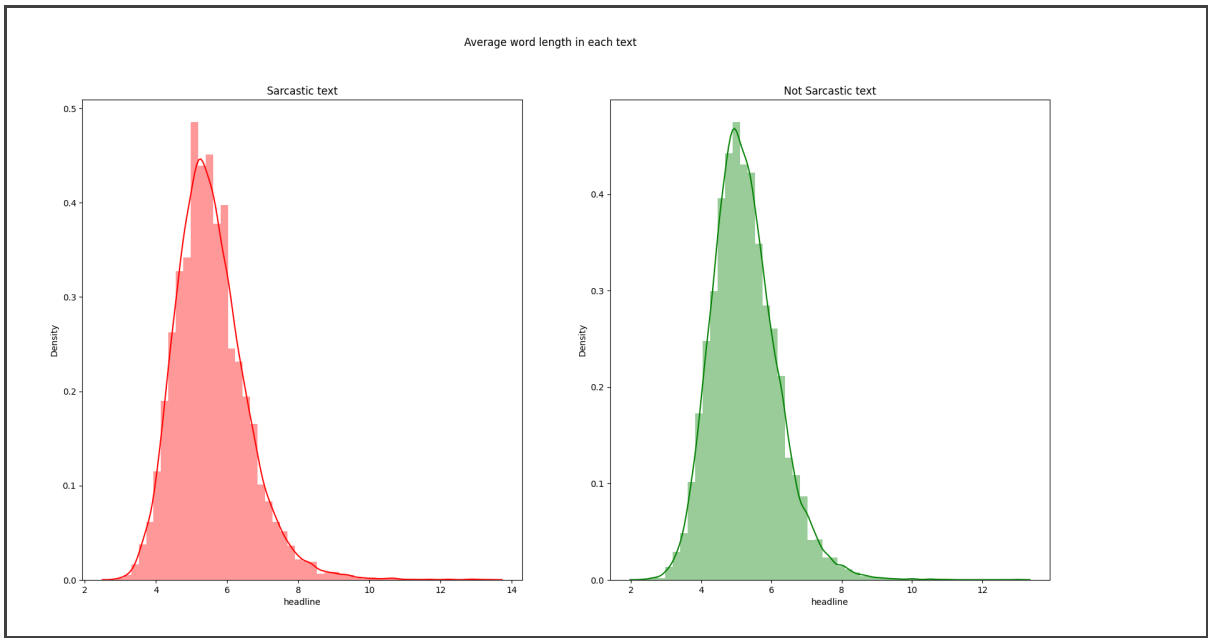

Figure 7: Word length

3.3.3 Headlines length Distribution and removal of outliers

It is very important to remove outliers that affect our machine learning model, So we can see that we have one outlier which was calculated by the headline sentence in the

figure Figure 8 given below, we can also analyze and can state that the length of the sarcastic headline is usually more than the non-sarcastic headlines. This will help our machine learning model to discriminate between the headlines and give a good accuracy for detecting sarcastic headlines.

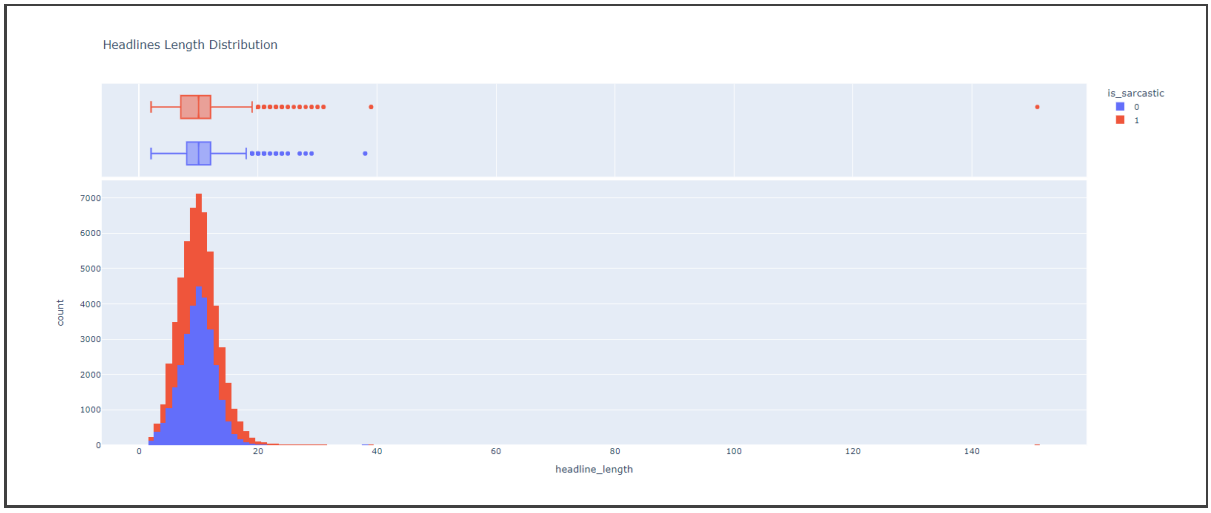

Figure 8: headlines length

4 Design Specification

In this research paper, we will be using 3 models that are a pre-trained encoder transformers model which are BERT(Bidirectional Encoder Representations from Transformers), Distilled BERT, and A-lite BERT whose parameters are discussed below:

- BERT:12-layer, 768-hidden, 12-heads, 110M parameters. Trained on lower-cased English text
- DILBERT: 6-layer, 768-hidden, 12-heads, 66M parameters. The DistilBERT model distilled from the BERT model bert-base-uncased checkpoint
- ALBERT:12 repeating layers, 128 embedding, 768-hidden, 12-heads, 11M parameters. ALBERT base model with no dropout, additional training data, and longer training

In this project, as computational needs were not matched by my personal computer so I have used google collab [2] which is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs. In which I have used GPU as a hardware accelerator and T4 as the GPU type, a snippet 9 is shown below.

