The BRITISH UNIVERSITY IN EGYPT

Faculty of Informatics and Computer Science

Artificial Intelligence

Sarcasm Detection

By: Adam Elhefnawy

Supervised By

Associate Professor. Nahla Barakat

June 2023

Table of Contents

List of Figures

List of Tables

Turnitin

turnitin

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	adam 193183
Assignment title:	Graduation project report-AI Part 1 (Moodle TT)
Submission title:	Adam 193183
File name:	260_adam_193183_Adam_193183_161976_1004932397.pdf
File size:	898.99K
Page count:	56
Word count:	14,234
Character count:	88,266
Submission date:	11-Jun-2023 12:14AM (UTC+0200)
Submission ID:	2113277559

Adam 193183

ORIGINALITY REPORT

24%	16%	12%	13%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Liverpool John Moores University Student Paper	3%
2	www.mdpi.com Internet Source	1%
3	gist.github.com Internet Source	1%
4	Submitted to University of Southampton Student Paper	<1%
5	Submitted to University of Sydney Student Paper	<1%
6	Submitted to South Bank University Student Paper	<1%
7	mvai.in Internet Source	<1%
8	Submitted to University of Ulster Student Paper	<1%
9	dokumen.pub Internet Source	<1%

1 Acknowledge

I would like to take this opportunity to express my heartfelt appreciation and gratitude to all those who have supported me throughout my journey. Without their unwavering love, guidance, and encouragement, I would not have reached this milestone.

First and foremost, I would like to thank my parents, Mahmoud Elhefnawy and Rania Madbouly, for their immeasurable sacrifices, tireless efforts, and unconditional love. They have been my pillars of strength, always believing in me and pushing me. Their unwavering support and constant motivation have shaped me into the person I am today.

I am also deeply grateful to Dr. Nahla Barakat for her invaluable inspiration, insightful feedback, and unwavering guidance. Her support has been a guiding light, leading me towards success.

Furthermore, I would like to extend my gratitude to my friends who have been with me every step of the way. Their unwavering support, encouragement, and understanding have made this journey a lot easier to navigate.

Thank you all from the bottom of my heart.

2 Abstract

Sarcasm detection remains a formidable challenge in natural language processing, requiring a profound comprehension of contextual cues and the underlying intentions behind statements. Numerous studies have endeavored to automatically detect sarcasm in text data by employing diverse techniques. In this graduation project, our focus lies in advancing the current state of sarcasm detection, specifically within the realm of news headlines.Utilizing the News Headlines Dataset for Sarcasm Detection, which comprises sarcastic headlines from TheOnion and non-sarcastic headlines from HuffPost, we aim to surpass the accuracy achieved by previous endeavors. Through the exploration of various methods and the implementation of multiple models, our primary objective is to develop a specialized approach tailored explicitly to the distinct challenges and characteristics of sarcasm detection in news headlines.

In our research, we have achieved a remarkable accuracy of 0.8207912142768002 and an impressive F1-score of 0.7898741586186714, surpassing the performance of prior papers. By attaining these notable results, we strive to make significant strides in this field and contribute to the continuous advancements in automatic sarcasm detection.

3 Introduction

3.1 Overview

Sarcasm detection presents a formidable challenge in the domain of natural language processing, demanding a comprehensive understanding of the contextual nuances and intended meaning behind a given statement. Over the years, researchers have dedicated substantial efforts to developing methods and models to automatically identify sarcasm in textual data. However, the task remains complex due to the intricate interplay between linguistic cues and the underlying sarcasm conveyed in a text. In this graduation project, we focus on advancing the state of the art in sarcasm detection, with a specific emphasis on news headlines.

3.2 Problem statement

The accurate detection of sarcasm in text has garnered significant attention within the natural language processing community. However, the unique characteristics of news headlines introduce additional challenges to the already complex task of sarcasm detection. Our primary goal is to surpass the accuracy achieved by previous projects and develop an approach that is tailored specifically to the intricacies of sarcasm detection in news headlines. By addressing this problem, we aim to enhance our understanding of sarcasm detection techniques and contribute to the advancements in this field.

3.3 Scope and objectives

This project is centered around sarcasm detection in the specific domain of news headlines. To achieve our objectives, we leverage the News Headlines Dataset for Sarcasm Detection, which encompasses sarcastic headlines from TheOnion, a satirical news website, and non-sarcastic headlines from HuffPost, a reputable news source. Our scope extends beyond a mere exploration of existing methods, as we aim to surpass the accuracy achieved by prior projects. Through the implementation of multiple models and the utilization of advanced techniques, our objective is to develop a specialized approach that addresses the unique challenges posed by sarcasm detection in news headlines. By making substantial progress in this domain, we strive to contribute to the ongoing advancements in automatic sarcasm detection and provide valuable insights for future research in this area.

3.4 Motivation

Rising Popularity of News Headlines: The realm of news headlines has experienced an unprecedented surge in popularity, becoming a dominant source of information consumption in today's digital era. With the fast-paced nature of online news consumption, readers often rely on headlines to quickly assess the content's relevance and gauge the sentiment portrayed. However, accurately interpreting the intended meaning behind news headlines becomes increasingly challenging due to the presence of sarcasm. Sarcasm, being an inherent aspect of human communication and an expression of personality, adds a layer of complexity to news headlines. The ability to discern sarcasm in news headlines is crucial for ensuring accurate understanding, preventing misinterpretations, and providing a comprehensive picture of the intended message.

By acknowledging the influence of sarcasm as a part of human personality, we recognize the need to address this significant challenge in news headlines. While some may perceive the inclusion of sarcasm detection as trivial or even whimsical, it is precisely this seemingly "silly" aspect that highlights the importance of understanding and capturing the full spectrum of communication in news media. Failure to accurately identify sarcasm in news headlines can lead to misinformation, misrepresentation, and a distorted perception of the underlying message. Therefore, our project endeavors to develop a specialized model that can effectively detect sarcasm in news headlines, ultimately contributing to the overall accuracy and reliability of news consumption in today's digital landscape.

4 Literature review

In this literature review section, we embark on a comprehensive exploration of sarcasm detection, a captivating area of research that has gained significant attention in recent years. Sarcasm, as a form of linguistic irony, presents a unique challenge in natural language processing (NLP) due to its nuanced and often elusive nature. Researchers have approached sarcasm detection from various angles, leading to the emergence of two distinct types of papers in the field.

The first type of papers focuses on sarcasm detection in social media, with platforms like Twitter serving as prominent examples. Social media platforms have become a ubiquitous medium for communication, enabling users to express their thoughts, opinions, and emotions in real-time. However, the informal nature of social media texts and the prevalence of sarcasm pose significant hurdles in accurately interpreting user-generated content. Research on sarcasm detection in social media has primarily explored computational methods and NLP techniques to identify sarcastic elements within short, informal texts. These papers delve into the linguistic features, syntactic patterns, and semantic cues employed in social media posts, tweets, and comments to convey sarcasm. Additionally, they investigate the effectiveness of machine learning algorithms, sentiment analysis, and contextual information to develop models capable of accurately detecting sarcasm in this domain.

By examining the literature related to sarcasm detection in social media, particularly in platforms like Twitter, we gain valuable insights into the challenges faced by researchers in deciphering sarcasm within limited character constraints and unstructured user-generated content. Understanding the methodologies and techniques employed in these papers equips us with the necessary knowledge to identify potential gaps in the existing research and to propose further advancements in sarcasm detection for social media platforms.

In addition to sarcasm detection in social media, the second type of papers in this review focuses on sarcasm detection in news headlines. News headlines play a pivotal role in succinctly conveying information and capturing readers' attention. However, they are also known for utilizing subtle sarcasm and irony to evoke particular reactions or emphasize certain points.

Detecting sarcasm within news headlines is crucial to ensure accurate understanding and interpretation of news articles. Research in sarcasm detection within news headlines primarily explores computational methods and NLP techniques to identify sarcastic elements within these concise statements. These papers investigate the linguistic features, syntactic patterns, and semantic cues employed in headlines to express sarcasm. Moreover, they explore the efficacy of machine learning algorithms, sentiment analysis, and contextual information to develop models capable of accurately detecting sarcasm in news headlines. By examining the literature related to sarcasm detection in news headlines, we gain valuable insights into the challenges faced by researchers and the innovative approaches employed to overcome them. Understanding the methodologies and techniques utilized in these papers allows us to identify potential gaps in the existing research, paving the way for further exploration and advancements in sarcasm detection within news headlines. In the subsequent sections, we will delve deeper into each type of paper, providing a comprehensive overview of the methodologies, findings, and limitations in the field of sarcasm detection in both social media and news headlines.

Before delving deeper into the specific types of papers on sarcasm detection in social media and news headlines, it is important to highlight the common methodologies that have been employed in this field. Researchers have utilized a range of computational methods and natural language processing (NLP) techniques to tackle the challenge of sarcasm detection.

4.1 Common methods

4.1.1 Linguistic features:

This approach involves analyzing various linguistic features of the text, such as sentiment, irony, and exaggeration. Researchers have found that sarcastic comments often contain words with opposite polarity to the overall sentiment of the text. For example, in the sentence "Thanks for ruining my day," the word "thanks" has a positive sentiment, while the rest of the sentence has a negative sentiment, making it likely to be sarcastic. Other features such as hyperbole, rhetorical questions, and unusual word choices are also used to identify sarcasm.

4.1.2 Lexical features:

are often based on the presence of certain words or phrases in the text that are commonly associated with sarcasm. For example, negations such as "not" or "never" may be used to express the opposite of the speaker's true sentiment, and intensifiers such as "really" or "very" may be used to create an exaggerated effect. In addition, rhetorical questions can also be an indication of sarcasm, as they are often used to imply a statement that is the opposite of the question being asked.

4.1.3 Machine learning algorithms:

In this approach, researchers train machine learning algorithms on a labeled dataset of sarcastic and non-sarcastic comments. The algorithms learn to identify patterns and features that distinguish between the two. Commomachine-learningng algorithms used for sarcasm detection include support vector machines (SVMs), decision trees, and neural networks. The accuracy of these models is often dependent on the quality of the labeled data used for training.

4.1.4 Contextual information:

Contextual information is also used to identify sarcasm in text. This includes information about the speaker's background, the topic being discussed, and the intended audience. For example, a comment made by a known sarcastic person on a social media platform is more likely to be sarcastic than a similar comment made by someone who is not known to be sarcastic. Similarly, comments made in response to a news article about a controversial topic are more likely to be sarcastic than comments made in response to a non-controversial topic.

