

INSTITUTO POLITÉCNICO NACIONAL

---

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

TESIS

“Automatic Sarcasm Detection”

QUE PARA OBTENER EL GRADO DE:

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

Oxana Vitman

DIRECTORES DE TESIS:

Dr. Alexander Gelbukh

Dr. Miguel Jesús Torres-Ruiz



Ciudad de México

Enero 2023



# INSTITUTO POLITÉCNICO NACIONAL SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

## ACTA DE REGISTRO DE TEMA DE TESIS Y DESIGNACIÓN DE DIRECTOR DE TESIS

Ciudad de México, a  de  del

El Colegio de Profesores de Posgrado del  en su Sesión

(Unidad Académica)

No  celebrada el día  del mes  de , conoció la solicitud presentada por el (la) alumno (a):

Apellido Paterno:	VITMAN	Apellido Materno:		Nombre (s):	OXANA
-------------------	--------	-------------------	--	-------------	-------

Número de registro:

del Programa Académico de Posgrado:

Referente al registro de su tema de tesis; acordando lo siguiente:

1.- Se designa al aspirante el tema de tesis titulado:

Objetivo general del trabajo de tesis:



2.- Se designa como Directores de Tesis a los profesores:


Director:  2° Director:   
No aplica:

3.- El Trabajo de investigación base para el desarrollo de la tesis será elaborado por el alumno en:

que cuenta con los recursos e infraestructura necesarios.

4.- El interesado deberá asistir a los seminarios desarrollados en el área de adscripción del trabajo desde la fecha en que se suscribe la presente, hasta la aprobación de la versión completa de la tesis por parte de la Comisión Revisora correspondiente.

Director(a) de Tesis  
  
 Dr. Alexander Gelbukh  
 Aspirante  
  
 C. Oxana Vitman

2° Director de Tesis  
  
 Dr. Miguel Jesús Torres Ruiz  
 Presidente del Colegio  
  
 Dr. Francisco Hiram Calvo Castro  
 DIRECCIÓN  
 IPN-CIC



ESTADOS UNIDOS MEXICANOS  
 INSTITUTO POLITÉCNICO NACIONAL  
 SECRETARÍA DE INVESTIGACIÓN  
 EN COMPUTACIÓN  
 DIRECCIÓN  
 IPN-CIC



# INSTITUTO POLITÉCNICO NACIONAL SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

## ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 11:00 horas del día 15 del mes de diciembre del 2022 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Posgrado de: Centro de Investigación en Computación para examinar la tesis titulada: "Automatic Sarcasm Detection" del (la) alumno (a):

Apellido Paterno:	VITMAN	Apellido Materno:	-	Nombre (s):	OXANA
-------------------	--------	-------------------	---	-------------	-------

Número de registro: A 2 1 0 5 5 1  
Aspirante del Programa Académico de Posgrado: Maestría en Ciencias de la Computación

Una vez que se realizó un análisis de similitud de texto, utilizando el software antiplagio, se encontró que el trabajo de tesis tiene 19 % de similitud. **Se adjunta reporte de software utilizado.**

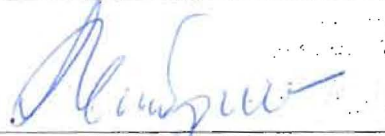

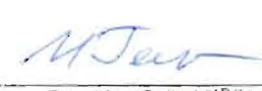




Después que esta Comisión revisó exhaustivamente el contenido, estructura, intención y ubicación de los textos de la tesis identificados como coincidentes con otros documentos, concluyó que en el presente trabajo SI  NO  **SE CONSTITUYE UN POSIBLE PLAGIO.**

**JUSTIFICACIÓN DE LA CONCLUSIÓN:** *(Por ejemplo, el % de similitud se localiza en metodologías adecuadamente referidas a fuente original)*  
Coinciden fragmentos pequeños de textos

**\*\*Es responsabilidad del alumno como autor de la tesis la verificación antiplagio, y del Director o Directores de tesis el análisis del % de similitud para establecer el riesgo o la existencia de un posible plagio.**

Finalmente y posterior a la lectura, revisión individual, así como el análisis e intercambio de opiniones, los miembros de la Comisión manifestaron **APROBAR**  **SUSPENDER**  **NO APROBAR**  la tesis por **UNANIMIDAD**  o **MAYORÍA**  en virtud de los motivos siguientes:  
Cumple con los requisitos para una tesis de maestría.

### COMISIÓN REVISORA DE TESIS

 Dr. Alexander Gelbukh Director de Tesis	 Dr. Grigori Sidorov	 Dra. Irina Gelbukh
 Dr. Miguel Jesús Torres Ruíz 2° Director de Tesis	 Dr. Ildar Batyrshin	 Dra. GuoHua Sun
		 Dr. Francisco Hiram Calvo Castro PRESIDENTE DEL COLEGIO DE PROFESORES



# INSTITUTO POLITÉCNICO NACIONAL

## SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

### *CARTA DE AUTORIZACIÓN DE USO DE OBRA PARA DIFUSIÓN*

En la Ciudad de México el día 5 del mes de enero del año, el (la) que suscribe Oxana Vitman alumno(a) del programa Maestría en Ciencias de la Computación con número de registro A210551, adscrito(a) a Centro de Investigación en Computación manifiesta que es autor(a) intelectual del presente trabajo de tesis bajo la dirección de Dr. Alexander Gelbukh, Dr. Miguel Jesús Torres-Ruiz y cede los derechos del trabajo intitulado "Automatic Sarcasm Detection", al Instituto Politécnico Nacional, para su difusión con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expresado del autor y/o director(es). Este puede ser obtenido escribiendo a las siguiente(s) dirección(es) de correo [ovitman2021@cic.ipn.mx](mailto:ovitman2021@cic.ipn.mx). Si el permiso se otorga, al usuario deberá dar agradecimiento correspondiente y citar la fuente de este.

Oxana Vitman

---

Nombre completo y firma autografiada del (de la)  
estudiante

# Contents

<b>Resumen</b>	<b>iv</b>
<b>Abstract</b>	<b>v</b>
<b>Acknowledgments</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Definition . . . . .	2
1.2 General Objective . . . . .	3
1.3 Particular Objectives . . . . .	3
<b>2 Theoretical Framework</b>	<b>4</b>
2.1 Types of Sarcasm . . . . .	4
2.2 Irony VS Sarcasm . . . . .	4
2.3 Sarcasm Detection Advantages and Limitations . . . . .	5
2.4 Problem Formulation . . . . .	6
<b>3 Related work</b>	<b>8</b>
3.1 Datasets . . . . .	8
3.1.1 Short Text . . . . .	8
3.1.2 Long Text . . . . .	9
3.1.3 Image . . . . .	10
3.1.4 Other Datasets . . . . .	10
3.2 Features . . . . .	11
3.2.1 Lexical . . . . .	11

3.2.2	Pragmatic . . . . .	12
3.2.3	Hyperbole . . . . .	12
3.2.4	Semantic . . . . .	13
3.2.5	Syntactic . . . . .	13
3.2.6	Sentiment and Emotion . . . . .	13
3.2.7	Context . . . . .	13
3.2.8	Image . . . . .	14
3.2.9	Feature Extraction Methods . . . . .	14
3.3	Methodologies . . . . .	16
3.3.1	Rule-based Approaches . . . . .	17
3.3.2	Classic Machine Learning-based Approaches . . . . .	18
3.3.3	Deep Learning-based Approaches . . . . .	20
3.3.4	Multi Task Learning . . . . .	23
3.3.5	Ensemble Learning . . . . .	23
3.3.6	Transformer based Approaches . . . . .	24
3.4	Trends . . . . .	26
3.4.1	Integrating Context . . . . .	27
3.4.2	Deep Learning . . . . .	29
3.4.3	Transformers . . . . .	30
<b>4</b>	<b>Experiments and Results</b>	<b>33</b>
4.1	Description of Datasets . . . . .	33
4.1.1	Reddit . . . . .	33
4.1.2	Internet Argument Corpus . . . . .	34
4.1.3	Twitter . . . . .	35
4.2	Proposed Method . . . . .	35
4.2.1	Overview of the Proposed Approach . . . . .	35
4.2.2	Sarcasm Pre-Trained Transformer . . . . .	38
4.2.3	Emotion Detection Pre-Trained Transformer . . . . .	38
4.2.4	Sentiment Detection Pre-Trained Transformer . . . . .	42

4.2.5	CNN . . . . .	42
4.3	Description of Baselines . . . . .	44
4.3.1	NBOW . . . . .	44
4.3.2	CNN . . . . .	45
4.3.3	CNN-LSTM-DNN . . . . .	45
4.3.4	SAWS . . . . .	49
4.3.5	ELMo . . . . .	54
4.4	Experiments . . . . .	58
<b>5</b>	<b>Conclusion and Future Work</b>	<b>59</b>
5.1	Results and Discussion . . . . .	59
5.2	Conclusion and Future Work . . . . .	61
5.3	Scientific Contribution . . . . .	62

# Resumen

Sarcasm detection plays an important role in Natural Language Processing as it has been considered one of the most challenging task in sentiment analysis and opinion mining applications. My work aims to recognize sarcasm in social media sites, microblogs and discussion forums, exploiting the potential of Deep Learning tools such as a combination of Transformers and Convolutional Neural Network. In this thesis, I (a) analyze multiple types of neural models and their efficiency when combined with word embeddings; (b) create a new multitasking framework that exploits the strong correlation between sarcasm, emotion and sentiment detection (c) test the performances of the model on four datasets; (d) compare results with other state-of-the-art models. I then discuss the benefits of research in the field of sarcasm detection and sentiment analysis and put the basis for some future research.