Using the Tesla T4 GPU in Google Colab provides significant advantages over a normal CPU runtime session.The T4 GPU is a strong hardware accelerator that is optimized for parallel processing, making it an excellent choice for deep learning tasks. With GPU

[2]colab: `https://colab.google/`

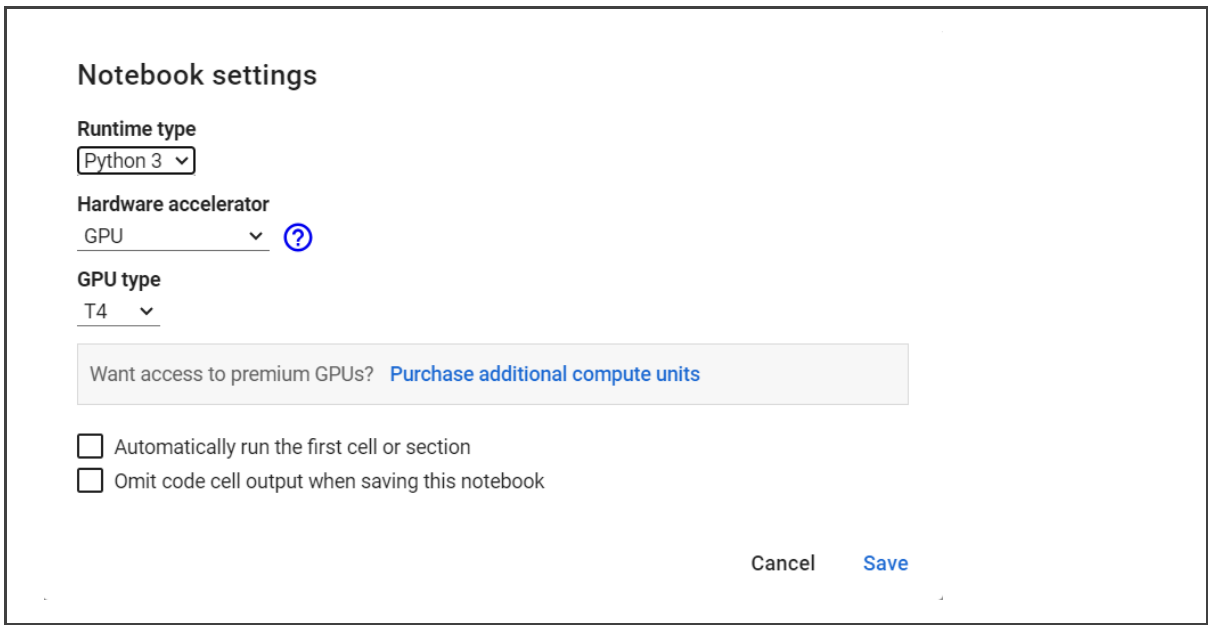

Figure 9: Run time Type

acceleration, training complex neural networks is much faster compared to CPU-only sessions, reducing training times from hours to minutes. The T4 GPU also supports larger datasets and models, allowing for advanced and accurate results. In contrast, a normal CPU session lacks the specialized processing capabilities of a GPU, leading to longer training times and limitations on the scale and complexity of deep learning tasks.

5 Implementation

For sarcasm detection in this research, we have built 3 pre-trained deep learning models which were BERT, DILBERT, and ALBERT which will be implemented on the news headlines clean dataset. First, we will use a BERT-based binary classification model for sarcasm detection.It implements the model using the TensorFlow library. It uses the 'bert-base-uncased' pre-trained BERT model as an encoder to convert input word IDs into contextualized word embeddings. The input consists of word IDs represented by a 16-token sequence.

The BERT encoder produces embeddings, and the Lambda layer extracts the first token's representation (i.e., [CLS] token) which contains a summary of the whole sequence. A dense layer with 128 units and ReLU activation is applied, followed by a dropout layer to prevent overfitting. Finally, a dense layer with 1 unit and a sigmoid activation function is used for binary classification, predicting the probability of the input being sarcastic or non-sarcastic. If validation loss does not improve for successive 5 epochs, early stopping with the patience of 5 epochs is employed to cease training.

The resulting model is a BERT-based binary classifier for sarcasm detection, taking tokenized input sequences and returning a probability score indicating the likelihood of the input being sarcastic.