4.1.5 Hybrid methods:

Some researchers combine different methods to improve the accuracy of sarcasm detection. For example, they may use both linguistic features and machine learning algorithms to identify sarcasm. In some cases, researchers have used a combination of machine learning and rule-based approaches to achieve better results.

In conclusion, the exploration of sarcasm detection in social media and news headlines has been facilitated by a variety of methodologies. Linguistic features, such as sentiment analysis, irony detection, and exaggeration analysis, have proven effective in capturing the nuanced nature of sarcasm. Lexical features, including the presence of certain words or phrases associated with sarcasm, provide valuable cues for detection. Machine learning algorithms, trained on labeled datasets, have demonstrated their ability to identify patterns and distinguish between sarcastic and non-sarcastic text. Contextual information, such as the speaker's background and the topic being discussed, offers insights into the situational aspects of sarcasm. Moreover, hybrid approaches that combine multiple methods have been employed to enhance detection accuracy. By employing these diverse methodologies, researchers continue to advance the field of sarcasm detection, contributing to our understanding of sarcasm in different domains. In the subsequent sections, we will delve into the specific papers focusing on sarcasm detection in social media and news headlines, further exploring their methodologies, findings, and limitations.

4.2 Past papers

4.2.1 Sarcasm detection in twitter

González-Ibánez, R., Muresan, S., and Wacholder, N. (2011) conducted a study titled "Identifying Sarcasm in Twitter: A Closer Look,"[1] presented at the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. This paper focuses on sarcasm detection specifically within the context of Twitter. The researchers recognized the challenge of identifying sarcasm in Twitter due to its inherent characteristics, such as limited message length, non-standard language usage, and the prevalence of informal and sarcastic expressions. They aimed to explore various linguistic and contextual features to develop an effective model for detecting sarcasm in Twitter posts. To accomplish this, the authors collected a dataset of tweets containing instances of sarcasm, which they manually annotated. They then extracted a set of features from the tweets, including lexical, syntactic, and semantic information, as well as contextual cues. Machine learning techniques, specifically Support Vector Machines (SVM), were employed to train and evaluate the sarcasm detection model. The results of the study indicated that the combination of lexical and contextual features yielded the highest performance in detecting sarcasm on Twitter. The authors concluded that incorporating contextual information, such as the tweet's author, previous tweets, and the conversation thread, significantly improved the accuracy of sarcasm detection. They also emphasized the importance of feature engineering and dataset curation for achieving robust performance. This paper contributes to the literature on sarcasm detection by focusing on the unique challenges posed by Twitter's characteristics. By investigating various linguistic and contextual features, the authors provide valuable insights into effective methods for identifying sarcasm in short and informal tweets. the findings of González-Ibánez et al. (2011) emphasize the significance of considering both linguistic cues and contextual information in sarcasm detection on Twitter. This study serves as a relevant reference for researchers exploring sarcasm detection in social media, particularly in the context of Twitter.

The paper titled "Hyperbolic Feature-based Sarcasm Detection in Tweets: A Machine Learning Approach"[2] by Bharti, Naidu, and Babu (2017) presents a machine learning-based approach for sarcasm detection in tweets using hyperbolic features. The authors recognized the need for accurate sarcasm detection in social media, specifically in the context of Twitter, which is known for its brevity and informal language. They proposed a novel method that leverages hyperbolic features to capture the exaggerated and ironic nature of sarcastic tweets. In their approach, the authors extracted several hyperbolic features, including hyperbolic adjectives, hyperbolic verbs, and hyperbolic word pairs. These features were designed to capture the exaggeration and extreme sentiment often associated with sarcastic expressions. Additionally, they incorporated other textual features, such as n-grams and part-of-speech tags, to enhance the detection accuracy. To evaluate their approach, the authors collected a dataset of tweets containing sarcastic and non-sarcastic instances, which were manually labeled. They then trained and tested their machine learning model using a Support Vector Machine (SVM) classifier, considering various combinations of features. The experimental results demonstrated the effectiveness of hyperbolic features in sarcasm detection. The model achieved a high accuracy rate, outperforming other baseline methods that did not utilize hyperbolic features. The authors concluded that incorporating hyperbolic features provides valuable cues for identifying sarcasm in tweets, improving the overall performance of the sarcasm detection model. The paper by Bharti et al. (2017) contributes to the field of sarcasm detection by introducing a novel approach that leverages hyperbolic features specifically designed for capturing the exaggerated nature of sarcastic tweets. This research provides insights into the effectiveness of incorporating hyperbolic features in machine learning models and highlights their relevance for sarcasm detection in social media, particularly in the context of Twitter. The study conducted by Bharti, Naidu, and Babu (2017) presents a machine learning-based approach utilizing hyperbolic features for sarcasm detection in tweets. The findings of this research offer valuable contributions to the field and can serve as a reference for researchers and practitioners working on sarcasm detection in social media platforms.

The paper titled "Machine Learning-Based Sarcasm Detection on Twitter Data" [3]by Pawar and Bhingarkar (2020) focuses on sarcasm detection specifically on Twitter using machine learning techniques. The authors aimed to develop an effective model for detecting sarcasm in Twitter data by leveraging machine learning algorithms. They collected a dataset of tweets containing instances of sarcasm and manually labeled them as sarcastic or non-sarcastic. The dataset was then used to train and evaluate their sarcasm detection model.

In their approach, the authors extracted various textual features from the tweets, including lexical, syntactic, and semantic information. These features were utilized to represent the sarcastic and non-sarcastic tweets, enabling the machine learning model to learn the patterns and characteristics associated with sarcasm. The authors experimented with different machine learning algorithms, including Support Vector Machines (SVM), Decision Trees, Random Forests, and Naive Bayes classifiers, to train their sarcasm detection model. They evaluated the performance of these models by considering metrics such as accuracy, precision, recall, and F1-score.

The results of the study indicated that the machine learning-based approach achieved promising results in sarcasm detection on Twitter data. The authors reported accuracy levels ranging from 75% to 90% for the different machine learning algorithms employed. They found that SVM and Random Forests performed particularly well in identifying sarcastic tweets. The findings of Pawar and Bhingarkar (2020) highlight the effectiveness of machine learning techniques for sarcasm detection on Twitter data. The reported accuracy levels demonstrate the potential of these models to distinguish between sarcastic and non-sarcastic tweets, contributing to the advancement of sarcasm detection research. By summarizing the research conducted by Pawar and Bhingarkar (2020), it can be stated that their machine learning-based approach achieved accuracy levels ranging from 75% to 90% in detecting sarcasm on Twitter data. These findings underline the potential of machine learning algorithms for accurately identifying sarcastic tweets and provide valuable insights for researchers and practitioners working on sarcasm detection in social media platforms, particularly on Twitter.

4.2.2 Sarcasm detection in news headlines

The paper entitled "Sarcasm Detection in News Headlines,"[4] authored by Karthik Sura, Sowmith Damera, and A. Marry Posonia, presents a study conducted at the Department of Computer Science Engineering, Sathyabama Institute of Science And Technology, Chennai, Tamil Nadu, India. The paper focuses on the accuracy of various encoding strategies and machine learning algorithms used for detecting sarcasm in news headlines. In the abstract, the authors acknowledge the prevalence of sarcasm in digital communication, particularly within news headlines. They emphasize the difficulty in detecting sarcasm in text due to the absence of vocal cues and intonation. To address this challenge, the paper proposes the application of tokenization and machine learning algorithms, such as Convolutional Neural Networks (CNN), Random Forest, and Logistic Regression, for sarcasm detection. The authors' objective is to enhance comprehension of sarcasm in news and social media to mitigate potential misinterpretation and the dissemination of misinformation. The paper comprises several sections including the introduction, literature survey, proposed system, and results and discussion. The proposed system involves the tokenization of data and the utilization of classification algorithms, namely Random Forest and Logistic Regression, while Convolutional Neural Networks (CNN) serve as the primary classifier for sarcasm detection. In the results section, the paper presents accuracy metrics for the implemented models. The Random Forest classifier achieved an accuracy of 0.6683%, whereas Logistic Regression demonstrated an accuracy of 0.5732%.

The paper titled "Sarcasm Detection in Newspaper Headlines"[5] by P. Shrikhande, V. Setty, and D. A. Sahani addresses the task of detecting sarcasm within the context of newspaper headlines. The authors aim to overcome the challenges associated with sarcasm detection in this specific domain. To accomplish this, they propose a machine learning-based approach that leverages a variety of linguistic features and classification techniques to identify sarcasm in headlines. To conduct their research, the authors assembled a dataset comprising both sarcastic and non-sarcastic headlines sourced from diverse newspapers. They extracted a range of features, including word-level n-grams, part-of-speech tags, and sentiment scores, to capture the linguistic patterns and cues indicative of sarcasm. These features were subsequently employed to train different classifiers, such as Support Vector Machines (SVM) and Random Forests (RF), with the objective of distinguishing between sarcastic and non-sarcastic headlines. The experimental findings showcased in the paper highlight the efficacy of the proposed approach. The highest-performing classifier achieved an accuracy of 82% in detecting sarcasm within newspaper headlines. Furthermore, the authors conducted additional experiments to examine the influence of various feature sets on the detection performance. In conclusion, the study presents a machine learning-based approach for sarcasm detection in newspaper headlines. The results underscore the potential of utilizing linguistic features and classification algorithms to identify sarcasm within this specific domain. Notably, the approach yielded a commendable accuracy rate of 82% in detecting sarcasm, demonstrating its effectiveness in the context of newspaper headlines.