# Abstract

Sarcasm detection is an essential task that can help identify the actual sentiment in user-generated data, such as discussion forums or tweets. Sarcasm is a sophisticated form of linguistic expression because its surface meaning usually contradicts its inner, deeper meaning. Such incongruity is the essential component of sarcasm, however, it makes sarcasm detection quite a challenging task. In this work, I propose a model, that incorporates emotion and sentiment features to capture the incongruity intrinsic to sarcasm. I use CNN and pre-trained Transformer to capture context features. The proposed approach outperformed previous state-of-the-art results on four datasets from social networking platforms and online media.

# Acknowledgments

First and foremost, I would like to express my sincere gratitude and appreciation to my supervisors, Dr. Alexander Gelbukh and Dr. Miguel Jesús Torres-Ruiz, for their patience, guidance, and support. I am very grateful for their insightful comments and encouragement throughout the course of this work. My endless gratitude goes to Dr. Grigori Sidorov for inspiring me and being always ready to help both scientifically and personally. A final word of gratitude is dedicated to my colleague, Yevhen Kostiuk, my family, and all my friends for their constant support along the way. This work would not have been possible without the financial support of the Mexican Government, CONACYT project 240844, and the CONACYT scholarship.

# Chapter 1

## Introduction

The world's use of online communications is expanding quickly as social media has become the most important source of news and public opinion about almost every daily topic. The most popular application of social media analysis is detecting consumer sentiment to help businesses and online merchants to address the needs of their clients, including handling and resolving complaints.

However, not always emotions or sentiment expressed directly. Frequently social media users tend to make their posts or messages sarcastic in order to have better responses from other users and stimulate the virality of social media content. Moreover, negative sarcastic tweets draw in a substantially higher number of responses from users when compared to actual negative tweets Peng, Adikari, Alahakoon, and Gero (2019). Thus, sarcasm identification in online communications, discussion forums, and electronic commerce websites has become crucial for hate speech detection, sentiment analysis, and opinion mining. However, sarcasm detection appears to be a quite challenging task due to its sophisticated nature.

The surface meaning of sarcasm frequently contrasts with the underlying deeper meaning, making it a particularly complex kind of verbal expression. Although, this incongruity is the essential component of sarcasm, the intention may also be to appear humorous, make fun of someone, or show contempt. As a result, sarcasm is seen as an extremely sophisticated and intelligent language construct that presents a number of difficulties for the perception of emotions.

The following example of a tweet illustrates the above mentioned nuances: *“On our 6 am walk, my daughter asked me where the moon goes each morning. I let her know it’s in heaven, visiting daddy’s freedom”*.

On the surface, this statement seems to indicate that the speaker is enjoying his morning walk with his daughter and telling her amusing stories. However, a close examination of the speaker’s emotions and sentiments reveals that the speaker is unhappy and experiencing some unpleasant emotions at the time of speaking.

This is where incongruity between sentiment and emotions plays an important role. Sentiment, emotion, and sarcasm are highly interconnected, and one helps in the understanding of the others. We propose a model, which utilizes sentiment and emotion detection and learns their dependencies related to sarcasm. We hypothesized that learning a pattern of contradiction between surface sentiment and the intended sentiment is a key component in sarcasm detection.

## 1.1 Definition

Sarcasm is a seemingly positive expression of a negative statement. It differs from lying because the speaker is not intending to deceive. Its major objective is to convey criticism or critique hidden in humor (Nakassis & Snedeker, 2002). Tepperman, Traum, and Narayanan (2006) claims that sarcasm is the term used to describe communication that has a semantic interpretation completely different from its literal meaning. Sarcasm is described as a harsh, bitter, or cutting word or remark; a bitter gibe or insult by Poria, Cambria, Hazarika, and Vij (2016). Sarcasm, on the other hand, is a deeper idea that is closely tied to language and common knowledge (Bouazizi & Ohtsuki, 2016). A 6-tuple sarcasm representation was proposed by Ivanko and Pexman (2003). It has to include two persons: one speaking, and another listening. Also, it consists of, a context, a statement, an actual intent, and an hidden intent.

## 1.2 General Objective

The task of sarcasm detection is a binary classification task. It predicts whether a given text is sarcastic or not. Precisely, if sarcasm is detected in any part of a given text, the text is considered to be sarcastic.

## 1.3 Particular Objectives

Particular objectives of this work are:

- Explore the problem of Sarcasm Detection
- Explore related works
- Search and select available datasets for the research
- Design a novel architecture for the given task
- Implement proposed method
- Search and select SOTA results in sarcasm detection
- Analysis of the obtained results and their comparison with SOTA results

# Chapter 2

## Theoretical Framework

### 2.1 Types of Sarcasm

Many authors have attempted to categorize sarcasm into different types. Seven of them are listed by Abulaish and Kamal (2018): self-deprecating, contemplative, deadpan, courteous, obnoxious, manic, and angry. Sarcasm was classified into four categories by Sundararajan and Palanisamy (2020): polite, rude, deadpan, and raging.

A few studies have only addressed the detection of self-condemnatory sarcasm (Kamal & Abulaish, 2019). According to Oprea and Magdy (2019), there are two sorts of sarcasm: intentional and detected, and each should be treated as a different occurrence. Hence, two different dataset types were used in this research, one of which was manually labeled (detected), and the other one was annotated through remote guidance (intended). Due to the annotators' possible lack of comprehension of the authors' genuine intentions, the performance on the manually labeled dataset was unsatisfactory.

### 2.2 Irony VS Sarcasm

Numerous papers (Dimovska, Angelovska, Gjorgjevikj, & Madjarov, 2018; Ilić, Marrese-Taylor, Balazs, & Matsuo, 2018a; Naseem, Razzak, Eklund, & Musial, 2020; Potamias, Siolas, & Stafylopatis, 2020) claimed it is difficult for even human experts to tell the difference between sarcasm and irony. However, a few researchers tried to differentiate

between sarcasm and irony. There is a fine line between sarcasm and irony in literature. Irony is the term used to describe something that seems to be going against your expectations. An ironic result would be white, not off-white or gray, if an expectation is black. However, because sarcasm is generally directed at a specific person and seems like a witty parody, sarcasm is typically employed negatively.

Ling and Klinger (2016) sought to identify the latent structural distinction between sarcastic and ironic tweets. It was suggested by Khokhlova, Patti, and Rosso (2016) that sarcastic tweets may appear more positive than ironic tweets based on their usage of hashtags, tweet structure, the frequency of parts of speech, words frequency, and comparison with the NRC Word-Emotion Association Lexicon (EmoLEx) In SemEval-2018 the 12th workshop on semantic evaluation, a shared task on sarcasm detection was set up. The dataset included tweets gathered using hashtags related to irony (e.g., #irony, #sarcasm, #not). Utilizing a before-mentioned dataset, Dimovska et al. (2018) showed the effects of several features on irony and sarcasm recognition independently. Their top-performing irony detection model was a linear SVM employing the hashing vectorizer on the word n-grams.

## 2.3 Sarcasm Detection Advantages and Limitations

Sarcasm is frequently implemented on social networking and microblogging websites. On such platforms, users make fun of or criticize in a way that makes it challenging for even individuals to determine whether what is said was sarcastic or not. Sarcasm is frequently a barrier for sentiment analysis due to its metaphorical nature (B. Liu et al., 2010). Despite having a positive appearance, it conveys an inferred negative sentiment. The difficulties of sarcasm and the advantages of sarcasm detection to sentiment analysis have generated interest in the research problem of automatic sarcasm detection.

Automatic sarcasm detection is described as a computing method that assess whether a text is sarcastic. Sarcasm detection is an important task in a number of significant disciplines. For instance, sarcasm detection can be used in a culture-concerns subjects. Joshi, Bhattacharyya, Carman, Saraswati, and Shukla (2016) conducted research on the

factors affecting the accuracy of sarcastic predictions. According to the researcher, using his method would help in determining the value of brand new data sets.

Another work by Kannangara (2018) used sarcasm detection to categorize people's political viewpoints. The researchers put up three models for categorizing the social and political polarity of microblog articles. The researchers also proposed a new algorithm for sarcasm detection. It classifies sarcastic beliefs using ideology and fine-grained opinion as parameters, along with other linguistic features.

Sarcasm detection has a significant industry implementation in addition to the political application by utilizing the social media platform. These platforms grow into vast e-environments where users can freely express themselves. As a result, businesses use these landscapes to discover public opinion about elements of their products and services and to offer dedicated customer support (Sarsam, 2019). Additionally, companies also maintain active social media accounts and a responsive workforce for marketing and customer service (Rajadesingan, Zafarani, & Liu, 2015).

As a result, social media websites produce a vast amount of information that businesses can use as tools like HootSuite to handle a variety of challenging tasks, including data management, sentiment analysis, and the extraction of messages for the company's customer service representatives to respond to. Unfortunately, these techniques struggle with the intelligence needed to decipher complex languageassessestures like sarcasm that convey hidden meanings (Rajadesingan et al., 2015). Therefore, people's emotions can be easily understood through the detection of sarcastic statements.

## 2.4 Problem Formulation

In this chapter I would like to examine the concepts used in earlier research to define the challenge of automatic sarcasm detection. A classification task is the most typical design for sarcasm detection. The objective is to forecast the sarcasm presence in a given text. However, the nature of these output lables changes in earlier work. For instance, Barbieri, Saggion, and Ronzano (2014) consider the following labels for the classifier: politics, humor, irony, and sarcasm and focused on understanding of the relationship between sarcasm,



irony, and humor. Similar formulation is used by Reyes and Saldivar (2022), who report pair-wise classification performance for these labels.