I will be using the same fine-tuning for the other 2 models which are DILBERT, and ALBERT.It implements the model using the TensorFlow library. The ALBERT encoder

is loaded from the 'albert-base-v2' pre-trained model and the DILBERT encoder is loaded from the "distilbert-base-uncased" pre-trained model. The model's architecture consists of an input layer that takes word IDs as input, followed by the BERT encoder layer to encode the input text. The output of BERT is passed through a Dense layer with 128 units and a ReLU activation function. A dropout layer with a dropout rate of 0.2 is added to prevent overfitting. The final output layer is a Dense layer with 1 unit and a sigmoid activation function, suitable for binary classification. The model aims to predict whether a given headline is sarcastic (1) or non-sarcastic (0). If validation loss does not improve over the next 5 epochs, early stopping with the patience of 5 epochs is employed to halt training.

5.1 Application Phase

During this stage, the three models underwent evaluation using various metrics such as accuracy, precision, recall, and F1 score to assess their performance. Additionally, a mix of sarcastic and non-sarcastic sentences was inputted into the neural network to gauge how well the model functioned.

6 Evaluation

6.1 Case Study 1: BERT

The model design has been explained in detail in the above section. I will implement the model using the training data, represented by train_ids (input sequences) and train_labels (corresponding target labels), which is used to train the model over 20 epochs. During training, progress updates are displayed as indicated by verbose = 1. The training data is divided into batches of size 64 (batch_size = 64), and an early stopping callback (early_stop) is employed to potentially halt training if the model's performance on a validation dataset (represented by test_ids and test_labels) stops improving. This training process aims to optimize the model's parameters based on the provided training data and labels while monitoring its performance on the validation set to prevent overfitting.

```
735/735 [==============================] - 243s 258ms/step - loss: 0.2609 - accuracy: 0.8886 - val_loss: 0.1507 - val_accuracy: 0.9423
Epoch 2/20
735/735 [==============================] - 187s 255ms/step - loss: 0.1177 - accuracy: 0.9564 - val_loss: 0.1060 - val_accuracy: 0.9604
Epoch 3/20
735/735 [==============================] - 176s 240ms/step - loss: 0.0520 - accuracy: 0.9817 - val_loss: 0.0889 - val_accuracy: 0.9725
Epoch 4/20
735/735 [==============================] - 177s 241ms/step - loss: 0.0270 - accuracy: 0.9909 - val_loss: 0.0940 - val_accuracy: 0.9746
Epoch 5/20
735/735 [==============================] - 176s 239ms/step - loss: 0.0161 - accuracy: 0.9948 - val_loss: 0.0594 - val_accuracy: 0.9851
Epoch 6/20
735/735 [==============================] - 176s 239ms/step - loss: 0.0133 - accuracy: 0.9956 - val_loss: 0.0734 - val_accuracy: 0.9811
Epoch 7/20
735/735 [==============================] - 176s 240ms/step - loss: 0.0105 - accuracy: 0.9965 - val_loss: 0.0746 - val_accuracy: 0.9818
Epoch 8/20
735/735 [==============================] - 176s 239ms/step - loss: 0.0082 - accuracy: 0.9972 - val_loss: 0.0707 - val_accuracy: 0.9843
Epoch 9/20
735/735 [==============================] - 175s 239ms/step - loss: 0.0062 - accuracy: 0.9978 - val_loss: 0.0684 - val_accuracy: 0.9873
Epoch 10/20
735/735 [==============================] - 176s 239ms/step - loss: 0.0062 - accuracy: 0.9979 - val_loss: 0.0710 - val_accuracy: 0.9857
```

Figure 10: BERT- model and their accuracies and loss values

from figure 10 we can see that it took 10 epochs for the model to reach validation accuracy at 99% and due to early stopping which is a method to avoid the model from overfitting. we have done early stopping on validation loss. we can also see the same

graphs for our accuracy 11 on the training and testing dataset acquiring 99 and 98% accuracy and the same for loss 12 on the training and testing dataset and acquiring as low as 0.0062 and 0.071 loss.

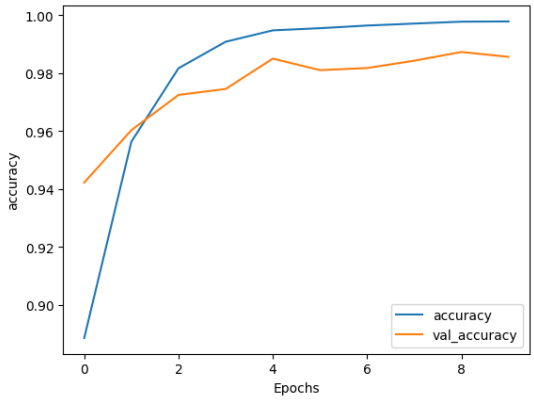

Figure 11: BERT Model Accuracy over 10 Epochs

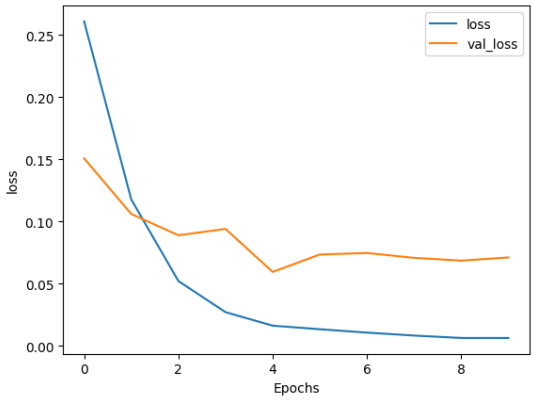

Figure 12: BERT Model loss over 10 Epochs

In figure 13 we can see that values in prediction are decimal points, so converting the decimal system by converting the values to 1 if it is greater than 0.5 and 0 to less than 0.5 to get a clear picture and help us to analyze more efficiently as shown in the figure 14

```
array([[1.9844949e-06],
       [1.6871287e-05],
       [9.9999607e-01],
       [9.9999654e-01],
       [9.9999833e-01],
       [4.8175636e-05],
       [9.9998105e-01],
       [9.9974185e-01],
       [2.6028870e-06],
       [1.2346713e-02]], dtype=float32)
```