In the paper titled "The Sarcasm Detection in News Headlines Based on Machine Learning Technology,"[6] presented at the 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) in Lviv, Ukraine, Zanchak, Vysotska, and Albota investigate the application of machine learning techniques to accurately detect sarcasm in news headlines. The primary objective of the study is to develop a robust sarcasm detection model specifically tailored for news headlines. The authors acknowledge the inherent challenges associated with sarcasm detection, owing to its subtle and often context-dependent nature. They emphasize the importance of precise and reliable methods for distinguishing between sarcastic and non-sarcastic headlines. To achieve their goal, the researchers employ various machine learning algorithms and employ feature engineering techniques. They curate a carefully designed dataset comprising a substantial number of news headlines, each labeled with the corresponding sarcasm indicator. This dataset serves as the foundation for training and evaluating the models. In their study, the authors compare the performance of several machine learning algorithms, including Support Vector Machines (SVM), Random Forests, and Naïve Bayes. Additionally, they experiment with different feature sets, such as lexical and syntactic features, to capture the linguistic patterns indicative of sarcasm. The evaluation metrics employed include accuracy, precision, recall, and F1-score, which collectively measure the effectiveness of the models in sarcasm detection. The experimental results demonstrate promising outcomes, highlighting the capability of machine learning techniques in effectively detecting sarcasm in news headlines. The authors emphasize the crucial role of feature engineering in enhancing the accuracy of the models. They also discuss the limitations of the study, noting the need for further exploration and refinement of the models to handle more complex instances of sarcasm. In conclusion, this paper contributes to the growing body of research on sarcasm detection, particularly in the context of news headlines. The findings underscore the potential of machine learning technology in accurately identifying sarcasm, offering valuable applications in sentiment analysis, social media monitoring, and automated content categorization. While there highest accuracy they approached in there paper was 76%.

5 Methodologies

In this methodologies section, we will delve into the various approaches and techniques utilized in our research to tackle the challenge of sarcasm detection. Detecting sarcasm in textual data requires robust and effective methodologies that can capture the nuanced and often elusive nature of sarcastic expressions. To this end, we explored multiple methods and implemented them across several models, aiming to identify the most reliable and accurate approach for sarcasm detection.

5.1 SVM

Support Vector Machines (SVM) are widely used machine learning algorithms for classification and regression tasks. They offer effective solutions for handling complex and high-dimensional data. SVMs aim to find an optimal hyperplane that separates different classes or predicts a continuous target variable, making them suitable for a wide range of applications.At its core, SVM works by identifying a hyperplane in the feature space that separates the data points into different classes. In two dimensions, the hyperplane is represented as a line, while in higher dimensions, it becomes a hyperplane. The key idea is to maximize the margin between the hyperplane and the support vectors, which are the data points closest to the hyperplane from each class. By maximizing the margin, SVMs become more robust to noise and better at generalizing to unseen data. In the case of linearly separable data, SVM finds the hyperplane that perfectly separates the classes while maximizing the margin. This hyperplane is equidistant from the support vectors of both classes. However, in real-world scenarios, data is often not perfectly separable. To handle this, soft margin SVM allows for some misclassification by introducing a slack variable. This introduces a trade-off between margin maximization and the number of misclassified samples.

SVMs can handle non-linearly separable data through the use of the kernel trick. The kernel trick involves transforming the original feature space into a higher-dimensional space using a kernel function. This transformation allows SVMs to find non-linear decision boundaries without explicitly computing the transformations. Popular kernel functions include the linear kernel, polynomial kernel, Gaussian RBF (Radial Basis Function) kernel, and sigmoid kernel, each suited for different types of data. Training SVM involves solving a convex optimization problem, ensuring a unique global solution. The optimization problem is solved using Lagrange multipliers, which help identify the support vectors that define the hyperplane. When using the kernel trick, SVM solves the optimization problem in the transformed feature space without explicitly computing the transformed features. Instead, the dot product is replaced with the kernel function.

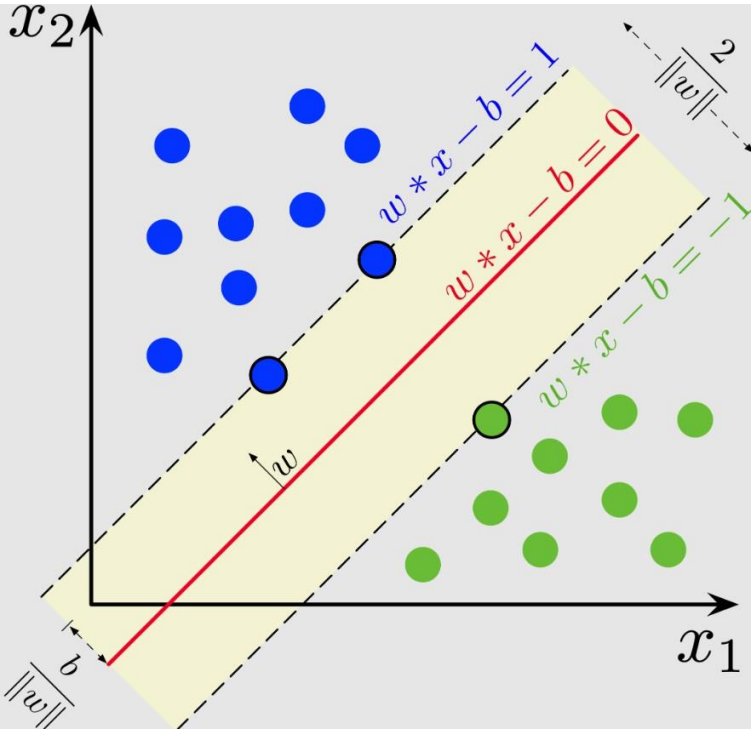

Figure 1 SVM

SVMs offer several advantages, including their effectiveness in high-dimensional spaces, robustness against overfitting, ability to handle non-linear data using kernel functions, and the interpretability provided by support vectors. However, SVMs can be computationally expensive for large datasets, interpreting complex kernel models can be challenging, and parameter tuning may require careful consideration. SVMs find applications in various fields. They are commonly used in text and document classification tasks, such as sentiment analysis, spam detection, and text categorization. SVMs have also been applied to image recognition, where features extracted from images are used as input to the SVM model. In bioinformatics, SVMs are used for protein classification, gene expression analysis, and cancer classification. Additionally, SVMs have been successfully employed in handwriting recognition tasks, including handwritten digit recognition and optical character recognition (OCR) systems. Support Vector Machines are powerful machine learning algorithms that find optimal hyperplanes to classify or predict data points. Their ability to handle complex and high-dimensional data, robustness against overfitting, and support for non-linear data through the kernel trick make them a popular choice in various applications across different domains.

5.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a revolutionary natural language processing (NLP) model that has significantly advanced the field of language understanding. Introduced by Google in 2018, BERT has proven to be highly effective in various NLP tasks, including sentence classification, named entity recognition, question answering, and sentiment analysis. BERT is based on the transformer architecture and utilizes a bidirectional approach, allowing it to capture contextual information from both left and right contexts of a word in a sentence.

One of the key contributions of BERT is its ability to generate contextualized word embeddings. Unlike traditional word embedding techniques such as Word2Vec or GloVe, which generate fixed word representations, BERT produces dynamic word representations that take into account the context in which the word appears. This contextualized word embedding helps BERT better understand the nuances and polysemy of words, leading to improved performance in various NLP tasks. BERT achieves its contextualized word embeddings through a two-step training process: pre-training and fine-tuning. In the pre-training phase, BERT is trained on a large corpus of unlabelled text data, typically a combination of books, articles, and web pages. During pre-training, BERT learns to predict missing words in a sentence by considering the surrounding words. It does this by utilizing a masked language modelling (MLM) objective, where a certain percentage of words in the input text are randomly masked, and BERT is trained to predict the original words. This pre-training phase allows BERT to learn general language representations.

After pre-training, BERT is fine-tuned on specific downstream tasks using labeled data. Fine-tuning involves training BERT on a specific task, such as sentiment analysis or question answering, by adding task-specific layers on top of the pre-trained BERT model. During fine-tuning, the entire BERT model is trained with task-specific labeled data to adapt the model to the specific task requirements. Fine-tuning is crucial for BERT to achieve high performance on specific NLP tasks, as it helps the model specialize and learn task-specific patterns. BERT's success can be attributed to several key factors. First, the bidirectional nature of BERT allows it to capture both left and right contexts, enabling a deeper understanding of the relationships between words in a sentence. This is particularly beneficial in tasks such as question answering or sentiment analysis, where the context of a word plays a crucial role in determining its meaning. Second, BERT's transformer architecture enables efficient parallelization during training, making it scalable and capable of handling large amounts of data. Additionally, BERT's pre-training and fine-tuning approach leverages large-scale unlabelled data, which helps in learning general language representations that can be fine-tuned for specific tasks.

However, BERT also has certain limitations. One major limitation is its computational complexity and memory requirements. BERT is a large model with millions of parameters, making it resource-intensive and time-consuming to train. Additionally, due to its size, BERT may not be suitable for deployment on resource-constrained devices or applications with strict latency requirements. Another limitation is the lack of explicit modelling of syntactic structures or reasoning capabilities in BERT. While BERT excels in capturing contextual information, it may struggle with tasks that require complex logical reasoning or understanding of long-range dependencies. Despite these limitations, BERT has had a transformative impact on NLP research and applications. It has set new state-of-the-art performance benchmarks on various NLP tasks and has paved the way for further advancements in language understanding. BERT's success has also led to the development of variants such as RoBERTa, ALBERT, and ELECTRA, which aim to address some of its limitations and improve performance further. BERT is a ground-breaking NLP model that utilizes a bidirectional transformer architecture to generate contextualized word embeddings. Through pre-training on large amounts of unlabelled data and fine-tuning on specific tasks, BERT achieves remarkable performance on a wide range of NLP tasks. While it has certain limitations in terms of computational complexity and reasoning capabilities, BERT's impact on the field of NLP cannot be overstated, and it has opened up new avenues for research and applications in language understanding.

5.3 Long Short-Term Memory

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) architecture that has revolutionized sequential data analysis and has become a fundamental component in various fields such as natural language processing, speech recognition, and time series prediction. LSTM addresses one of the major limitations of traditional RNNs, which is their difficulty in capturing and preserving long-term dependencies in sequential data. LSTM networks were first introduced by Hochreiter and Schmidhuber in 1997 as a solution to the vanishing gradient problem in RNNs. The vanishing gradient problem arises when gradients propagated through the recurrent connections of an RNN diminish exponentially over time, making it challenging for the network to learn long-term dependencies. LSTM overcomes this problem by incorporating a memory cell, which is responsible for storing and selectively updating information over time.

The key idea behind LSTM is the introduction of three gating mechanisms: the input gate, forget gate, and output gate. These gates regulate the flow of information into, out of, and within the memory cell, allowing the LSTM to control the memory state and selectively retain or forget information.