There have also been suggestions of other sarcasm detecting formulas. In contrast to the conventional classification definition, Joshi, Goel, Bhattacharyya, and Carman (2016) model sarcasm detection for dialogue as a sequence labeling task. In this sequence, each statement in a discourse is regarded as an observable unit, whereas the hidden variables whose values must be predicted are the sarcasm labels. Sarcasm detection is modeled by D. Ghosh, Guo, and Muresan (2015) as a sense disambiguation job. According to them, a word can have both a literal and a satirical meaning. Their objective is to determine a word's sense in order to recognize sarcasm.

While there have been a number of intriguing conclusions, two stand out: (a) tweets are the most common texts utilized for sarcasm detection, and (b) the incorporation of extra-textual context is a recent development in sarcasm detection.

# Chapter 3

## Related work

### 3.1 Datasets

The datasets utilized in various sarcasm detection research are described in this section. The majority of these experiments test and train on widely used datasets that are already out there, such as those proposed by Cai, Cai, and Wan (2019); Khodak, Saunshi, and Vodrahalli (2018); Riloff et al. (2013). I found that, although Reddit, Amazon, and a few discussion forums were also used, Twitter is primarily the most popular social media network for sarcasm detection datasets. Short text, long text, and image data types are the three basic divisions that can be made of these datasets.

#### 3.1.1 Short Text

The the most common type of datasets used in the sarcasm detection studies are short texts. Because most social media platforms have a length restriction for posts and comments, the social media datasets from the studies I focused on are mostly short texts. Most of these brief texts are discovered to have been found on Twitter and Reddit. Due to its 280-character character limit, the social networking site Twitter is characterized as a microblogging platform.

Twitter is a great venue for gathering information about sarcasm and irony because of

its 330 million monthly active users<sup>1</sup>, who range in age from teenagers to seniors. It has been discovered that a lot of people utilize the Twitter API<sup>2</sup> to collect data. A Twitter dataset with 3,200 tweets proposed by Riloff et al. (2013), where 742 tweets were classified as sarcastic and 2,458 were classified as non-sarcastic.

The dataset was later used in numerous important papers on sarcasm detection (D. Ghosh et al., 2015; Joshi, Sharma, & Bhattacharyya, 2015; Tay, Tuan, Hui, & Su, 2018). Another such platform that permits slightly higher message sizes is Reddit. As Reddit still has a length restriction, and it can also be placed under the short text category (Joshi, Goel, et al., 2016), unlike most discussion forums that have no fixed length restrictions. Reddit has more than 430 million active monthly users<sup>3</sup>, the majority of whom are young people, making it an excellent source of information for sarcasm detection. The SARC (Self-Annotated Reddit Corpus)<sup>4</sup> dataset, which was proposed in the paper of Khodak et al. (2018) is one of the most well-known sarcasm detection datasets.

1.3 million sarcastic and 532 million non-sarcastic posts are included in the SARC dataset. Later, this dataset was utilized in additional studies, one of them was conducted by Hazarika et al. (2018). Aside from these, there are other other datasets with a focus on short sentences that were either built using brand new Twitter or Reddit data or collected as a subset of these two datasets. In papers for the SemEval-2018 (Semantic Evaluation 2018) shared task, such as in the works of Ilić, Marrese-Taylor, Balazs, and Matsuo (2018b) and Wu et al. (2018), the Twitter and Reddit dataset were used. Other short text data gathering methods from book excerpts, online comments, and other sources were also employed (Bharti, Vachha, Pradhan, Babu, & Jena, 2016; Joshi, Bhattacharyya, et al., 2016).

### 3.1.2 Long Text

The second most common type of datasets used in sarcasm detection research are long texts. The majority of these datasets include Amazon product reviews (Agrawal & An,

---

<sup>1</sup><https://financesonline.com/number-of-twitter-users/>

<sup>2</sup><https://developer.twitter.com/en/docs/twitter-api>

<sup>3</sup><https://earthweb.com/how-many-people-use-reddit/>

<sup>4</sup><https://nlp.cs.princeton.edu/SARC/>

2018; Dharwal, Choudhury, Mittal, & Kumar, 2017; Parde & Nielsen, 2018). With millions of products and hundreds of reviews, Amazon is the largest e-commerce marketplace. A dataset with 437 sarcastic and 817 neutral Amazon reviews was developed by Filatova (2012). Misra and Arora (2019) generated another dataset of this type. Although it also contained data in various formats and types. Long text data are also commonly found on discussion boards.

A dataset of 2,496 sarcastic and non-sarcastic comments from debate forums was developed by Oraby et al. [40]. It was noted that other types of data from other social media platforms are frequently utilized in addition to these discussion forum data. One such dataset, containing information from Twitter, books, discussion forums, and product evaluations and comments, was presented in the work of Bharti et al. (2016). Long-text sarcasm data can also be found in news portals, Facebook posts, and Yelp reviews. Subramanian, Sridharan, Shu, and Liu (2019) used data from both Facebook and Twitter. Due to the rise in popularity of other e-commerce or review websites and online portals outside Twitter and Reddit over the past few years, the use of datasets containing long texts is a relatively recent trend.

### 3.1.3 Image

The majority of the multimodal datasets on which I have concentrated were produced using tweets that included both texts and images. Cai et al. (2019) are one example. A multimodal dataset comprising 14075 sarcastic and 10560 non-sarcastic tweets, including photos, was implemented by them. Later, this dataset was used in numerous studies (Pan, Lin, Fu, & Wang, 2020; Wang, Wu, Wang, & Ren, 2015; Xu, Zeng, & Mao, 2020). By combining text and images from Instagram, Tumblr, and Twitter, Schifanella, De Juan, Tetreault, and Cao (2016) produced another well-known multimodal dataset.

### 3.1.4 Other Datasets

Other innovative datasets also have been used. Tepperman et al. (2006) employ 131 transcripts from call centers. Each time the phrase “yeah right” appears, it is noted

as sarcastic or not. The objective is to determine which “yeah right” is sarcastic. 20 sarcastic and 15 non-sarcastic samples from Kreuz and Caucci (2007) work are used, and 101 students grade them. Finding linguistic signs of sarcasm is the aim. The focus of Veale and Hao (2010) study is on determining whether similes are ironic. They start by looking for the pattern “as a” on the internet. As a result, 20,000 unique similes are produced, which are then classified as sarcastic or not. Sentences from the MTV program *Daria* were collected in a crowdsourced dataset by Rakov and Rosenberg (2013).

In a similar manner, Joshi, Jain, Bhattacharyya, and Carman (2016) present their dataset of the TV show “Friends” that has been manually annotated. Every “utterance” (sic) in a scene has two classifications next to it: sarcastic or not. D. Ghosh et al. (2015) get a non-sarcastic version of a statement using a crowdsourcing method. For instance, it is anticipated that “Who doesn’t love being ignored” be changed to “Not many adore being ignored”. Using quotes from the website [sarcasmsociety.com](http://sarcasmsociety.com), Misra and Arora (2019) create a manually labeled dataset of quotations.

## 3.2 Features

Before the emergence of deep learning-based models, sarcasm in social media communications was identified using hand-selected features extracted from texts. The features can be divided into a few key categories, which are concisely described in this section.

### 3.2.1 Lexical

The majority of features employed in social media sarcasm detection are lexical features. Characters, n-grams, phrases, integers, hashtags, and other feature types are examples of lexical features. N-grams, the majority of which are unigrams and bigrams, are frequently employed features in all categories of natural language processing research (Joshi, Goel, et al., 2016; Ling & Klinger, 2016; Schifanella et al., 2016). In addition to whole phrases, many studies also utilize character n-grams as features. Dimovska et al. (2018) uses four separate types of features, including character unigrams, character n-grams (where n was set between 1 and 4), word unigrams, and word n-grams (where n was set between 1 and

3). Additionally, some investigations have numerical components (Dubey, Joshi, & Bhattacharyya, 2019; A. Kumar, Narapareddy, Srikanth, Malapati, & Neti, 2020). Hashtags, since they can reveal sarcastic intents, have also been employed as features (A. Ghosh & Veale, 2016; Ilić et al., 2018b).

### 3.2.2 Pragmatic

In textual data, pragmatic elements typically take the form of expressions and reactions. For instance, smileys and emoticons, which are widely used to indicate emotions, can be used to distinguish between sarcastic and non-sarcastic statements. Emoticons and smileys were a part of the feature sets utilized by Bharti, Babu, and Jena (2015). Likewise, ratings and comments made in response to social media posts may also be sarcastic or ironic.

Thus, pragmatic features can be found in many social media sarcasm detection studies. Six different user reaction counts in relation to a Facebook post were one of the features used by Das and Clark (2018). The feature set of Parde et al. [37] included Amazon star ratings. In their experiment, Felbo, Mislove, Søggaard, Rahwan, and Lehmann (2017) combined pragmatic and lexical features. Onan (2019) included pragmatic aspects along with lexical, implicit incongruity, and explicit incongruity based feature sets in addition to feature sets based on word-embeddings.

### 3.2.3 Hyperbole

Hyperbole features can aid in understanding the links between words and other features as well as the relevance level of a sentence, like interjections, intensifiers, and punctuation, and they are also crucial in the detection of sarcasm.

The number of question marks, exclamation points, periods, capital letters, and “or” usages were the five punctuation-based features that A. Kumar et al. (2020) examined in their analysis of the tweets.

Capitalization, which can indicate stress on specific n-grams and is thus potentially significant, is another form of exaggeration trait. The feature set of Prasad, Sanjana, Bhat, and Harish (2017) contains a case study of capitalization being used as a feature.

### 3.2.4 Semantic

Semantic features include things like word length on average, word frequency, sequence length, etc. These are used to provide context for a statement. Semantic incongruity was a feature in the experiment by Chakrabarty, Ghosh, Muresan, and Peng (2020).