Figure 13: Prediction Values Before

	Actual	Predicted
0	0	0
1	0	0
2	1	1
3	1	1
4	1	1
...	...	...

Figure 14: Prediction Values After

We can also see the results in the given figures 15 and 16 from this we can understand that Bert was struggling to find sarcastic comments and predicted non-sarcastic comments which he predicted wrong for 93 such headlines and where it was not sarcastic model predicted sarcastic was just 26, which is 4x times less.

	precision	recall	f1-score
0	0.99	0.99	0.99
1	0.98	0.99	0.98
accuracy			0.99
macro avg	0.99	0.99	0.99
weighted avg	0.99	0.99	0.99

Figure 15: Classification report for BERT Model

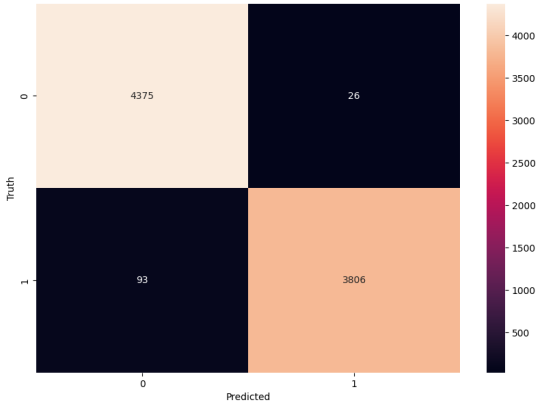

Figure 16: -Confusion Matrix for BERT model

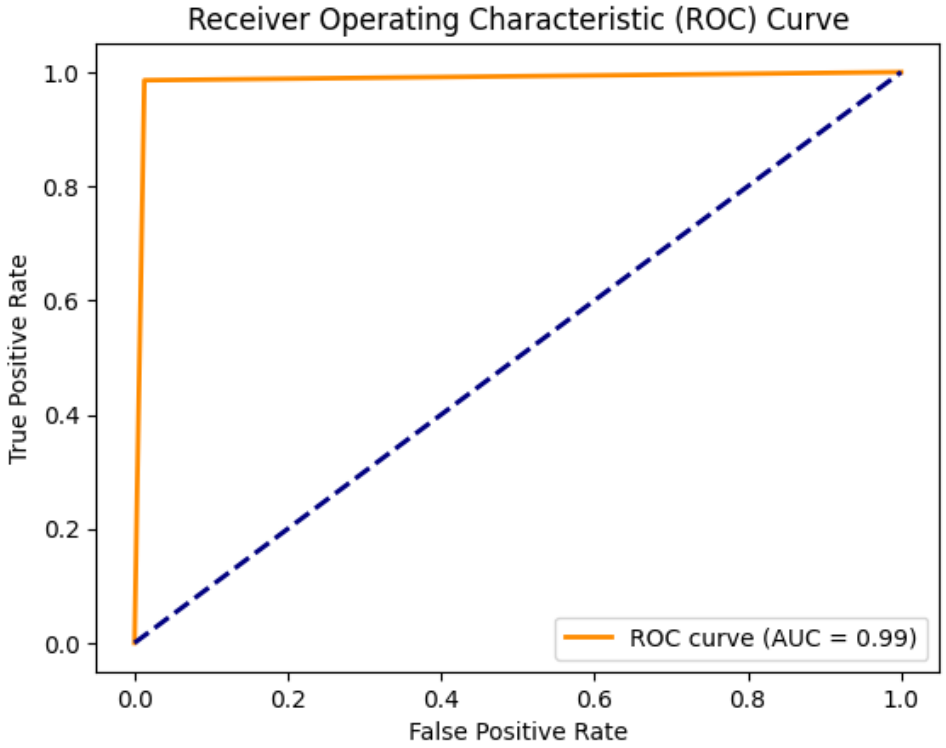

Figure 17: ROC Curve for BERT model

6.2 Case Study 2: ALBERT

While we discussed BERT earlier, with the same configurations for the model with the same fine tunning and using a pre-trained model "albert-base-v2" which tends to use 11M parameters but it takes the same amount of time to run 1 epoch as compared to Bert with 1 epoch 18. As compared to BERT, ALBERT has achieved 98% 19 of accuracy on the training dataset and 98% on the validation dataset with 11 epochs out of 20 epochs. As we can see in 20 the loss was as low as 0.0077 on the training dataset and 0.0964 on the testing dataset.

```
735/735 [==============================] - 213s 263ms/step - loss: 0.4096 - accuracy: 0.7848 - val_loss: 0.2201 - val_accuracy: 0.9137
Epoch 2/20
735/735 [==============================] - 182s 247ms/step - loss: 0.1564 - accuracy: 0.9425 - val_loss: 0.1331 - val_accuracy: 0.9512
Epoch 3/20
735/735 [==============================] - 171s 233ms/step - loss: 0.1192 - accuracy: 0.9539 - val_loss: 0.1712 - val_accuracy: 0.9480
Epoch 4/20
735/735 [==============================] - 180s 245ms/step - loss: 0.0568 - accuracy: 0.9814 - val_loss: 0.1193 - val_accuracy: 0.9671
Epoch 5/20
735/735 [==============================] - 180s 245ms/step - loss: 0.0300 - accuracy: 0.9906 - val_loss: 0.0983 - val_accuracy: 0.9740
Epoch 6/20
735/735 [==============================] - 169s 230ms/step - loss: 0.0230 - accuracy: 0.9928 - val_loss: 0.0823 - val_accuracy: 0.9771