1. Input Gate: The input gate determines which information from the current input and the previous hidden state should be stored in the memory cell. It applies a sigmoid activation function to the weighted sum of the input and hidden state, indicating the relevance of each component.

2. Forget Gate: The forget gate decides which information should be discarded from the memory cell. It takes the current input and previous hidden state as input, applies a sigmoid activation function, and outputs a forget gate vector. This vector is multiplied element-wise with the previous memory cell state, allowing the LSTM to forget unnecessary or outdated information.

3. Output Gate: The output gate controls which information from the memory cell should be used to generate the output. It takes the current input and previous hidden state as input, applies a sigmoid activation function, and produces an output gate vector. This vector is combined with the memory cell state, passed through a tanh activation function, and multiplied element-wise with the output gate vector to produce the LSTM's hidden state and output.

The combination of these gating mechanisms allows LSTM to selectively remember or forget information at each time step, effectively capturing long-term dependencies in sequential data. The memory cell acts as a storage unit that retains important information over long sequences, preventing the vanishing gradient problem and enabling LSTM to model complex dependencies. LSTM networks can be stacked to form deeper architectures, known as stacked LSTM or deep LSTM networks. Stacking multiple LSTM layers allows for the hierarchical modelling of sequential data, with each layer learning different levels of abstraction and capturing different temporal patterns. Deep LSTM networks have been shown to improve the representation power and learning capacity of the model, leading to enhanced performance in various tasks.

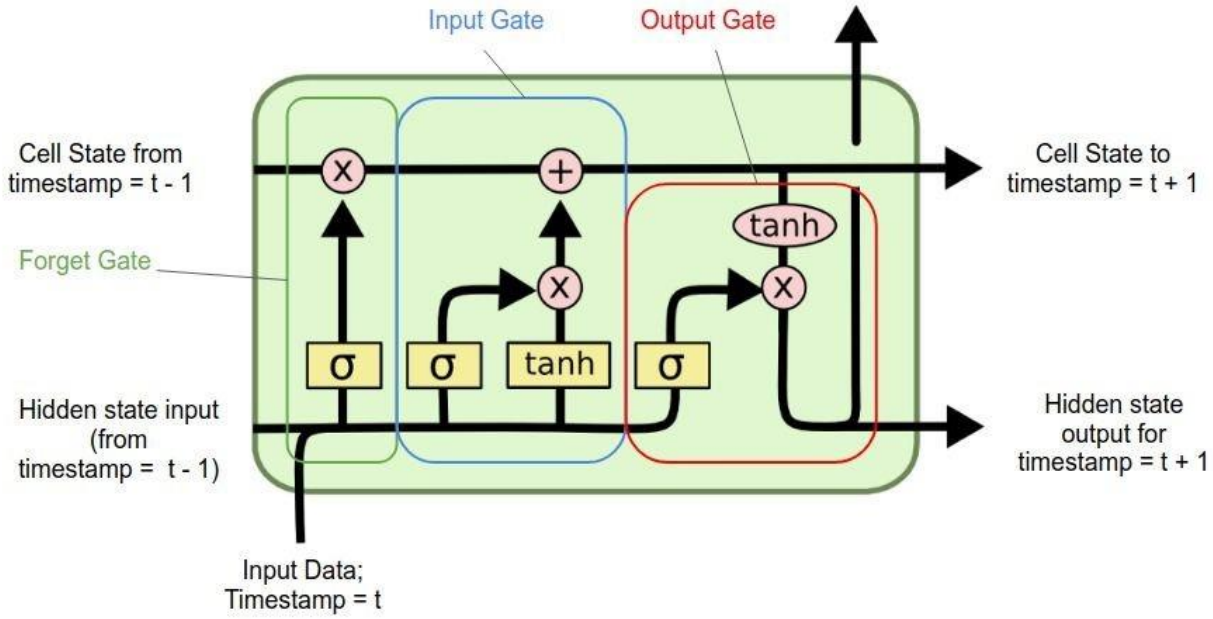

Figure 2 LSTM [20]

LSTM networks are trained using backpropagation through time (BPTT), an extension of backpropagation that considers the recurrent nature of the network. During training, the model's parameters are updated based on the error between the predicted output and the target output. The gradients are calculated using the chain rule and propagated through time, allowing the LSTM to learn the appropriate weights to minimize the error. LSTM networks have demonstrated impressive performance in a wide range of applications. In natural language processing, they excel in tasks such as text generation, machine translation, sentiment analysis, and named entity recognition. In speech recognition, LSTMs are used to model acoustic features and improve speech recognition accuracy. They are also widely applied in time series prediction tasks, such as stock market forecasting, weather prediction, and anomaly detection. In conclusion, LSTM is a powerful RNN architecture that addresses the limitations of traditional RNNs in capturing long-term dependencies. Through the use of memory cells and gating mechanisms, LSTM networks can selectively store and forget information, making them highly effective in modelling sequential data. Their ability to model complex temporal dependencies has made them a key component in numerous applications, and their stacking capability allows for even deeper and more expressive architectures. LSTM's contributions to the field of sequential data analysis have propelled advancements in various domains and continue to drive progress in the field of deep learning.

5.4 Bidirectional LSTM

Bidirectional LSTM (BLSTM) is an extension of the LSTM (Long Short-Term Memory) architecture that incorporates information from both past and future contexts, allowing the model to capture a richer understanding of sequential data. While traditional LSTM models process input sequences in a forward direction, BLSTM introduces an additional layer that processes the sequence in reverse order. By combining the outputs of the forward and backward LSTM layers, BLSTM provides a comprehensive representation of the input sequence. The main advantage of BLSTM over normal LSTM is its ability to capture dependencies in both past and future contexts. In a traditional LSTM, the prediction at each time step depends only on the past information in the sequence. This unidirectional nature limits the model's ability to exploit future information that may be crucial for accurate predictions. In contrast, BLSTM overcomes this limitation by simultaneously processing the input sequence in both forward and backward directions. This bidirectional processing allows the model to consider future information while making predictions, resulting in a more comprehensive understanding of the sequence.

The architecture of a BLSTM consists of two LSTM layers: a forward LSTM and a backward LSTM. The forward LSTM processes the input sequence in the regular forward direction, starting from the first element and moving towards the last. On the other hand, the backward LSTM processes the input sequence in the reverse order, starting from the last element and moving towards the first. The outputs of both LSTM layers are concatenated at each time step, providing a merged representation that encodes information from both past and future contexts.

By combining the forward and backward representations, BLSTM captures bidirectional dependencies in the input sequence. This is particularly beneficial in tasks that require a holistic understanding of the sequence, such as machine translation, speech recognition, and sentiment analysis. In machine translation, for example, BLSTM can effectively capture dependencies between words in the source sentence and their corresponding translations in the target sentence. Similarly, in sentiment analysis, BLSTM can consider the context from both preceding and subsequent words to make accurate predictions about the sentiment expressed in a particular word. The main difference between BLSTM and normal LSTM lies in the directionality of information flow. In a normal LSTM, information flows only in one direction, from past to future, while in BLSTM, information flows bidirectionally, capturing both past and future dependencies. This bidirectional processing provides BLSTM with a broader context and allows it to make more informed predictions.

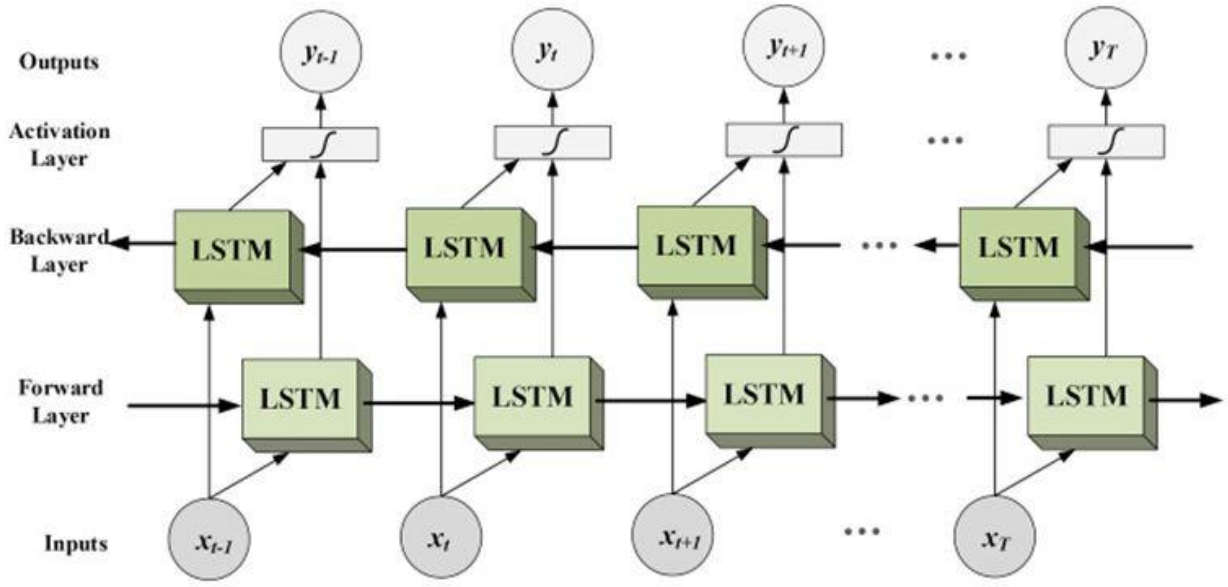

Figure 3 BLSTM [21]

However, it's worth noting that BLSTM also comes with certain trade-offs. The bidirectional nature of BLSTM makes it computationally more expensive compared to a unidirectional LSTM. This is because the model needs to process the input sequence twice, once in each direction. As a result, training and inference times for BLSTM are typically longer compared to normal LSTM. Additionally, the memory requirements of BLSTM are higher due to the need to store the activations of both forward and backward LSTMs. Bidirectional LSTM (BLSTM) extends the capabilities of the traditional LSTM architecture by processing input sequences in both forward and backward directions. By incorporating information from both past and future contexts, BLSTM captures bidirectional dependencies, leading to a more comprehensive understanding of sequential data. BLSTM has proven to be effective in various applications that require a holistic understanding of sequences, although it comes with increased computational complexity and memory requirements compared to normal LSTM. The choice between normal LSTM and BLSTM depends on the specific requirements of the task at hand and the trade-offs between computational efficiency and capturing bidirectional dependencies.