### 3.2.5 Syntactic

Among other things, Amir, Wallace, Lyu, and Silva (2016) experiments included syntactic features. The ability of Parts-of-Speech (POS) tags to denote the nature of words or tuples makes them particularly popular. Sarcasm was identified using Parts-of-Speech tags by A. Ghosh and Veale (2016). The extraction technique employed was called POS tagging, which involves breaking down a sentence into words or tuples and assigning tags to each one of them.

### 3.2.6 Sentiment and Emotion

Sentiment qualities, such as a statement's polarity or emotional intensity, are thought to be helpful in spotting irony and sarcasm. Sentiment polarity was mentioned as one of the features in the experiment by Khokhlova et al. (2016). Sentiment is seen as being employed as an essential type of feature in sarcasm identification because sarcasm and irony are strategies to evoke a particular sentiment in a person. Sentiment was considered a form of feature in a number of sarcasm detection studies (Amir et al., 2016; A. Ghosh & Veale, 2016; Joshi, Goel, et al., 2016). In their investigation, Poria et al. (2016) found that sentiment and emotion features were the most helpful features, aside from baseline features. From their multimodal dataset, Schifanella et al. (2016) proposed characteristics for sarcasm identification which included subjectivity and sentiment scores.

### 3.2.7 Context

In recent years, contextual features have become increasingly popular. Contextual features significantly facilitated sarcasm detection, and numerous studies have since done so (Amir

et al., 2016; A. Ghosh & Veale, 2016, 2017; Poria et al., 2016; Sreelakshmi & Rafeeqe, 2018). The conversational circumstances between the users and the corresponding responses in the datasets from Twitter and Reddit included in the FigLang2020<sup>5</sup> shared task are used to determine whether the comments are sarcastic or not. These contextual characteristics can include details about the author or addressee, the audience, the reaction, the setting, the past, etc.

Hazarika et al. (2018) created user embeddings that capture distinctive behavioral patterns for sarcasm by combining contextual information with user profile, which processed both content and contextual information. In their sarcasm detection study, Zhang, Zhang, and Fu (2016) used local and contextual features. They found that using only local tweet features increased the neural model’s accuracy to 78.55%, whereas using local and contextual features together increased accuracy to 90.74%, demonstrating the importance of contextual features in sarcasm detection.

### 3.2.8 Image

In this work, I have mostly focused on textual datasets, with a few instances of multimodal datasets where texts are associated with certain images as captions, as described in the Datasets section. Text features and image features were the two main feature groups employed in studies utilizing multimodal datasets, such as the well-known dataset of Cai et al. (2019). The context and meaning of accompanying writings can often be inferred from an image’s attributes. After adapting a visual neural network with parameters learned from ImageNet13 to multimodal (text+image) sarcastic posts, Schifanella et al. (2016) came to the conclusion that visual characteristics improve the performance of the textual models.

### 3.2.9 Feature Extraction Methods

Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency are the two most widely used feature extraction techniques (TF-IDF). The simplest approach of feature

---

<sup>5</sup><https://competitions.codalab.org/competitions/22247>



extraction in textual data, the bag-of-words method simply reduces a document to a collection of words. By giving weights to words within the documents, TF-IDF expands the BoW approach to include two new terms.

The Bag-of-Words feature extraction technique was used by D. Ghosh, Fabbri, and Muresan (2017); Jamil et al. (2021); Xiong, Zhang, Zhu, and Yang (2019) in their investigations. Zhang et al. (2016) employed TF-IDF to extract a few of the features from the feature sets. In their respective studies, Dharwal et al. (2017); Jain, Agrawal, Goyal, and Aggrawal (2017), and Onan (2019) also employed TF-IDF as one of the feature extraction strategies. These two methods do have certain drawbacks, though. Both BoW and TF-IDF, which focus on frequency of occurrence, are unable to comprehend the context of a text, which may be extremely important for sarcasm identification.

Words can be converted into vectors using a variety of word embedding approaches. A well-known word embedding method is Word2Vec. Word2Vec uses unsupervised learning, mapping words to the words they occur with most frequently. The Word2Vec technique was utilized by Schifanella et al. (2016) to extract features from multimodal dataset.

For feature extraction, the Word2Vec approach was also employed by Joshi, Tripathi, Patel, Bhattacharyya, and Carman (2016) and Oraby et al. (2017). Continuous Bag of Words (CBoW) and Skip-Gram15 are the two basic Word2Vec approaches. In contrast to the Skip-Gram technique, which predicts a target word based on the words around it16, the CBoW method bases its predictions on context. A few Word2Vec variants, such Doc2Vec, and Emo adopting Eisner, Rocktäschel, Augenstein, Bošnjak, and Riedel (2016) ideas, use a related concept to create vectors from various corpora.

While Subramanian et al. (2019) retrieved and embedded emoji tokens using the Emoji2Vec technique, Khotijah, Tirtawangsa, and Suryani (2020) extracted features using the Doc2Vec technique. Another word embedding method is GloVe (Global Vectors),17 which creates a vector space with useful substructure by training the model simply on the nonzero elements of a wordword co-occurrence matrix rather than the full sparse matrix or specific context windows in a huge corpus (Pennington, Socher, & Manning, 2014). Instead of simply concatenating the feature vectors from various modalities, Cai et al. (2019) performed modality fusion in their multimodal dataset and used the GloVe technique to

obtain those vectors.

One of the most popular Word Embedding strategies is FastText. Although the FastText technique is somewhat similar to Word2Vec, it also uses N-grams to generate a variety of distinct word variations. Word2Vec, GloVe, and FastText approaches were employed by Mehndiratta and Soni (2019) to extract features. These three feature extraction methods were also employed by Onan (2019). Although Word2Vec and Glove word embedding techniques are quite effective in mapping and labeling data, they cluster words together that have polar opposite meanings, making it challenging to discern sarcasm. They also struggle with words that they don't know, while FastText partially solves this problem by utilizing n-grams.

Many well-known machine learning models, including the Convolution Neural Network (CNN), Support Vector Machine (SVM), numerous iterations of the Bidirectional Encoder Representations from Transformers (BERT) and Long Short-Term Memory (LSTM), Embeddings from Language Models (ELMo), etc., are also used to facilitate the feature extraction process. For POS tagging, Bharti et al. (2016) used a Hidden Markov model, or HMM-based approach.

The NRC Word-Emotion Association Lexicon (EmoLEx) was employed in numerous researches. The eight fundamental emotions identified by Plutchik are anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. This lexicon contains a list of 14,182 English words (unigrams) that belong to two feelings (positive or negative) (Khokhlova et al., 2016). In their study, Agrawal and An (2018) used EmoLex to construct sentiment labels. ResNet, SentiWordNet (Baccianella, Esuli, & Sebastiani, 2010), SentiBank (Borth, Chen, Ji, & Chang, 2013), TExtBlob20, LIWC21, COMET (Bosselut et al., 2019), etc. are a few additional important feature extraction techniques.

### 3.3 Methodologies

In this section, the methods and approaches that have been employed in sarcasm detection over time are explored.

### 3.3.1 Rule-based Approaches

Set of rules that express sarcastic cues and indicators comprise rule-based techniques. Since rule-based techniques only use a set of rules based on the provided data, training is not required. The majority of earlier publications (Davidov, Tsur, & Rappoport, 2010; González-Ibáñez, Muresan, & Wacholder, 2011) have developed rule-based or pattern-based techniques for sarcasm identification. In order to determine if the sentiment disagrees with the hashtag, Maynard and Greenwood (2014) used the hashtag from the sample tweet. If conflict is present, the tweet is labeled as sarcastic.

A rule-based classifier that employs a bootstrapping method was proposed by Riloff et al. (2013) to identify a positive verb in a negative context in a tweet. On a dataset built from tweets, Bharti et al. (2015) proposed two rule-based classifiers for sarcasm detection. One of the classifiers looks for interjections and intensifiers that occur simultaneously in order to identify hyperboles. The second classifier employs a parse-based lexicon-building technique that creates phrase parse trees and identifies kinds of expressions that represent emotion.

In an effort to structure sarcasm, Bharti, Pradhan, Babu, and Jena (2017b) developed six rule-based algorithms for sarcasm recognition based on the many types of sarcasm they classified in tweets. Such as Positive Tweet that contains a word and its antonym pair, Contradiction between Tweets and the Universal Facts (CTUIFs), Contradiction between Tweet and its Temporal Facts (CTTFs), Tweet Start with Interjection Word (TSIW), and Parsing-based algorithm for lexicon creation (PBALN) (PTCAP). These algorithms outperformed the most recent methods in terms of precision, recall, and F-measure.

In order to identify tweets with references to oneself that may be self-deprecating sarcasm, Abulaish and Kamal (2018) utilized a rule-based technique that leverages regular expressions to build filtering rules rather than string matching for the first layer of the model. Sundararajan and Palanisamy (2020) suggested a rule-based method consisting of a fuzzy rule-based type detection and a rough set-based type detection and validation in order to further categorize sarcastic tweets into distinct types of sarcasm.

A hybrid approach was proposed by Rendalkar and Chandankhede (2018) that com-

bines SentiNetWord22-based word-based detection to determine the sentiment of the input with an Emoticon-based approach to leverage emoticons found in comments to identify sarcasm.

The types of sarcastic occurrences depend on languages, places, and various social media platforms even though no training is necessary because the rules are based on data. Because the model does not learn anything from the new data, the same principles will not hold true for a novel and varied dataset. It is also challenging to develop a complete set of standards for the enormous amount of data on social media because sarcasm does not always follow norms and can have a variety of structures and markers.