Epoch 7/20
735/735 [==============================] - 179s 244ms/step - loss: 0.0154 - accuracy: 0.9958 - val_loss: 0.0909 - val_accuracy: 0.9808
Epoch 8/20
735/735 [==============================] - 170s 232ms/step - loss: 0.0126 - accuracy: 0.9964 - val_loss: 0.0901 - val_accuracy: 0.9790
Epoch 9/20
735/735 [==============================] - 180s 245ms/step - loss: 0.0107 - accuracy: 0.9966 - val_loss: 0.0900 - val_accuracy: 0.9804
Epoch 10/20
735/735 [==============================] - 180s 245ms/step - loss: 0.0105 - accuracy: 0.9964 - val_loss: 0.0914 - val_accuracy: 0.9830
Epoch 11/20
735/735 [==============================] - 180s 245ms/step - loss: 0.0077 - accuracy: 0.9974 - val_loss: 0.0964 - val_accuracy: 0.9827
```

Figure 18: ALBERT- model and their accuracies and loss values

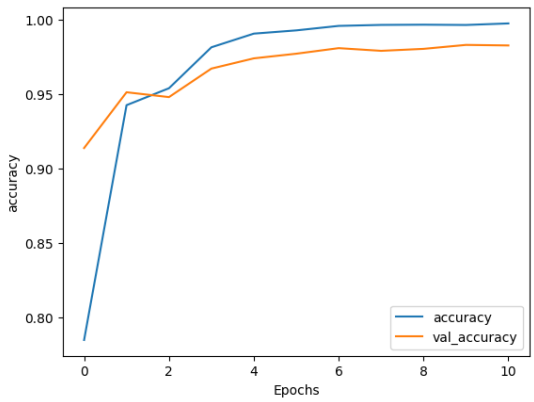

Figure 19: ALBERT Model Accuracy over 11 Epochs

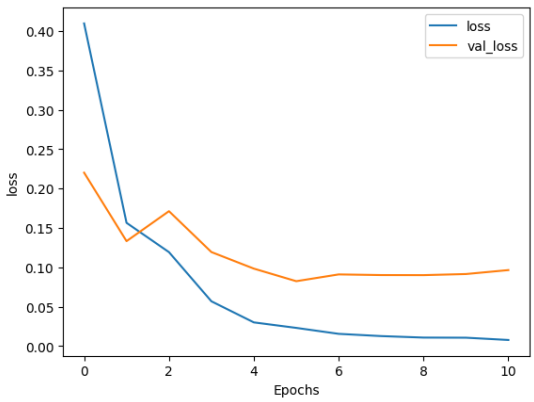

Figure 20: ALBERT Model Loss over 11 Epochs

From the figure 22 23we can see that for evaluation, we have used the ROC curve and Confusion matrix we can analyze that ALBERT was equally struggling to find sarcasm and non-sarcasm detection and was not too fast with less number of parameters.

	precision	recall	f1-score	support
0	0.98	0.99	0.98	4466
1	0.98	0.98	0.98	3834
accuracy			0.98	8300
macro avg	0.98	0.98	0.98	8300
weighted avg	0.98	0.98	0.98	8300

Figure 21: Classification report for Albert Model

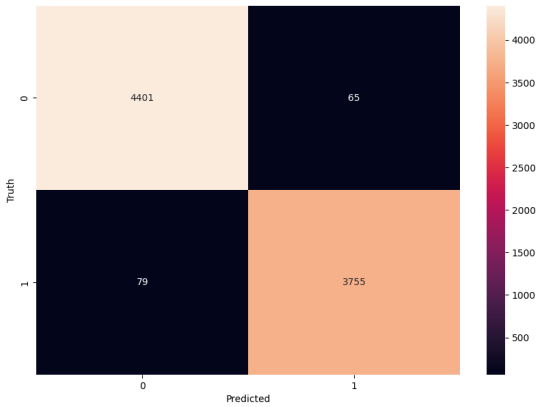

Figure 22: Confusion Matrix for ALBERT model

6.3 Case Study 3: DILBERT

As we have discussed BERT and ALBERT, DILBERT is a distilled bert with 66M parameters and is the fastest model in our project taking the time of approx-90-100s for each epoch as shown in figure 24. From this we can understand that DILBERT might have been designed to take better advantage of parallel computing, leading to faster processing on multi-core CPUs or GPUs. But it takes 16 epochs with early stopping to avoid overfitting shown in the below figures 25 26 and achieved 99% on the training dataset and validation with 97% accuracy and with loss of 0.0124 and 0.138 which is much higher than other 2 models compared.