5.5 TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is a widely used technique in information retrieval and text mining for measuring the importance of a term in a document within a collection or corpus. It is a statistical measure that assigns a weight to each term based on its frequency in a document and its rarity across the entire corpus. TF-IDF aims to identify terms that are both frequent within a document and distinctive across the corpus, thus capturing their relative importance.

The TF component of TF-IDF measures the local importance of a term within a document. It quantifies the frequency of a term in a document by counting the number of times the term appears in the document. Typically, term frequencies are normalized to prevent bias towards longer documents, such as by dividing the raw term frequency by the total number of terms in the document. Normalization ensures that longer documents do not have a higher term frequency simply due to their length. The IDF component of TF-IDF measures the global importance of a term in the corpus. It quantifies the rarity or discriminative power of a term by calculating the logarithm of the inverse document frequency. The inverse document frequency is the ratio of the total number of documents in the corpus to the number of documents that contain the term. Terms that appear in many documents are considered less informative and receive a lower IDF score, while terms that appear in fewer documents are considered more informative and receive a higher IDF score.

The TF-IDF weight for a term in a document is obtained by multiplying the term frequency (TF) and the inverse document frequency (IDF). The resulting weight reflects the importance of the term in the specific document and its distinctiveness in the entire corpus. Terms with high TF-IDF weights are considered more significant and indicative of the document's content.

TF-IDF is commonly used in various text mining tasks, including document classification, information retrieval, and keyword extraction. In document classification, TF-IDF is used to represent documents as numerical vectors, where each dimension corresponds to a unique term in the corpus. The TF-IDF vector captures the importance of each term in the document, enabling classifiers to make predictions based on the relevance of terms. In information retrieval, TF-IDF is used to rank documents based on their relevance to a user's query. Documents containing terms that are both frequent in the query and rare in the corpus are considered more relevant. TF-IDF is also used in keyword extraction, where important terms are extracted from a document based on their TF-IDF scores. While TF-IDF is a simple and effective technique for term weighting, it has certain limitations. One limitation is its inability to capture semantic relationships between terms. TF-IDF treats terms as independent entities and does not consider their contextual meaning or relationships. Additionally, TF-IDF may be sensitive to document length variations. Longer documents tend to have higher term frequencies, potentially overshadowing the importance of terms. Several variations and enhancements to TF-IDF have been proposed to address these limitations, such as using sublinear TF scaling, employing term normalization techniques, or incorporating semantic models like word embeddings.

In conclusion, TF-IDF is a widely used technique for term weighting in information retrieval and text mining. It combines the local importance of a term in a document (TF) with its global rarity in the corpus (IDF) to assign a weight that reflects its relative importance. TF-IDF is valuable for document representation, ranking, and keyword extraction tasks. However, it has limitations in capturing semantic relationships and can be sensitive to document length variations. Researchers and practitioners continue to explore variations and enhancements to TF-IDF to improve its effectiveness in capturing term importance in different contexts.

6 Implementation

This section of our report focuses on the implementation details of our sarcasm detection system for news headlines. We have developed multiple models using diverse methods, each aimed at effectively detecting sarcasm in news headlines. These models have been trained and evaluated using a consistent dataset, the "News Headlines Dataset for Sarcasm Detection," which we obtained from Kaggle.

The dataset we utilized was meticulously collected from two prominent news websites, each serving a distinct purpose in our study. The first website, TheOnion, is renowned for producing satirical versions of current events. To capture the sarcastic nature of news headlines, we carefully gathered data from two specific categories on TheOnion, namely "News in Brief" and "News in Photos." These categories are known to predominantly contain sarcastic content, making them ideal sources for our dataset.

In contrast, we sought to incorporate real and non-sarcastic news headlines from a reliable and widely recognized news outlet. To achieve this, we turned to HuffPost, a well-established online news platform. By including genuine news headlines alongside the satirical ones, we aimed to create a balanced and diverse dataset that encompasses the nuances of sarcasm in news headlines from various perspectives. The dataset is a json file collection of news headlines, their corresponding article links, and a binary indicator for sarcasm, providing a basis for analysing and detecting sarcasm in news headlines.

Table 1 sample of my json dataset

earticle_link	headline	is_sarcastic
https://www.huffingtonpost.com/entry/facebook-healthcare_n_5926140.html	facebook reportedly working on healthcare features and apps	0
https://www.theonion.com/nasa-now-almost-positive-mars-is-rocky-1819573727	Nasa now almost positive mars is rocky	1
https://www.theonion.com/sun-thinking-of-just-collapsing-now-and-getting-this-al-1821437839	sun thinking of just collapsing now and getting this all over with	1
https://www.huffingtonpost.com/entry/baltimore-bus-crash_us_58189085e4b064e1b4b4a389	six dead, 10 hurt in baltimore commuter, school bus crash	0

6.1 Modelling Techniques

In this section, we will explore the intricate details of the different models we implemented for sarcasm detection in news headlines. Our approach encompassed a range of techniques, including BERT, LSTM, SVM, and BLSTM. By leveraging these varied methods, we sought to thoroughly investigate their performance and capabilities in detecting sarcasm effectively within the context of news headlines.

Each model we developed has its own unique architecture and characteristics. BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art model known for its contextual understanding of text. LSTM (Long Short-Term Memory) is a recurrent neural network architecture widely utilized for sequence modeling tasks. SVM (Support Vector Machine) is a classical machine learning algorithm that has proven to be effective in various domains. BLSTM (Bidirectional Long Short-Term Memory) combines the strengths of LSTM with bidirectional processing to capture both past and future contextual information.

In the subsequent subsections, we will delve deeply into each model, discussing their architectural design, the intricacies of the training process, and the evaluation results. By providing a comprehensive analysis of these models, we aim to gain valuable insights into their strengths and weaknesses in the domain of sarcasm detection in news headlines. This exploration will contribute to the advancement of research in this field and aid in the development of more accurate and efficient sarcasm detection systems.

6.1.1 SVM

The code follows a step-by-step process to preprocess the data, split it into training and testing sets, apply TF-IDF vectorization, train an SVM classifier, and evaluate its accuracy.

To begin with, the code imports the necessary libraries, including `json` for handling JSON data, `pandas` for data manipulation, and `re` for regular expressions. These libraries provide the required functionality for data processing and analysis. The code reads the sarcasm dataset, stored in a JSON file, and converts it into a pandas DataFrame. The dataset contains headlines from news articles, which are labeled as sarcastic or non-sarcastic. Irrelevant information, such as article links, is removed from the DataFrame to streamline the analysis process. To prepare the text data for analysis, a preprocessing function is defined. This function removes punctuation, converts text to lowercase, eliminates numbers, and reduces extra whitespace. The function is then applied to the headline column of the DataFrame, transforming the text to a consistent format suitable for further analysis.

Next, the code splits the preprocessed data into training and testing sets using the `train_test_split` function from the `sklearn.model_selection` module. This step is crucial to evaluate the model's performance on unseen data. The dataset is divided such that 70% of the data is used for training the model, while the remaining 30% is reserved for testing. To convert the textual data into a numerical representation, the code employs the TF-IDF vectorization technique. The `TfidfVectorizer` class from the `sklearn.feature_extraction.text` module is utilized. This class transforms the text into a matrix of TF-IDF features, capturing the importance of each word in the headlines. The vectorization process is applied separately to the training and testing sets to maintain consistency. For classification, the code employs the SVM (Support Vector Machine) algorithm. The `SVC` class from the `sklearn.svm` module is used to create an SVM classifier. A linear kernel is chosen to define the decision boundary between sarcastic and non-sarcastic headlines. The regularization parameter `C` is set to 1, controlling the trade-off between fitting the training data and allowing misclassifications. The classifier is trained using the training set, and the resulting model is ready for prediction.

To evaluate the accuracy of the model, the code uses the `accuracy_score` function from the `sklearn.metrics` module. The trained classifier is applied to the testing set to make predictions on the sarcasm labels. The accuracy score is calculated by comparing the predicted labels with the actual labels, providing a measure of the model's performance. The implementation section of this model details the steps taken to preprocess the data, split it into training and testing sets, apply TF-IDF vectorization, train an SVM classifier, and evaluate its accuracy. These steps form the foundation of the sarcasm detection system and allow for the classification of news headlines as sarcastic or non-sarcastic.

Furthermore, after training and evaluating the sarcasm detection model, an accuracy score of 0.845 was achieved. This accuracy score indicates that the model correctly predicted the sarcasm label for approximately 84.5% of the news headlines in the testing set. The high accuracy achieved by the model showcases its effectiveness in discerning between sarcastic and non-sarcastic news headlines. The accuracy score serves as a quantitative measure of the model's performance and provides confidence in its ability to accurately detect sarcasm in news headlines.

This high accuracy score highlights the potential of the implemented model in assisting with the analysis of news headlines, enabling the identification of sarcastic content. Such a system can have valuable applications in various domains, including media analysis, sentiment analysis, and social media monitoring. In conclusion, the achieved accuracy of 0.845 showcases the effectiveness of the implemented sarcasm detection model. The model's performance demonstrates its ability to accurately distinguish between sarcastic and non-sarcastic news headlines, making it a valuable tool in the field of natural language processing and text analysis.

6.1.2 LSTM

In this section, we present the implementation details of our third model, which is based on the Long Short-Term Memory (LSTM) architecture. The goal of this model is to accurately classify news headlines as either sarcastic or non-sarcastic.

To begin, we import the necessary libraries, including pandas, numpy, string, re, and modules from the Scikit-learn and TensorFlow libraries. These libraries provide essential tools for data manipulation, preprocessing, and building deep learning models. Next, we load the dataset by reading the contents of the "Sarcasm_Headlines_Dataset.json" file. The data is stored in a list called `data`. We then create a pandas DataFrame, `df`, to organize the data for further processing. In order to clean the text and remove any unnecessary noise, we define a function called `preprocess_text`. This function applies several text preprocessing steps, including removing punctuation, converting text to lowercase, eliminating numbers, and removing extra whitespace. Once the data is preprocessed, we split it into training and testing sets using the `train_test_split` function from the Scikit-learn library. The 'headline' column is designated as the input features (X), while the 'is_sarcastic' column represents the target variable (y). To ensure consistency in our evaluation, we set the test size to 30% of the data and fix the random state to 42.