### 3.3.2 Classic Machine Learning-based Approaches

Numerous works have been using machine learning classifiers for sarcasm detection for many years. SVM is one of the most widely used classifiers, as evidenced by the various works (Davidov et al., 2010; Joshi, Goel, et al., 2016; Joshi et al., 2015). Employment of Sidentitiesth SMO (Sequential Minimal Optimization) and logistic regression described in the work of González-Ibáñez et al. (2011). According to Riloff et al. (2013), their SVM-based model outperformed the standard rule-based approaches. In the work of Jansi, Sajja, and Goyal (2018), SVM even outperformed Neural Networks in the identification of sarcasm.

In an effort to distinguish rhetorically sarcastic questions, Oraby et al. (2017) used their SVM-Word2Vec model. They found that the greatest results came from incorporating post-level scores from the Linguistic Inquiry and Word Count (LIWC) (Pennebaker, Francis, & Booth, 2001) method in the model. However, to determine whether a statement is sarcastic or not, Abercrombie and Hovy (2016) and Bamman and Smith (2015) both employed logistic regression.

In a limited dataset, Naive Bayes outperforms an unsupervised fuzzy c-means clustering algorithm for sarcasm detection, as demonstrated by Mukherjee and Bala (2017b). There is a predisposition for the utilization of relevant features that can serve as sarcasm indicators in machine learning algorithms. Naive Bayes also outperformed Maximum Entropy for the majority of tweet datasets that used content and function words as features,

as demonstrated by Mukherjee and Bala (2017a).

Naive Bayes was used by Thakur, Singh, and Singh (2018) to demonstrate that POS tags are not a particularly helpful feature in sarcasm identification. For their domain-general sarcasm detection for both Twitter and Amazon product evaluations, Parde and Nielsen (2018) used Naive Bayes. With their proposed algorithm PBLGA (parsing-based lexical generation algorithm), Bharti, Pradhan, Babu, and Jena (2017a) retrieved lexical, hyperbolic, behavioral, and universal data and demonstrated that Decision Tree outperformed SVM, Naive Bayes, and Maximum Entropy for sarcasm detection. In the experiment conducted by Sreelakshmi and Rafeeqe (2018), SVM with Radial Basis Function (RBF) Kernel surpassed Decision Tree.

Gradient Boosting outperformed the other classifiers in a comparison of different machine learning algorithms that included Prasad et al. (2017) proposed emoji and slang vocabulary. In the work of Khatri et al. (2020), BERT and GloVe embeddings were successfully incorporated as features with Logistic Regression. Additionally, GloVe Embeddings outperformed SVM and Decision Tree when used with Random Forest (Eke, Norman, Shuib, Fatokun, & Omame, 2020). The characteristics for sarcasm detection are also assembled using ensemble algorithms based on Random Forest and Ada-Boost (Sundararajan, Saravana, & Palanisamy, 2021). Banerjee, Bhattacharjee, Ghosh, and Chatterjee (2020) claimed that KNN and other lazy learners are not suited for sarcasm detection when minority oversampling is utilized in an effort to eliminate class imbalance.

Latent semantic analysis (LSA) was used by Di Gangi, Bosco, and Pilato (2019) to take advantage of a Distributional Semantics method, which was then tested using a variety of machine learning classifiers (SVM, Logistic Regression, Random Forests, and Gradient boosting). The issue with standard machine learning models is that to make the patterns in the data accessible to the learning algorithm and to simplify the model, a set of hand-crafted features must be created. For the hard core feature extraction, where the features are most suited for sarcasm detection, we need to be familiar with that specific domain and data patterns.

### 3.3.3 Deep Learning-based Approaches

The use of deep learning models or hybrid models (a combination of deep and traditional machine learning models) for sarcasm detection has increased as more deep learning models have been developed. Deep learning models gained popularity because of their ability to automatically extract features, which lets us avoid gathering manually created features. Poria et al. (2016) were the first to automatically extract sentiment, emotion, and personality variables and send them to an SVM for the final classification using a pre-trained CNN.

Many different deep learning models are being used in tandem, like in the experiment by A. Ghosh and Veale (2016), where the authors presented a CNN model whose output is fed into an LSTM layer before being fed into a DNN layer in order to capitalize on the strength of semantic modeling. With an F1-score of 0.92, the model appeared to perform better than the work of Riloff et al. (2013) and Davidov et al. (2010). The performance of deep learning models has been demonstrated to quickly surpass that of conventional machine learning models. A NN model combining RNN and LSTM that automatically extracts features was developed by Porwal, Ostwal, Phadtare, Pandey, and Marathe (2018) and Salim, Ghanshyam, Ashok, Mazahir, and Thakare (2020).

For their Reddit dataset, Guo and Shah (n.d.) demonstrated that LSTM outperformed baseline Bag-of-Words, Naive Bayes, and even the conventional 'vanilla' neural network because LSTM enables the network to more effectively learn sequential data without overfitting, utilizing the contextual data for sarcasm detection. Mehndiratta et al. Mehndiratta and Soni (2019) demonstrated that their single LSTM model outperformed their hybrid LSTM-CNN model for sarcasm detection.

According to D. Ghosh et al. (2017), employing an LSTM model with conversational context as one of the inputs outperformed using an LSTM model without conversational context in the case of identifying sarcasm in social media interactions. In order to demonstrate that a multiple-LSTM architecture with sentence-level attention performed better than a single-LSTM design, D. Ghosh, Fabbri, and Muresan (2018) used the prior sentence of the sarcastic utterance as context. Diao et al. (2020) developed an end-to-end Multi-

dimension Question Answering model based on Bi-LSTM and attention mechanism using multi-granularity representations to model the semantic relationship between the candidate text and its context. This model outperformed the state-of-the-art machine learning model by a significant margin in case of F1 score.

In order to determine which sort of context is more effective in detecting sarcasm, Ren, Ji, and Ren (2018) tested two different types: conversational context and context based on history (views and opinions about specific events and persons). They used two different forms of context-augmented CNNs—CANN-Key and CANN-ALL—to conduct their research. CANN-KEY integrates only the most important contextual data, whereas CANN-ALL integrates all contextual data. They discovered that while history-based settings performed better due to the relatively small quantity of conversation-based tweets, CANN-ALL has a greater capability in recognizing subtle signs of sarcasm in conversation-based contexts.

Hazarika et al. (2018) proposed a model called CASCADE (a ContextuAl SarCasm DEtector), which uses context information from discussion forums and user embeddings that encode stylometric and personality features of the users with a CNN-based textual model. Sarcasm differs in nature and expression from person to person. Adding the personality traits and the environment substantially enhanced the model’s performance, according to experiments.

Kolchinski and Potts (2018) attempted to model author embeddings and used two different approaches: a straightforward Bayesian approach that only accounts for an author’s inherent propensity for sarcasm and a dense embedding approach that takes into account complex interactions between the author and the text. The baseline bidirectional RNN with GRU cells (BiGRU), which models all user comments, is then extended by these author embeddings. On the entire SARC23 dataset, the approach somewhat underperformed CASCADE, while it outperformed CASCADE on posts from the r/politics subreddit of the same dataset.

On the contrary, Misra and Arora (2019) argued that the model only picks up discriminative lexical signals without access to common sense and current events information. Instead, they eliminated the user embeddings and concentrated on leveraging up-to-date

information and common sense using their LSTM-CNN module, which increased the accuracy of the baseline models by about 5%. These sequence-to-sequence models have a performance issue when dealing with lengthier sentences since they stack all the information from the input source sentence into a fixed-length vector, potentially losing important information.

As the relevance of each word in a sarcastic statement is learned by the attention layer, which then assigns each word a different weight, the subsequent studies use attention based deep learning models to overcome the long range dependence problem. Using manually created auxiliary features, A. Kumar et al. (2020) demonstrated a Multi-Head self-Attention based Bidirectional LSTM (MHA-BiLSTM) that outperforms a feature-rich SVM model (semantic, sentiment and punctuation features). Previously, this SVM outperformed just a BiLSTM model, but on the balanced dataset and unbalanced dataset, it dropped behind the MHABiLSTM model by a large margin of 4.45% and 7.88%, respectively.

GRU with Multi-Head self-Attention, which offers essential indicators for sarcasm, was utilized by Y. Liu et al. (2021) to boost the interpretability of the model in addition to its great performance. Another attention-based hybrid model was built by A. Kumar et al. (2019) that, for improved performance, combines punctuation-based additional features and soft-attention-based BiLSTM with a deep convolution network.

In order to investigate the impact of various linguistic characteristics on sarcasm recognition, Pandey, Kumar, Singh, and Tripathi (2021) integrated 16 manually produced features with the automated extracted features from their Hybrid attention based LSTM model. Although adding an attention mechanism made the issue of long-range dependencies in sequence models easier to deal with, parallel processing is still not possible due to the sequential character of the models. As we will see in section 6.4, transformer-based models are able to handle these issues.

GRU and LSTM only analyze words one at a time, which makes it difficult to model many phrases' contrast, incongruity, and long-range relationships. Tay et al. (2018) were the first to suggest a Multi-dimensional Intra-Attention Recurrent Network (MIARN) that takes advantage of intra-sentence relationships based on the theory of compositional learning to address this. Across all six datasets used, MIARN outperformed models like



NBOW, CNN, LSTM, ATT-LSTM (Attention-based LSTM), GRNN (Gated-RNN), and CNN-LSTM-DNN. Akula and Garibay (2021) employed MIARN as encoders for their Dual-Channel Network to model both superficial and deep meanings of feelings in order to identify sentiment conflict in the input texts.