The model stopped at the 16th epoch as it saw the validation loss was fluctuating as shown in figure 26, so after having the patience of 5 epochs it stopped the model to train more and test it.

As we can see through the confusion matrix 27 and 28 Dilbert has an accuracy of 98% which is almost the same accuracy as Bert, but it was not only struggling with sarcastic headlines but also non-sarcastic headlines to predict it properly

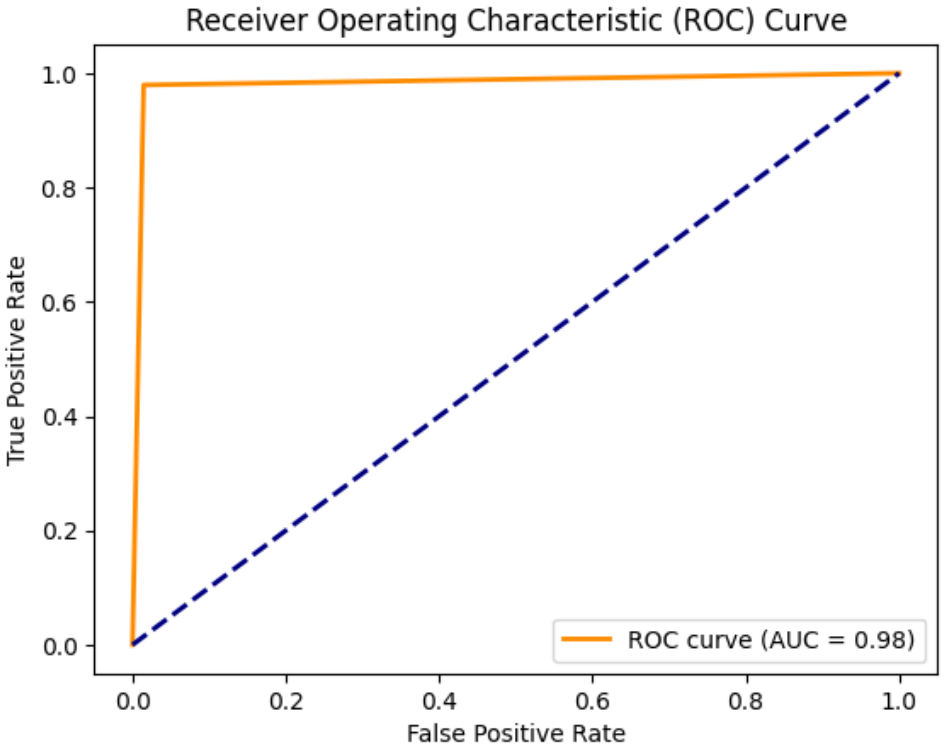

Figure 23: ROC Curve for ALBERT model

```
Epoch 9/20
735/735 [==============================] - 93s 126ms/step - loss: 0.0376 - accuracy: 0.9872 - val_loss: 0.1158 - val_accuracy: 0.9698
Epoch 10/20
735/735 [==============================] - 92s 125ms/step - loss: 0.0319 - accuracy: 0.9886 - val_loss: 0.1355 - val_accuracy: 0.9680
Epoch 11/20
735/735 [==============================] - 91s 124ms/step - loss: 0.0275 - accuracy: 0.9906 - val_loss: 0.1058 - val_accuracy: 0.9723
Epoch 12/20
735/735 [==============================] - 92s 125ms/step - loss: 0.0209 - accuracy: 0.9928 - val_loss: 0.1407 - val_accuracy: 0.9690
Epoch 13/20
735/735 [==============================] - 92s 125ms/step - loss: 0.0188 - accuracy: 0.9936 - val_loss: 0.1233 - val_accuracy: 0.9737
Epoch 14/20
735/735 [==============================] - 92s 125ms/step - loss: 0.0147 - accuracy: 0.9947 - val_loss: 0.1340 - val_accuracy: 0.9737
Epoch 15/20
735/735 [==============================] - 92s 125ms/step - loss: 0.0142 - accuracy: 0.9950 - val_loss: 0.1246 - val_accuracy: 0.9746
Epoch 16/20
735/735 [==============================] - 97s 132ms/step - loss: 0.0124 - accuracy: 0.9958 - val_loss: 0.1380 - val_accuracy: 0.9765
```