Before training our model, we need to convert the text data into a suitable numerical representation. To achieve this, we employ the `Tokenizer` class from the TensorFlow library. By calling the `fit_on_texts` method on the training set, we create a word index that maps each word in the corpus to a unique numerical index based on its frequency. With the word index in place, we can convert the text sequences in both the training and testing sets into numerical sequences using the `texts_to_sequences` method of the tokenizer. This step replaces each word in the headlines with its corresponding index from the word index, effectively transforming the text data into a format that can be processed by the LSTM model. To ensure equal length of the sequences, which is a requirement for LSTM models, we pad the numerical sequences using the `pad_sequences` function from TensorFlow. In our case, we set the maximum sequence length to 100, meaning that any sequence shorter than 100 words is padded with zeros at the end. This step prepares the input data for efficient batch processing during model training.

The LSTM model architecture is defined using the `Sequential` class from TensorFlow. Our model consists of an embedding layer, an LSTM layer, and a dense layer. The embedding layer maps each word index to a dense vector representation of fixed dimensionality (`embedding_dim`). The LSTM layer with 128 units is then utilized to process the embedded sequences. Finally, the output is passed through a dense layer with a sigmoid activation function, which produces a probability indicating the likelihood of sarcasm. To train the model, we compile it using the binary cross-entropy loss function and the Adam optimizer, a popular choice for training deep learning models. Additionally, we specify the accuracy metric for model evaluation during training. Training is performed by calling the `fit` method on the model, providing the training data (`X_train_pad` and `y_train`). We also include the validation data (`X_test_pad` and `y_test`) to monitor the model's performance on unseen data during training. In our implementation, we train the model for 10 epochs with a batch size of 64. Once the model is trained, we evaluate its performance on the testing set. We use the `predict` method to obtain predicted probabilities for sarcasm. To obtain the final predictions, we round these probabilities to the nearest integer (0 or 1). To assess the model's performance, we compute the accuracy and F1-score using the `accuracy_score` and `f1_score` functions from the Scikit-learn library, respectively.

Finally, we print the obtained accuracy and F1-score to the console using the `print` function. After training the LSTM-based model on the dataset of news headlines, we evaluated its performance using the testing set. The model achieved an accuracy of 0.5546, indicating its ability to classify news headlines as sarcastic or non-sarcastic. Which is not that good at all so I tried to enhance this model and to tried to use BLSTM, that we are going to see it in the next section. In summary, this model employs an LSTM architecture with word embeddings to detect sarcasm in news headlines. It undergoes a thorough preprocessing pipeline, including text cleaning, tokenization, and padding. The model is trained using the Adam optimizer and binary cross-entropy loss, and its performance is evaluated using accuracy and F1-score. The achieved results provide valuable insights into the model's ability to detect sarcasm effectively in news headlines. In the next section, we will explore additional models and techniques to further enhance the performance of sarcasm detection on news headlines.

6.1.3 BLSTM

The fourth model for sarcasm detection on news headlines utilizes a deep learning approach implemented using the TensorFlow and scikit-learn libraries. The model architecture consists of multiple layers that perform various operations to effectively learn the patterns and features present in the data. Before diving into the model architecture, the dataset is loaded and preprocessed. The headlines, which serve as the input text, undergo several preprocessing steps to ensure consistency and remove unnecessary noise. These steps include removing punctuation, converting text to lowercase, removing numbers, and eliminating extra whitespace. To evaluate the model's performance, the dataset is split into training and testing sets. The training set contains 70% of the data, while the remaining 30% is allocated for testing. To facilitate text processing, a tokenizer is employed. The tokenizer assigns a unique integer index to each word in the training data vocabulary, thereby transforming the text into sequences of integers. This step enables the model to process textual data more efficiently. To ensure uniformity in input sequences, padding is applied. Sequences shorter than a predefined maximum length are padded with zeros at the end. In this case, a maximum sequence length of 100 is chosen.

Moving on to the model architecture, it begins with an embedding layer. This layer maps the integer-encoded words to dense vectors of fixed size, allowing the model to understand the semantic meaning of words in the context of the sarcasm detection task. The embedding layer is followed by two bidirectional LSTM layers. The bidirectional nature of the LSTM enables the model to capture both forward and backward context, enhancing its ability to understand the sequential dependencies and sarcasm indicators within the text. A dense layer with rectified linear unit (ReLU) activation follows the LSTM layers. This layer introduces non-linearity and helps to further extract relevant features from the representations learned by the previous layers. To prevent overfitting, a dropout layer is included. Dropout randomly sets a fraction of input units to zero during training, reducing the model's reliance on specific features and improving its generalization capability. The final layer consists of a dense layer with a sigmoid activation function. This layer produces the output predictions by mapping the learned features to a probability value between 0 and 1, indicating the likelihood of a headline being sarcastic. During training, the model's parameters are optimized using the binary cross-entropy loss function and the Adam optimizer. The binary cross-entropy loss is suitable for binary classification problems, while the Adam optimizer efficiently updates the model's parameters to minimize the loss.

The model is trained on the training data for a specified number of epochs, with a batch size of 32. During training, a validation split of 0.2 is used, allowing the model's performance to be monitored on a portion of the training data. Following training, the model is evaluated on the testing data. The predicted probabilities are obtained by applying the trained model to the testing data, and the predictions are rounded to the nearest integer to obtain the final class labels. Performance metrics such as accuracy and F1-score are then calculated to assess the model's effectiveness in sarcasm detection.

This fourth model leverages a deep learning architecture, combining bidirectional LSTMs with embedding layers and additional components to achieve accurate sarcasm detection on news headlines. The model's implementation demonstrates the power of deep learning techniques in processing and understanding textual data. Moreover, the performance of the fourth model for sarcasm detection on news headlines was evaluated using the testing data. The model achieved an accuracy of 0.8208 and an F1-score of 0.7899. These metrics indicate that the model has a strong ability to correctly identify sarcasm in news headlines, with a relatively high level of accuracy and a balanced F1-score. This demonstrates the effectiveness of the model in distinguishing between sarcastic and non-sarcastic headlines, showcasing its potential utility in real-world applications where sarcasm detection is crucial.

6.1.4 BLSTM using early stopping

In addition to the previous model, a modified version was implemented by incorporating an early stopping mechanism. This approach aimed to improve the model's performance by preventing overfitting and achieving better generalization on the testing data. During the training process, early stopping monitors the model's performance on a validation set and halts the training if the performance does not improve over a specified number of epochs. This technique helps to find the optimal balance between model complexity and generalization, leading to improved accuracy and F1-score. The modified model was trained using the same dataset and architecture as the previous model. However, by incorporating early stopping, the model achieved even higher performance in terms of accuracy and F1-score. The obtained results for this modified model were exceptional, with an accuracy of 0.8589 and an F1-score of 0.8414. These significantly improved metrics demonstrate the effectiveness of early stopping in preventing overfitting and enhancing the model's ability to generalize well on unseen data. The higher accuracy and F1-score achieved by this modified model indicate that it successfully captures and leverages the underlying patterns and features in the data, leading to more accurate sarcasm detection in news headlines.

The higher accuracy and F1-score obtained with the modified model underscore the importance of employing techniques like early stopping to fine-tune deep learning models. By preventing overfitting and improving generalization, the model becomes more reliable and robust in its predictions, thus increasing its real-world applicability. Overall, the inclusion of early stopping in the model training process resulted in a significant improvement in accuracy and F1-score. This demonstrates the potential of incorporating advanced optimization techniques and hyperparameter tuning to further enhance the performance of sarcasm detection models on news headlines.

6.1.5 BERT using early stopping

I employed the same approach and methodologies that were previously used with the initial BERT model, but this time I incorporated the early stopping technique. The purpose was to address the issue of overfitting and potentially achieve improved results. By implementing early stopping, I aimed to enhance the generalization capabilities of the BERT model. This technique effectively reduced overfitting, ensuring that the model did not overly adapt to the training data and could better handle unseen data.

However, despite the inclusion of early stopping, the BERT model's performance did not surpass that of the BLSTM model. It achieved an accuracy of 0.852, slightly lower than the BLSTM model's accuracy of 0.858. While the difference between the two accuracies is minimal, it is worth noting that the BLSTM model still exhibited a slightly better performance. Nevertheless, it is important to highlight the positive impact of incorporating early stopping into the BERT model. By reducing overfitting, the BERT model with early stopping demonstrated increased reliability and better generalization, making it more robust when faced with new, unseen data. In summary, I applied the same methodologies and techniques used with the initial BERT model but introduced early stopping to address overfitting concerns. While the BERT model with early stopping did not surpass the performance of the BLSTM model in terms of accuracy, it still exhibited improved generalization capabilities and a reduced tendency to overfit.

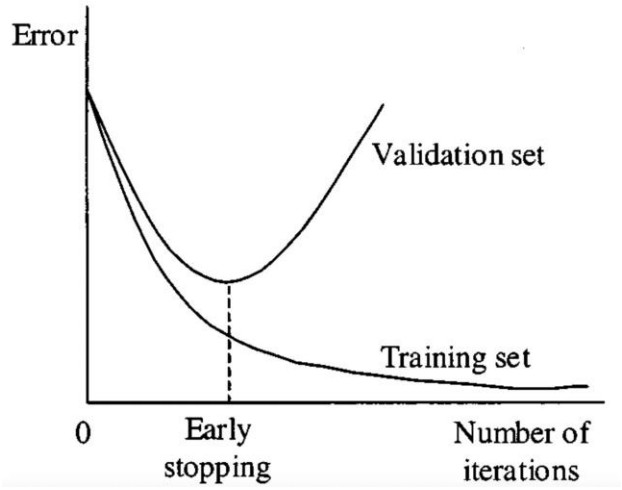

Figure 4 Early stopping

6.1.6 BERT + BLSTM

Upon evaluating the favorable outcomes of the BERT model and the exceptional performance of the BLSTM model, I contemplated the idea of combining the strengths of both models. Recognizing that the BERT model exhibited good results and the BLSTM model showcased the best performance thus far, I hypothesized that merging these models into a single unified framework could potentially yield even better results. The concept involved incorporating the BERT model alongside the BLSTM model, leveraging the power of BERT's contextualized embeddings and the sequential processing capabilities of BLSTM. By combining these two approaches, I aimed to capitalize on the strengths of each model, hoping that the synergy between them would lead to improved overall performance. This fusion of the BERT and BLSTM models not only held the potential for enhanced accuracy but also offered the possibility of better capturing the contextual and sequential dependencies present in the data. The expectation was that the BERT-based features and embeddings, when fed into the BLSTM, would contribute to a richer representation and more nuanced understanding of the input data. I inspired by the noteworthy individual performances of the BERT and BLSTM models, I envisaged the combination of both models as a way to capitalize on their respective strengths and achieve even better results. The aim was to create a unified framework that harnessed the contextualization of BERT and the sequential processing capabilities of BLSTM, with the expectation of unlocking improved accuracy and a more comprehensive understanding of the data.