Using a convolution module (CNN), an importance weighting module, and a self-attention module, Pan et al. (2020) introduced snippet-level self-attention to model the incongruity between sentence snippets. This method performed better on Twitter datasets than long text datasets.

### 3.3.4 Multi Task Learning

Multitask learning is a cutting-edge learning method that use a single neural network to carry out multiple categorization tasks at once. Few works employed multitask learning to identify irony.

In order to classify sentiments and identify sarcasm, Majumder et al. (2019) used multitasking learning. They used Glove word embeddings for word representations and GRU with an attention mechanism to accomplish sentence representations. For the two challenges, they used two different softmax layers for classification.

The multitask classifier’s performance for detecting sarcasm was significantly enhanced with Twitter’s addition of NTN (neural tensor network). Similarly, Savini and Caragea (2020) used sentiment classification as an auxiliary job in multi-task learning to enrich the primary task of sarcasm detection. The BiLSTM model, ELMo24 embeddings, and FastText embeddings are used across both challenges, however, the multi-layer perceptron is different and no user embeddings. This model outperformed word embeddings modeling methods (CNN-SVM, CUE-CNN), although it is just 0.7% less efficient than CASCADE snippet-level

### 3.3.5 Ensemble Learning

Few studies have used ensemble learning strategies rather than a single classifier to identify sarcasm. As their sarcasm detection classifier, Jain et al. (2017) employed Random

Forest and Weighted Ensemble, two distinct ensemble learning techniques. Naive Bayes, Linear Regression, and Random Forest serve as the weighted ensemble model’s component classifiers. They claim that ensemble-based methods have higher recall and precision, and that the efficiency of these methods is mostly influenced by the efficiency of the individual classifiers. Potamias, Siolas, and Stafylopatis (2019) introduced the Deep Ensemble Soft Classifier (DESC), which consists of three deep models: a BiLSTM, an AttentionLSTM, and a Dense NN.

DESC outperformed every model used in the Sentiment Analysis test for SemEval-2015. A vote classifier was used by Gupta, Kumar, Agrawal, et al. (2020) to select the best outcome from among the results provided by various machine learning classifiers. An ensemble strategy that is trained using both extra features and the anticipated sarcasm probabilities of four component models was suggested by Lemmens, Burtenshaw, Lotfi, Markov, and Daelemans (2020).

### 3.3.6 Transformer based Approaches

Transformer models address the limitations of earlier architectures by using attention blocks that account for short- and long-range dependencies with the same likelihood. ALBERT (Lan et al., 2019), RoBERTa(Y. Liu et al., 2019), and BERT or some variant of BERT are among the Transformer models that the majority of modern architectures typically utilize. To handle the input contexts for a specific answer, Srivastava, Varshney, Kumari, and Srivastava (2020) employed a hierarchical BERT-based model, which comprises of a context-summarization layer, a context-encoder layer, a CNN-Layer, and ultimately a fully-connected layer.

Gregory et al. (2020) investigated novel techniques using a range of transformers to categorize sarcasm in tweets. They discovered that BERT outperformed the separate models. However, an ensemble model that combined the output from five different pre-trained transformer models—BERT, RoBERTa, XLNet, RoBERTa-large, and ALBERT—was their best performer. They made the assumption that the ensemble model, as opposed to a single type of embedding, allows for more information in each of the embeddings. In their comparison of BERT’s performance with deep learning models

(RNN-LSTM) and classical machine learning models (SVM, Naive Bayes, Logistic Regression), Kalaivani and Thenmozhi (2020) found that BERT outperforms all other models because it is effective with continuous chat dialogues.

Potamias et al. (2020) were the first to develop an end-to-end model that used an unsupervised pre-trained transformer technique in figurative language to lessen the burden of data-preprocessing. With no hand-crafted engineering features or lexicon dictionaries, the pre-trained RoBERTa was integrated with an RCNN (Recurrent Convolutional Neural Network) to capture various types of contextual information.

To identify sarcasm on Twitter and Reddit, Javdan, Minaei-Bidgoli, et al. (2020) suggested using a combination of aspect-based sentiment analysis and BERT. While LCF-BERT, an aspect-based sentiment classification method first described in the work of Zeng, Yang, Xu, Zhou, and Han (2019), performed best on the Twitter dataset, the individual BERT model performed best in Reddit. With less complex data, such as Twitter posts, treating them as two independent portions works better as aspect-based approaches aim to learn the incoming data more dynamically.

Two publicly accessible BERT models, TD-BERT (Gao, Feng, Song, & Wu, 2019) and BERT-AEN (Song, Wang, Jiang, Liu, & Rao, 2019), which use attention encoder networks to model the semantic interaction between the given sentence and the potential target, were modified by Parameswaran, Trotman, Liesaputra, and Eyers (2021) to demonstrate that BERT models outperform the current state-of-the-art models for sarcasm target detection.

It was assumed that the multiple attention mechanism in BERT-AEN would perform better than TD-BERT, but surprisingly, TD-BERT outperformed both BERT-AEN and the other baseline models. Simply taking into account the target's position improved TD-comprehension BERT's of the situation, which may have contributed to the performance improvement. Adversarial and Auxiliary Features-Aware BERT (AAFAB), a new model for sarcasm detection developed by A. Kumar et al. (2021), combines high quality manually extracted auxiliary features with an attempt to encode the semantic meaning of a sentence.

A further phase known as adversarial training was carried out by introducing perturbations to the input word embedding in order to increase parameter generalization along with BERT word embedding, BERT encoding, and feature concatenation. On both balanced

and unbalanced datasets, AAFAB outperformed a number of baseline models based on deep learning. The first Affective Dependency Graph Convolutional Network (ADGCN) framework for sarcasm detection has been proposed by Lou et al. (2021). Based on affective commonsense knowledge and dependency trees, this method generates an affective graph and a syntax-aware dependency graph for each of the sentences. It employs multi-layer GCNs to take advantage of the emotive dependencies of the context and BERT to learn the vector representations of the context for sarcasm detection.

Reactive supervision is a novel data collection technique that Shmueli, Ku, and Ray (2020) devised in order to gather less noise sarcastic data utilizing dialogue signals. They produced SPIRS27, a new sizable dataset with added features and improved labeling. They enabled the new task of sarcasm detection, where the model seeks to determine whether the sarcasm is meant or not. Pre-trained BERT outperformed other deep learning techniques in the evaluation. Although a transformer model can handle hierarchical inputs more quickly with parallel processing since each step of the calculation is done in parallel, the model is still unable to use the highest-level representations of the input sequence from the past to compute the current representation (Fan, Lavril, Grave, Joulin, & Sukhbaatar, 2020).

## 3.4 Trends

In this section, I looked at various patterns in earlier computational sarcasm detection research.

Tepperman et al. (2006) published the earliest known research on sarcasm detection. Following this, numerous studies experimented with supervised and semisupervised techniques that were centered on finding patterns and feeding these patterns as features to a statistical or rule-based classifier. Hashtag-based remote monitoring spread as Twitter gained popularity as a source of data. After that, it became popular to use contextual data, including author, audience, dialogue, visual data, and so forth. Most recently, researches have indicated a fascination with deep learning and transformer-based techniques. With this section, I hope to provide readers with a thorough understanding of the methods used

to identify sarcasm on social networking sites nowadays. Hence, the more recent trends will be the main focus of my analysis.

### 3.4.1 Integrating Context

Using the context to formulate predictions has become more and more common in recent years. The text that has to be classified will be referred to as “target text” from this point on until the end of the section, and “context” will refer to any information that is not the target text. Contextual information can be conversational, authorial, visual, target, or cognitive[55]. The importance of context in sarcastic text detection was initially studied and advanced by Wallace, Kertz, Charniak, et al. (2014). They discovered that the sentences that the machine learning system misclassified were also the ones for which their human annotators commonly asked for further context while utilizing the Bag-of-Words (BoW) method in their Reddit ironic corpus<sup>29</sup>. They reasoned on the basis of a such justification that machine learning algorithms need context and human annotators require context as well. The results of Wallace et al. (2014) inspired other subsequent investigations to incorporate context into their sarcasm detection systems Joshi, Goel, et al. (2016); Kolchinski and Potts (2018); Plepi and Flek (2021). I came across numerous context-sensitive architectures. Topical context, authorial context, and conversational context are the three sorts of contexts we have seen thus far.

#### Topical Context

As the name implies, topical context usually alludes to the subject matter of the target text. In their study, Wang et al. (2015) used such topic-based context and viewed sarcasm detection as a sequential classification challenge. By extracting the entire tweet sequence—including numerous tweets using the same hashtag before the target message for the topic-based context—they were able to create a Twitter dataset. Using a Twitter dataset, Joshi, Tripathi, et al. (2016) suggested a topic model for finding sarcasm-prevalent themes and topic-level sentiment. They found that during the time of their study, sarcasm was more common on topics like “work”, “gun laws”, and “weather”.

### Authorial Context

Authorial context is the trail the target text’s author has left behind. Sarcasm was associated with the mismatch between the conveyed emotion and the author’s situation, according to Tay et al. (2018). For the authorial context, Bamman and Smith (2015) retrieved extra-linguistic data such as the author’s previous salient terms, themes, profile information, sentiment, and profile unigrams. These characteristics, along with the background from the intended (or perceived) audience and the dialogue between them, significantly improved their accuracy when compared to earlier experiments. This raised the new threshold for sarcasm recognition on social media networks. Mukherjee and Bala (2017b) attempted to determine the author’s writing style revealed authorial characteristics, such as function words and part of speech n-grams, to be crucial for sarcasm detection. The idea of considering the author’s mood was put forth by A. Ghosh and Veale (2017). They used mood cues from the most recent tweets to represent the author at the time the utterance was created using a deep neural network architecture. Attributes extracted from the response utterances were used to model the context.