Figure 24: DILBERT- model and their accuracies and loss values

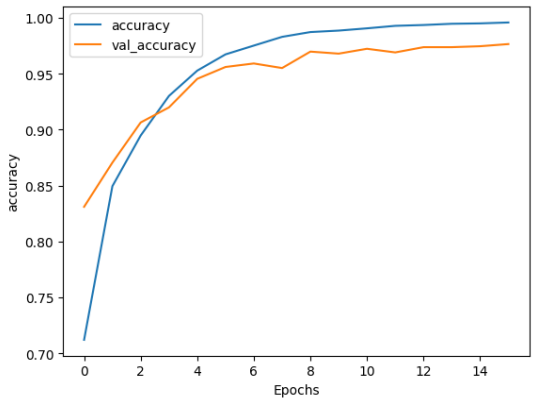

Figure 25: DILBERT Model Accuracy over 16 Epochs

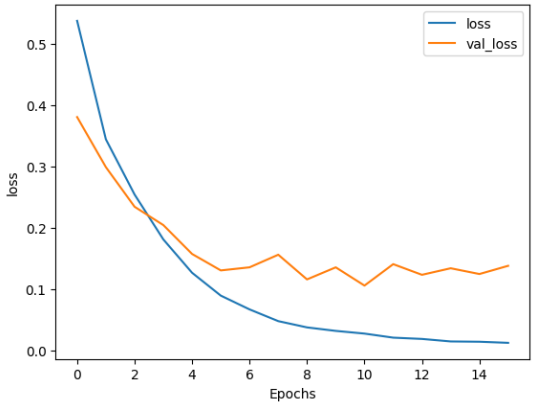

Figure 26: DILBERT Model loss over 16 Epochs

	precision	recall	f1-score	support
0	0.98	0.98	0.98	4466
1	0.98	0.97	0.97	3834
accuracy			0.98	8300
macro avg	0.98	0.98	0.98	8300
weighted avg	0.98	0.98	0.98	8300

Figure 27: Classification report for DILBERT Model

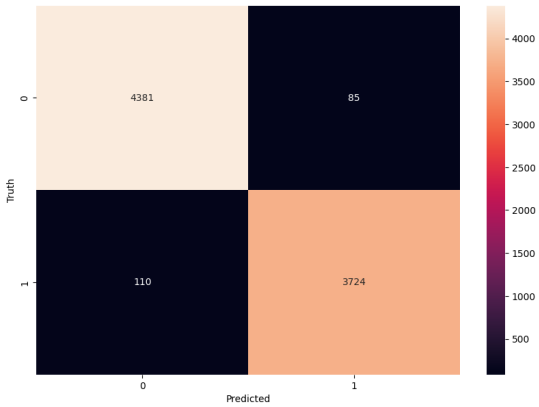

Figure 28: Confusion Matrix for DILBERT model

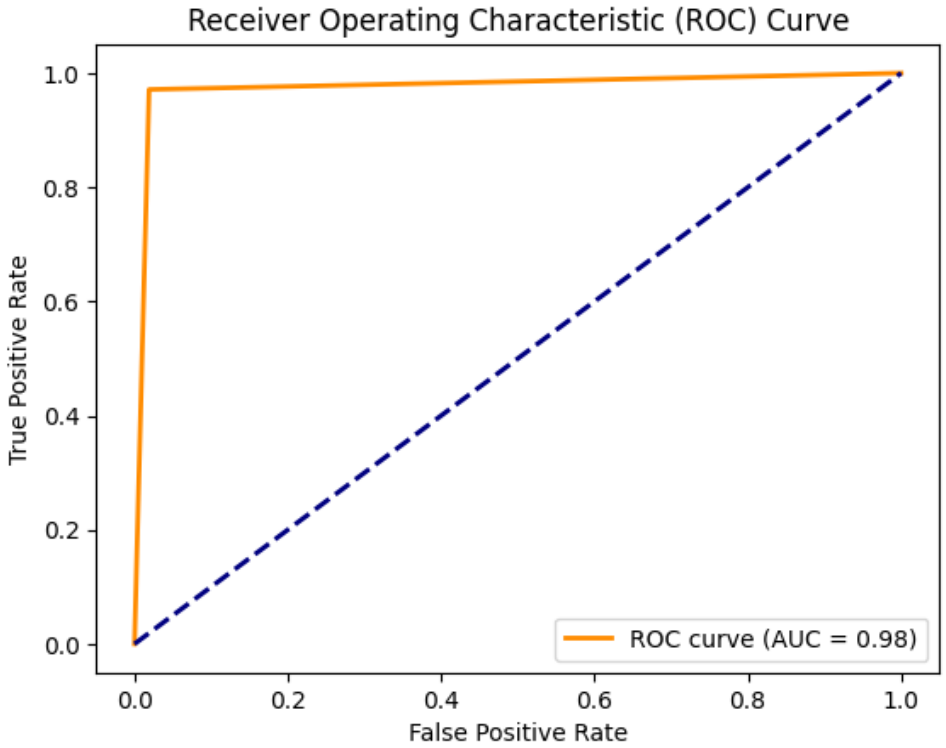

Figure 29: ROC Curve for DILBERT model

6.4 Discussion

In this research paper, we have discussed 3 models which were BERT, ALBERT, and DILBERT. For all of these models, we had taken similar model complexity to not create any biases, these models were fine-tuned to get the best results, and as well as we can understand the difference between the model in detail. All of the models outperformed and got an accuracy of 98 to 99% and a validation loss of less than 0.1. Detailed observation is given in Table 1 But to avoid any overfitting we used early stopping for these models with patience= 5 epochs, which indeed helped us to model to stop when the validation loss was fluctuating and was not decreasing gradually. From all the above models we can see that DILBERT was the fastest running model with approx 90-100s for each epoch, as it was the fastest but when the models were tested on the Y data set it took around 10-11 epochs to reach an accuracy of 98.57% and 98.27% on BERT and Albert while DILBERT reached 97.23% on 11 epochs but overall it took 16 epochs to reach 97.65% accuracy overall. This concludes that DILBERT can be used as one of the optimized solutions for sarcasm detection as compared to BERT as it takes less time and fewer parameters to reach a higher accuracy.