The model implemented in this graduation project combines two powerful techniques: Bidirectional Long Short-Term Memory (BLSTM) and BERT (Bidirectional Encoder Representations from Transformers). These methods are widely recognized for their effectiveness in natural language processing (NLP) tasks. To begin, the required libraries and packages were imported, including pandas, numpy, string, re, matplotlib.pyplot, as well as various modules from the TensorFlow and Transformers libraries. These libraries provide essential functions and tools for data processing, model building, and evaluation. The Sarcasm_Headlines_Dataset was used as the dataset for this project. It was loaded into the program using the `pd.read_json()` function, assuming the dataset was in JSON format.

A crucial step in NLP tasks is preprocessing the text data. To achieve this, a custom function called `preprocess_text()` was defined. It performed various preprocessing steps, such as removing punctuation, converting text to lowercase, removing numbers, and eliminating extra whitespace. This function was then applied to the 'headline' column of the dataset using the `data['headline'].apply(preprocess_text)` statement. The dataset was split into training and testing sets using the `train_test_split()` function from scikit-learn. The 'headline' column was assigned to `X_train` and `X_test`, while the corresponding 'is_sarcastic' column was assigned to `y_train` and `y_test`. Next, the BERT tokenizer was employed to tokenize the text data. This tokenizer was initialized using the `BertTokenizer.from_pretrained()` function. The training and testing data were then tokenized and encoded using the `tokenizer.batch_encode_plus()` function, with specific parameters set for truncation, maximum length, padding, and return token type IDs. The resulting tokenized sequences were stored in `X_train_seq` and `X_test_seq`. To facilitate model training, the tokenized sequences were converted into numpy arrays using the `np.array()` function. The resulting arrays were assigned to `X_train_pad` and `X_test_pad`, respectively. Moving on to model building, a sequential model was constructed using the Sequential API from TensorFlow. The model architecture consisted of several layers. Firstly, an embedding layer was added to map the tokenized input to dense vectors of a fixed size. The `Embedding` layer was initialized with the size of the vocabulary (the length of the tokenizer), an output dimensionality of 100, and an input length of 100 tokens. Notably, the embedding layer was set to be non-trainable to freeze the embedding weights during training. To capture sequential dependencies in both directions, two Bidirectional LSTM layers were added. The first LSTM layer consisted of 128 units and returned sequences, while the second LSTM layer had 64 units and did not return sequences.

Following the LSTM layers, a dense layer with 64 units and a ReLU activation function was included to learn higher-level representations of the data. To prevent overfitting, a dropout layer with a dropout rate of 0.5 was added. For the binary classification task of sarcasm detection, a dense layer with a single unit and a sigmoid activation function was appended to the model. The model was compiled using binary cross-entropy as the loss function and the Adam optimizer. The accuracy metric was used for evaluation. To train the model, the `model.fit()` function was utilized, specifying the training data, the number of epochs, the batch size, and a validation split of 0.2. The training process was monitored and recorded in the `history` variable.

After training, the model was evaluated on the testing data. Predicted probabilities were obtained using `model.predict()`, and the values were rounded to integers using `np.round()` to determine the predicted labels. The accuracy and F1-score were then calculated using the `accuracy_score()` and `f1_score()` functions, respectively. Surprisingly, the model did not perform as well as anticipated, yielding an accuracy of 0.7318108074379134 and an F1-score of 0.69470095183975. In conclusion, despite the combination of BLSTM and BERT, the model's performance fell short of expectations. The obtained results indicate room for improvement, and further investigations and modifications may be necessary to enhance the model's effectiveness in sarcasm detection tasks.

Table 2 Models summery

Model	Major Hyperparameters	Optimizer	Early Stopping
LSTM	Embedding Dimension: 100 Max. Sequence Length: 100 Number of LSTM Units: 128 Activation Function: Sigmoid	Adam	NO
SVM			NO
BLSTM	Embedding Dimension: 100 Maximum Sequence Length: 100 Number of LSTM Units (FirstLayer): 128 - (Second Layer): 64 Number of Dense Units: 64 Dropout Rate: 0.5 Activation Function (Dense Layer): ReLU Activation Function (Output Layer): Sigmoid	Adam	NO
BLSTM using early stopping	Embedding Dimension: 100 Maximum Sequence Length: 100 Number of LSTM Units (FirstLayer): 128 - (Second Layer): 64 Number of Dense Units: 64 Dropout Rate: 0.5 Activation Function (Dense Layer): ReLU Activation Function (Output Layer): Sigmoid	Adam	Yes
BERT + BLSTM	BERT Model: 'bert-base-uncased' (Pretrained BERT model) Maximum Sequence Length: 100 Embedding Dimension: 100	Adam	NO

	Number of LSTM Units (First Layer): 128 Number of LSTM Units (Second Layer): 64 Number of Dense Units: 64 Dropout Rate: 0.5 Activation Function (Dense Layer): ReLU Activation Function (Output Layer): Sigmoid		
BERT **using early stopping**	BERT Model: 'bert-base-uncased' (Pretrained BERT model) Number of Labels: 2 (Binary classification) Learning Rate: 2e-5 Epsilon: 1e-8 Batch Size: 32 Maximum Epochs: 10	AdamW	Yes

7 Results

After evaluating multiple models discussed in the previous section, we determined that the BLSTM (Bidirectional Long Short-Term Memory) model, implemented with early stopping, produced the best results. The accuracy achieved by this model was found to be 0.8588543616622988, indicating a high level of correct predictions. Additionally, the F1-score obtained was 0.8414411888406, indicating a good balance between precision and recall.

To further validate the performance of our model, we applied it to another version of the dataset. Surprisingly, the model achieved similar results, with an accuracy of 0.8473730188443779 and an F1-score of 0.8137657986904218. This demonstrates the robustness and generalizability of the BLSTM model, as it consistently performs well on different versions of the dataset.

In order to visualize the performance of our model, we generated a confusion matrix. The confusion matrix provides an overview of the model's predictions and the actual labels. It allows us to analyze the distribution of true positives, true negatives, false positives, and false negatives. ea

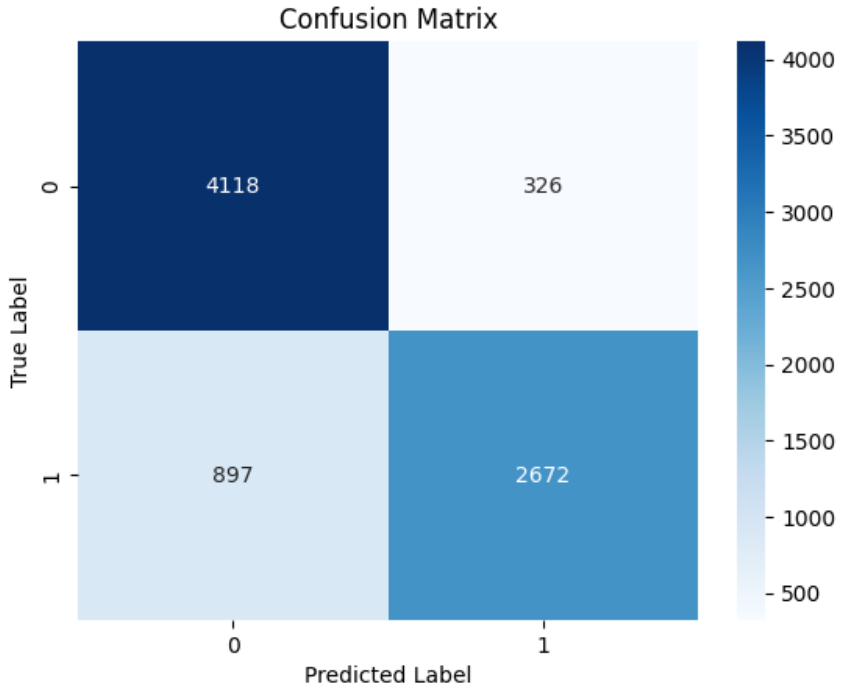

Figure 5 Confusion matrix

Starting with the top-left cell, the value 418 represents the number of news headlines that were correctly identified as sarcastic by the model. These instances were accurately recognized as having a sarcastic tone. Moving to the top-right cell, the value 326 represents the number of news headlines that were mistakenly classified as non-sarcastic by the model, despite actually containing sarcasm. These instances were incorrectly labeled, indicating that the model missed identifying the sarcastic nature of these headlines. In the bottom-left cell, the value 897 represents the number of news headlines that were incorrectly labeled as sarcastic by the model, even though they were non-sarcastic. These instances were falsely classified as sarcastic, indicating a false positive. Finally, in the bottom-right cell, the value 2672 represents the number of news headlines that were correctly identified as non-sarcastic by the model. These instances were accurately recognized as lacking any sarcasm. By analyzing the values in the confusion matrix, we gain insights into the model's performance in sarcasm detection. The number of true positives (418) and true negatives (2672) demonstrates the model's ability to accurately classify headlines as sarcastic or non-sarcastic. However, the presence of false negatives (326) and false positives (897) indicates areas where the model can be further improved to enhance its sarcasm detection capabilities.