### Conversational Context

The dialogue or discussion between the author and the target text’s reader is what is meant as the conversational context. Bamman and Smith (2015) retrieved binary indicators of pairwise Brown characteristics between the original and response messages in order to capture the conversational context between the target tweet and the tweet to which it is responding. The whole tweet thread, including tweets that came before the target tweet and represented the dialogue with other users, was compiled by Wang et al. (2015).

Both of these techniques were used by D. Ghosh et al. (2017) to construct their conversational context, which included 25,991 instances. The same dataset produced by Wang et al. (2015) was used by Ren et al. (2018). In order to assess the role of context, Ren et al. (2018) chose a neural network architecture like CNN since Wang et al. (2015) employed  $SVM^{multiclass}$  and  $SVM^{hmm}$  in their investigation. On the other hand, D. Ghosh et al. (2018) selected LSTM and conditional LSTM networks to confirm the potency of

conversational context in sarcasm detection. As conversational background, they took into account the turn that came before, the turn that followed, or both. In addition to these experiments, techniques based on the Transformer have frequently used conversational context. The subheading (section) on Transformers goes into further detail about this.

### 3.4.2 Deep Learning

The earliest study on deep learning models for NLP dates back to 2011 by Collobert et al. (2011). However, the work of Poria et al. (2016) was the first to employ deep learning in the field of sarcasm detection. They built three models—sentiment, emotion, and personality—using CNN, each using its own dataset. The attributes from these pre-trained models were then supplied to an SVM, which categorizes the text. The prior state-of-the-art approaches were exceeded by their architecture. They also showed how important sentiment incongruity, emotion, and personality factors are for sarcasm identification using their system.

Amir et al. (2016) proposed a CNN-based model to learn and take advantage of user embeddings. Majumder et al. (2019) created an architecture that made use of the association between sarcasm detection and sentiment analysis rather than treating them as distinct tasks. Their multitask framework’s performance on sarcasm detection was enhanced by the addition of NTN fusion, while the performance on sentiment classification was boosted by the introduction of an attention network shared by both tasks. With the advent of deep learning, many types of data in addition to textual data were supplied to the models for training.

### Numerical Data

This particular trend developed as a result of the need to recognize sarcasm in numerical expressions. Examples include “Love traveling three hours to work every day” and “Started the day with 22% charge on my phone”. Understanding the function of numbers is essential for spotting the underlying irony in statements. Along with several rule-based and machine learning-based approaches, L. Kumar, Somani, and Bhattacharyya (2017)

used a number of deep learning techniques in their study to deal with such scenarios. The best outcome was generated by their CNN-FF (CNN followed by Fully Connected Layer) model, which was based on deep learning. To cope with sarcasm stated through numerical expression, Dubey, Kumar, Somani, Joshi, and Bhattacharyya (2019) also suggested certain deep learning architectures. A CNN-FF model and an attention network were used to create the first. Both performed better than their previous works, with CNN-FF outperforming the attention network with an F1-score of 0.93 as opposed to 0.91.

### 3.4.3 Transformers

Another recent trend in sarcasm detection is the use of transformers. A deep learning model called a transformer utilizes a self-attention mechanism to weigh the incoming data in accordance with a recognized relevance metric. Although Vaswani et al. (2017) first introduced transformers in 2017, the use of transformers for sarcasm detection is still relatively new.

The first to use an approach based on unsupervised, pre-trained transformers was Potamias et al. (2020). Recurrent CNN is applied to a pre-trained transformer-based network architecture in their suggested methodology, RCNN RoBERTa, which outperforms cutting-edge techniques including BERT, XLnet, ELMo, and USE. Following this, using pre-trained language models to solve sarcastic classification issues only became more popular. In this field, transformers have been applied in a variety of ways. Two such transformer-based architectures: using conversational context and using Multiple Modals should be better examined in following subsections.

#### Utilizing Conversational Context

Context is used by transformer-based models to detect sarcasm. In the studies (Avvaru, Vobilisetty, & Mamidi, 2020; Dong, Li, & Choi, 2020; Gregory et al., 2020; Javdan et al., 2020), transformer-based designs have utilized conversational context more often than other types of context. Such situations are prevalent throughout the 2nd Workshop at the



Figurative Language Processing 2020 shared task (FigLang2020<sup>6</sup>), where conversational context was added to the datasets from Twitter and Reddit. Both the immediate context (i.e., merely the previous discussion turn) and the complete discourse thread, if available, were included in the contextual information. Many teams choose an architecture that uses transformer layers in their model in order to take advantage of the given context since transformers are efficient at tracking relationships among sequential data.

Dong et al. (2020) provided a model for the FigLang2020 shared task that made advantage of deep transformer layers and the complete conversational environment. This led to improvements of 3.1% and 7.0% for the Twitter and Reddit datasets, respectively, above the baselines of 0.67 and 0.6 F1 scores published by D. Ghosh et al. (2018). Lee, Yu, and Kim (2020) presented an architecture for the same shared work that stacks a transformer encoder with BiLSTM and NeXtVLAD Lin, Xiao, and Fan (2018). The conversational context of the unlabeled dataset was used to produce fresh training samples using the data augmentation approach known as CRA (Contextual Response Augmentation). They also investigated various context lengths using the context ensemble method. As a result, they noticed a notable improvement in F1 scores of 0.931 and 0.834 in the corresponding datasets for Twitter and Reddit.

### Using Multiple Modals

Recently, image encoders and transformer-based models have been applied to multimodal sarcasm detection. One of the earliest prominent works that included numerous modals in its sarcasm detection architecture was the Hierarchical Fusion Model by Cai et al. (2019). Their architecture, as previously indicated, separates the text, picture, and image-attribute features before reconstructing and fusing them. However, rebuilding features required omitting certain information.

Pretraining models using image text data gained popularity as transformer-based models were developed Alberti, Ling, Collins, and Reitter (2019); Lu, Batra, Parikh, and Lee (2019); Lu, Goswami, Rohrbach, Parikh, and Lee (2020). But according to Wang et al. (2015), BERT and ResNet should both be pre-trained on significantly bigger text data

---

<sup>6</sup><https://competitions.codalab.org/competitions/22247>

sets rather than only image-text data. As a result, they created an architecture that uses pre-trained BERT and pre-trained ResNet together, without the need for additional pre-training, and constructed a bridge between the two. As a result, the architecture is flexible since BERT can be replaced with any transformer-based model and ResNet may be converted to any other visual model. The Cai et al. (2019) architecture was also improved upon by Pan et al. (2020). The incongruity between images, text, and hashtags was the subject of Pan et al. (2020) study. With the use of a co-attention matrix, they established a relationship between text and hashtags. Text and picture matching was then carried out using BERT for text and ResNet-152 for image encoding, respectively. The outcomes of these two interactions were then contrasted. The effectiveness of multimodal sarcasm detection was greatly enhanced by this merging of transformer-based encoders with picture encoders.

# Chapter 4

## Experiments and Results

### 4.1 Description of Datasets

I conducted experiments on four benchmark datasets: two Reddit (Khodak et al., 2018) subreddits datasets: SARC/movies and SARC/technology, a subset of Internet Argument Corpus-V2 (M. Walker, Tree, Anand, Abbott, & King, 2012) (IAC-V2), and Twitter (Ptáček, Habernal, & Hong, 2014). All of them have been widely used in evaluating sarcasm detection. The details are shown in Table 4.1.

Datasets	Train		Test	
	Sarcastic	Non-sarcastic	Sarcastic	Non-sarcastic
SARC/movies	2,533	2,707	641	669
SARC/technology	2,738	1,815	677	462
IAC_V2	2,616	2,600	644	660
Twitter (Ptacek et al., 2014)	22,323	25,785	5,648	6,379

Table 4.1: Statistics the experimental data.

#### 4.1.1 Reddit

Khodak et al. (2018) introduced the Self-Annotated Reddit Corpus (SARC). It includes more than a million statements, both sarcastic and not, that have been pulled from Reddit, along with some contextual data including author information, score, and parent comment.

Reddit is a social networking platform where users can converse on subreddits, or discussion boards, dedicated to particular topics. Voting and replying to submissions or comments results in the formation of a tree-like structure. The primary distinguishing characteristic of the dataset is the writers' direct annotation of sarcastic statements through the use of the marker “/s” in their comments. This approach offers accurate and reliable data. The fact that practically every comment consists of a single sentence is another significant factor. We only execute our experiments on the two subreddits: SARC/movies and SARC/technology. The subreddits contain news and discussions concerning films and development, application, and related issues of technology, respectively.

### 4.1.2 Internet Argument Corpus

Internet Argument Corpus(IAC) (M. Walker et al., 2012), one of the first larger scale corpora available for opinion sharing dialogue (M. A. Walker et al., 2012). Internet Argument Corpus 2.0 (IAC-V2) is a subset of IAC, developed by incorporating discussions from additional websites and restructuring them into a brand new data architecture in SQL.

The language in discussion forums, and particularly in online forums on social and political issues, is very different from newspaper articles or mainstream news. Subjective writing in traditional media have a tendency to be both monologic and formal. Online debates, on the other hand, are intensely dialogic, interpersonal, composed of less formal language and frequently involve expressive and emotional vocabulary. The IAC-V2 offers a larger dataset made of dialogues from *ConvinceMe.Net*, *CreateDebate.com*, and *4forums.com*.

*4forums.com* is an online forum for political discussion and debate. It covers a wide range of topics relevant particularly to the US political arena. Users can initiate discussions or reply to others in an ongoing conversation, as well as cite other posts entirely or in partially.