7 Conclusion and Future Work

This research introduced and executed a sarcasm detection approach by integrating the BERT machine learning algorithm and subsequently comparing its performance against DILBERT and ALBERT. The research looked at the Sarcasm news dataset to use it for testing and learning. Throughout the research, a comprehensive modeling process was meticulously followed, encompassing critical stages such as data preparation, feature engineering, model creation, and performance evaluation.

The data preparation phase encompassed essential tasks like data cleansing—removing stopwords, punctuation, and irrelevant text, along with tokenization and lemmatization. Subsequently, the research concentrated on fine-tuning and evaluating three distinct models: BERT, ALBERT, and DILBERT. Remarkably, this endeavor yielded significantly enhanced accuracy compared to prior research endeavors (Gosavi; 2022). The application of algorithms to the dataset was a major novelty of this research; especially, while BERT had previously been applied, the research introduced the first-time implementation of ALBERT and DILBERT on News Headline data.

Although the research was successful, certain model limitations were identified. Particularly, the ALBERT model exhibited fewer parameters yet consumed equivalent processing time as BERT. This discrepancy provides an opportunity for future advancement since improving ALBERT's runtime could considerably improve its practical utility in effectively evaluating customer perceptions for organizational advantage. In essence, this study not only exhibited higher precision, but it also opened the door to improving model efficiency and effectiveness, leading the way for more advanced sentiment analysis methodologies.

Model	Epochs	Precision	Accuracy	Average Running Time for 1 epoch(s)
BERT	10	0.9932	0.9856	183.8
ALBERT	11	0.9829	0.9826	180.36
DILBERT	11	0.9737	0.9723	96.18
DILBERT	16	0.9776	0.9765	95.18

Table 1: Performance Metrics of Different Models

References

Alharbi, A. S. M. and de Doncker, E. (2019). Twitter sentiment analysis with a deep neural network: An enhanced approach using user behavioral information, *Cognitive Systems Research* **54**: 50–61.

Aslam, F., Awan, T. M., Syed, J. H., Kashif, A. and Parveen, M. (2020). Sentiments and emotions evoked by news headlines of coronavirus disease (covid-19) outbreak, *Humanities and Social Sciences Communications* **7**(1).

Davidov, D., Tsur, O. and Rappoport, A. (2010). Semi-supervised recognition of sarcasm in twitter and amazon, *Proceedings of the fourteenth conference on computational natural language learning*, pp. 107–116.

Felbo, B., Mislove, A., Søgaard, A., Rahwan, I. and Lehmann, S. (2017). Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm, *arXiv preprint arXiv:1708.00524* .

Ghosh, A. and Veale, T. (2016). Fracking sarcasm using neural network, *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pp. 161–169.

González-Ibánez, R., Muresan, S. and Wacholder, N. (2011). Identifying sarcasm in twitter: a closer look, *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pp. 581–586.

Gosavi, S. R. (2022). *Transformer based Detection of Sarcasm and it's Sentiment in Textual Data*, PhD thesis, Dublin, National College of Ireland.

Kumar, A., Narapareddy, V. T., Srikanth, V. A., Malapati, A. and Neti, L. B. M. (2020). Sarcasm detection using multi-head attention based bidirectional lstm, *Ieee Access* **8**: 6388–6397.

Lee, H., Yu, Y. and Kim, G. (2020). Augmenting data for sarcasm detection with unlabeled conversation context, *arXiv preprint arXiv:2006.06259* .

Maynard, D. G. and Greenwood, M. A. (2014). Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis, *Lrec 2014 proceedings*, ELRA.

Misra, R. and Arora, P. (2019). Sarcasm detection using hybrid neural network, *arXiv preprint arXiv:1908.07414* .

Pan, H., Lin, Z., Fu, P., Qi, Y. and Wang, W. (2020). Modeling intra and inter-modality incongruity for multi-modal sarcasm detection, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1383–1392.

Potamias, R. A., Siolas, G. and Stafylopatis, A.-G. (2020). A transformer-based approach to irony and sarcasm detection, *Neural Computing and Applications* **32**: 17309–17320.

Rajadesingan, A., Zafarani, R. and Liu, H. (2015). Sarcasm detection on twitter: A behavioral modeling approach, *Proceedings of the eighth ACM international conference on web search and data mining*, pp. 97–106.

Riloff, E., Qadir, A., Surve, P., De Silva, L., Gilbert, N. and Huang, R. (2013). Sarcasm as contrast between a positive sentiment and negative situation, *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 704–714.

Shrivastava, M. and Kumar, S. (2021). A pragmatic and intelligent model for sarcasm detection in social media text, *Technology in Society* **64**: 101489.

Tsur, O., Davidov, D. and Rappoport, A. (2010). Icwsm—a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews, *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 4, pp. 162–169.