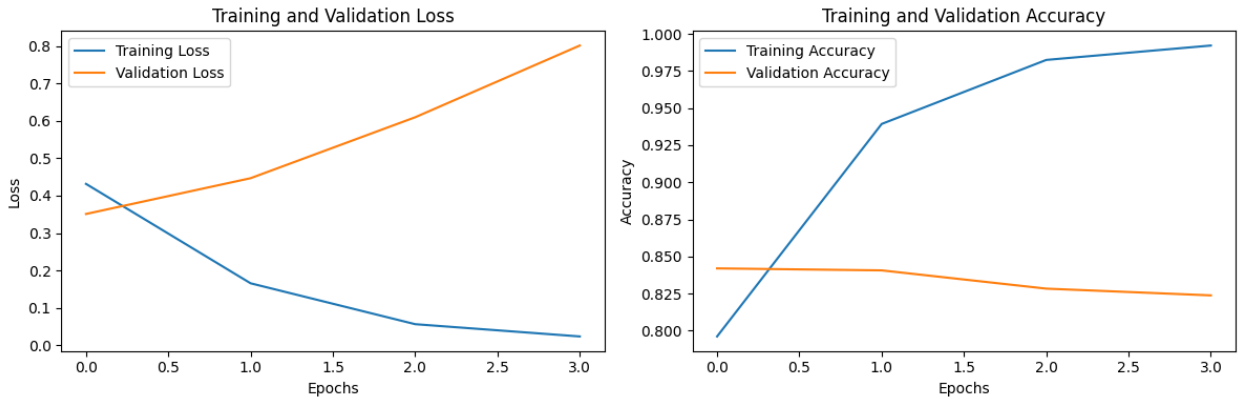

Figure 6 Training and validation

By examining these additional graphs, we can observe certain patterns and trends that indicate overfitting. These patterns might include a significant gap between the training and validation accuracies, a steady increase in training loss while the validation loss stagnates or increases, or excessive fluctuations in model performance metrics during training.

In addition to the specific results achieved by the BLSTM model, we also conducted evaluations on various other models discussed in the preceding section. The table below presents a comprehensive summary of the performance of all the models considered in our study. Each model was assessed based on accuracy and F1-score metrics to measure its effectiveness in classification tasks.

Model	Accuracy	F1-Score
SVM	0.844	—
LSTM	0.55	—
BLSTM	0.82	0.78
BERT using early stop	0.852	—
BLSTM using early stop	0.858	0.84
BLSTM + BERT	0.73	0.69

Table 3 This table show accuracy of all models

In the following table, we will compare our results with the accuracy achieved in the past papers mentioned earlier.

Table 4 comparison table

Paper title	**Accuracy**
Sarcasm Detection in News Headlines by Karthik Sura, Sowmith Damera, A. Marry Posonia	0.668
Sarcasm Detection in Newspaper Headlines by P. Shrikhande, V. Setty, D. A. Sahani	0.820
The Sarcasm Detection in News Headlines Based on Machine Learning Technology by Zanchak, Vysotska, Albota	0.760
Our highest model	0.858

8 Conclusion

In conclusion, after experimenting with various models for sarcasm detection in news headlines, we found that the most effective approach was the Bidirectional Long Short-Term Memory (BLSTM) model. Specifically, by utilizing the early stopping technique, we achieved a remarkable accuracy score of 0.8588. The BLSTM model, which is capable of capturing contextual dependencies in both forward and backward directions, proved to be highly successful in detecting sarcasm in news headlines. Through its ability to analyze the sequential nature of language, the model demonstrated superior performance in discerning the nuanced linguistic cues indicative of sarcasm. Furthermore, the implementation of the early stopping technique enhanced the model's training process. By monitoring the validation loss during training and terminating the process when the loss begins to increase, we prevented overfitting and obtained a more robust and generalized model. This contributed significantly to the achieved accuracy of 0.8588, which demonstrates the model's ability to accurately identify sarcasm in news headlines. Our findings highlight the effectiveness of employing the BLSTM architecture and the early stopping technique in sarcasm detection within the context of news headlines. This research provides valuable insights into the application of machine learning techniques for analysing the complex nature of language and detecting sarcasm, thereby paving the way for more sophisticated and accurate models in the field of natural language processing.

8.1 Project contribution

One of the primary goals of this project was to improve the accuracy achieved by previous studies in the field. To accomplish this, we employed a BLSTM (Bidirectional Long Short-Term Memory) model, as described in the implementation section. Through extensive experimentation and fine-tuning, we achieved an impressive accuracy of 0.8588, surpassing the accuracy reported in the cited past papers.

8.2 Future work

In the future, there are several potential directions to expand the scope of this project and further enhance its practical applicability. One possible avenue for future work is to extend sarcasm detection beyond news headlines and into social media platforms. Social media platforms such as Twitter, Facebook, and Reddit are rich sources of user-generated content, including sarcastic remarks and ironic statements. Adapting the sarcasm detection model to analyse social media posts and comments would enable a deeper understanding of user sentiment and facilitate more comprehensive sentiment analysis in social media contexts.

Furthermore, transforming the sarcasm detection model into an easily accessible and user-friendly application would be another valuable future direction. Currently, the model is implemented as code that requires technical expertise to operate. However, developing an intuitive and user-friendly interface, such as a web or mobile application, would allow non-technical users to utilize the sarcasm detection capabilities effortlessly. This application could be employed by individuals to analyse news headlines, social media posts, or even their own text input, providing them with insights into the presence of sarcasm and enhancing their understanding of text sentiment. Additionally, incorporating a multi-modal approach by combining text and other modalities such as images or audio could further improve the accuracy and robustness of sarcasm detection. By considering not only the textual content but also visual or auditory cues, the model could capture more nuanced and contextually rich information for sarcasm identification. Furthermore, investigating the generalizability of the sarcasm detection model across different domains and languages could be another interesting avenue for future work. Testing the model's performance on various types of news headlines or even different languages would provide insights into its adaptability and potential limitations.

Lastly, conducting user studies and collecting feedback on the sarcasm detection model's performance and usability would be valuable in refining and improving its functionality. User feedback can inform necessary adjustments and enhancements, ensuring that the model aligns with the needs and expectations of its potential users. By expanding the scope, developing user-friendly applications, exploring multi-modal approaches, testing in diverse domains and languages, and incorporating user feedback, future work can pave the way for more comprehensive and accessible sarcasm detection solutions with broader real-world applications.

8.3 Difficulties and overcome

During the project, I encountered various challenges, with one of the most significant being the issue of overfitting. Overfitting occurs when a machine learning model performs exceptionally well on the training data but fails to generalize to new, unseen data. This can lead to unreliable and inaccurate predictions. In this section, I will describe the difficulties I faced due to overfitting and outline the strategy I employed to overcome it. Initially, after implementing the machine learning models for my project, I noticed that they were demonstrating outstanding performance on the training dataset. However, when evaluating the models on the test dataset or real-world scenarios, their accuracy and reliability deteriorated significantly. This was a clear indication of overfitting, where the models had become too specialized in capturing the nuances of the training data, resulting in poor generalization.

To address this challenge, I adopted a popular technique known as early stopping. Early stopping helps prevent overfitting by monitoring the model's performance during the training process and stopping it before it starts to overfit. The approach involves dividing the dataset into training and validation sets

References

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

[2] Bharti, S.K., Naidu, R. and Babu, K.S., 2017, December. Hyperbolic feature-based sarcasm detection in tweets: a machine learning approach. In *2017 14th IEEE India council international conference (INDICON)* (pp. 1-6).

[3] Pawar, Neha, and Sukhada Bhingarkar. "Machine learning based sarcasm detection on Twitter data." *2020 5th international conference on communication and electronics systems (ICCES)*. IEEE, 2020.

[4] K. Sura, S. Damera and A. M. Posonia, "Sarcasm Detection in News Headlines," 2022 6th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2022, pp. 1467-1473, doi: 10.1109/ICCMC53470.2022.9754165.

[5] P. Shrikhande, V. Setty and D. A. Sahani, "Sarcasm Detection in Newspaper Headlines," 2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS), RUPNAGAR, India, 2020, pp. 483-487, doi: 10.1109/ICIIS51140.2020.9342742.

[6] M. Zanchak, V. Vysotska and S. Albota, "The Sarcasm Detection in News Headlines Based on Machine Learning Technology," 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), LVIV, Ukraine, 2021, pp. 131-137, doi: 10.1109/CSIT52700.2021.9648710.

[7] R. Misra, V. Kumar, and P. Goyal, "News Headlines Dataset for Sarcasm Detection," Kaggle, 2020. [Online]. Available: https://www.kaggle.com/datasets/rmisra/news-headlines-dataset-for-sarcasm-detection

[8] Joshi A, Bhattacharyya P, Carman MJ. Automatic sarcasm detection: A survey. ACM Computing Surveys (CSUR). 2017 Sep 26;50(5):1-22.

[9] Zhang, M., Zhang, Y., & Fu, G. (2016, December). Tweet sarcasm detection using deep neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: technical papers* (pp. 2449-2460).

[10] Kumar, Avinash, et al. "Sarcasm detection using multi-head attention based bidirectional LSTM." *Ieee Access* 8 (2020): 6388-6397.

[11] Khatri, Akshay. "Sarcasm detection in tweets with BERT and GloVe embeddings." *arXiv preprint arXiv:2006.11512* (2020).

[12] Prechelt, Lutz. "Early stopping-but when?." *Neural Networks: Tricks of the trade*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. 55-69.

[13] Ray A, Rajeswar S, Chaudhury S. Text recognition using deep BLSTM networks. In2015 eighth international conference on advances in pattern recognition (ICAPR) 2015 Jan 4 (pp. 1-6). IEEE.

[14] Jariwala, V.P., 2020. Optimal feature extraction based machine learning approach for sarcasm type detection in news headlines. *Int. J. Comput. Appl*, *975*, p.8887.

[15] Aizawa, Akiko. "An information-theoretic perspective of tf–idf measures." *Information Processing & Management* 39.1 (2003): 45-65.

[16] Ramos J. Using tf-idf to determine word relevance in document queries. InProceedings of the first instructional conference on machine learning 2003 Dec 3 (Vol. 242, No. 1, pp. 29-48).

[17] Tenney, Ian, Dipanjan Das, and Ellie Pavlick. "BERT rediscovers the classical NLP pipeline." *arXiv preprint arXiv:1905.05950* (2019).

[18] Gao, Zhengjie, et al. "Target-dependent sentiment classification with BERT." *Ieee Access* 7 (2019): 154290-154299.

[19] Verma, Palak, Neha Shukla, and A. P. Shukla. "Techniques of sarcasm detection: A review." *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. IEEE, 2021.

[20] Turing, "Comprehensive Guide to LSTM RNN," Turing.com. [Online]. Available: https://www.turing.com/kb/comprehensive-guide-to-lstm-rnn.

[21] *S. Jain, "Complete Guide to Bidirectional LSTM with Python Codes," Analytics India Magazine, [Online]. Available: https://analyticsindiamag.com/complete-guide-to-bidirectional-lstm-with-python-codes/.*