Dialogues from *ConvinceMe.net*, a highly structured discussion website, are also included in IAC-V2 in addition to the *4forums* dataset. The dataset consists 65,368 postings in 5,413 disputes, and 5,783 authors. Users can initialize a debate by stating the issue and opposing viewpoint. Then, other users are pressured to identify their position while

commenting by posting. Other users are compelled to declare their own position by either posting on the side they favor or by using a response system that moves their post to the opposing party.

Similar to *ConvinceMe*, the discussion website *CreateDebate.com* employs a well-structured two-sided approach. The user initiating the discussion establishes the topic, an introduction statement, and the sides/parties of the argument. Depending on their stance, top-level posts are positioned in the left or right column. When responding, users must choose one of the available sides to represent their position and use the tags “support”, “clarify”, or dispute tag. Responses are shown inline under their parents, in contrast to *ConvinceMe* responses, which promotes a more natural discussion.

Even if users self-identify with the same stance, it is still feasible for one user to dispute the other, providing researchers a chance to study how advocates of the same opinion may disagree on particular sub-issues (Sridhar, Foulds, Huang, Getoor, & Walker, 2015). *CreateDebate* lets users vote on other posts, which are also included in the dataset. The effectiveness of persuasion has been studied using these votes (Jaech, Zayats, Fang, Ostendorf, & Hajishirzi, 2015).

IAC-V2 divides sarcasm into three sub-types, (i.e., general sarcasm, hyperbole, and rhetorical questions). I use the largest subset (general sarcasm) in our experiments.

### 4.1.3 Twitter

We use Twitter dataset provided by Tomáš Ptáček (Ptáček et al., 2014), who collected 780,000 (130,000 sarcastic and 650,000 non-sarcastic) tweets in English with the *#sarcasm* hashtag. I implemented the English balanced version.

## 4.2 Proposed Method

### 4.2.1 Overview of the Proposed Approach

The model consists of four blocks: Sarcasm Pre-Trained Transformer (SarcPTT), Emotion Detection Pre-Trained Transformer (EmoDPTT), Sentiment Detection Pre-Trained

Transformer (SentDPTT), and CNN block. EmoDPTT and SentDPTT are used as feature extractors and are not trainable during model fitting. Other modules (CNN and SarcPTT) are trainable. A detailed overview of the blocks is presented in the following subsections.

The general flow is presented in Figure 4.1. The input text is tokenized via transformer tokenizer  $\{CLS, T_1, \dots, SEP\}$  and is passed through SarcPTT. The output of this step is a last hidden state, but only vector representation  $V_{CLS}$  of the first token ( $CLS$ ) is used for further processing. After that, the output is passed through Fully Connected Network, and a new vector  $V_l$  is obtained.

The non-trainable blocks (EmoDPTT and SentDPTT) are used as feature extractors. After passing the input text tokenized via transformer tokenizers ( $\{CLS_e, T_1^e, \dots, SEP_e\}$  and  $\{CLS_s, T_1^s, \dots, SEP_s\}$  respectively) through the models, the following features are obtained:

- EmoDPTT vector representation of the first token of the last hidden state:  $U_{CLS}$ ;
- Probability distribution of dimension 28 of EmoDPTT labels, e.g., amusement, relief, disgust, neutral, etc.:  $\{EL_1, EL_2, \dots, EL_{28}\}$ ;
- SentDPTT vector representation of the first token of last hidden state:  $S_{CLS}$ ;
- Probability distribution of dimension 2 of SentDPTT labels, i.e., positive and negative sentiments  $\{SL_1, SL_2\}$ .

Similarly to the SarcPTT output, text representations were passed through Fully Connected Networks, and new vectors were obtained:  $S_l$  and  $U_l$ . Probability distributions of labels were concatenated and passed through a Fully Connected Network, producing vector  $D_l$  as output.

For passing data through CNN block, the input text  $T = \{W_1, W_2, \dots, W_N\}$  was tokenized using *nltk* library<sup>1</sup>, where each  $W_i$  represents a tokenized word. The text  $T$  was passed through CNN, and text representations were obtained as vector  $C$ . Vector  $C$  was passed through a Fully Connected Network, transforming it into vector  $C_l$ .

---

<sup>1</sup><https://www.nltk.org/>

As a final step, all the output feature vectors  $V_l, U_l, D_l, S_l, C_l$  were concatenated and passed through a Fully Connected Network and Softmax transformation, obtaining the prediction.

We made our code available at GitHub<sup>2</sup>.

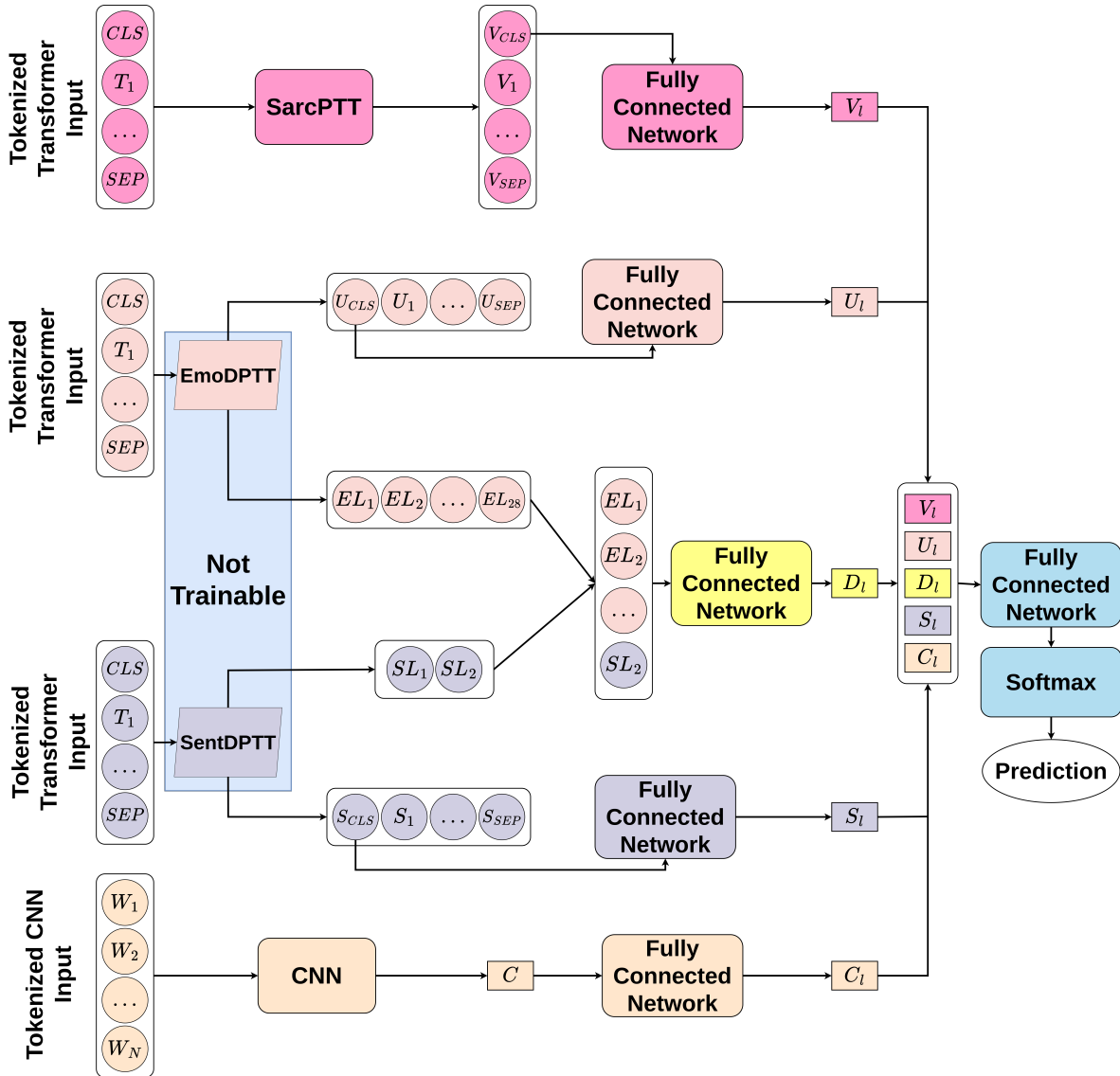


Figure 4.1: Proposed architecture of the model.

<sup>2</sup><https://github.com/Wittmann9/SarcasmTuneUntrainableSentEmo>

## 4.2.2 Sarcasm Pre-Trained Transformer

I merged all training parts of the datasets and then we pre-trained RoBERTa model on it using masked language modeling objective.

## 4.2.3 Emotion Detection Pre-Trained Transformer

As an emotion feature extractor, we used BERT model<sup>3</sup> pre-trained on GoEmotions dataset (Demszky et al., 2020) on multi-label (28) emotion classification task. The input text was passed through this model and corresponding vector representation of CLS token of the last hidden state was obtained, as well as labels' distribution. The architecture is described in Figure4.2.

### GoEmotion Dataset

GoEmotion dataset consists of 58K Reddit comments that have been classified as or one more of 28 emotions including Neutral.

The reddit-data-tools project is used, which includes comments from January 2005 (the year Reddit was launched) to January 2019. Deleted and non-English comments are removed from subreddits that have at least 10,000 comments.

Using pre-defined categories of offensive, vulgar, identity, and religious phrases, and damaging comments are detected. It is employed for data filtering and masking.

Subreddits that are unsuitable for the task, and where more than 10% of comments contain profane, rude, or adult language were deleted. The remaining comments that include offensive or mature language are deleted as well. Offensive remarks are kept because authors think they are essential to understanding how negative emotions work. The list of filtered tokens is included in the dataset.

Identity comments are manually reviewed. Offensive comments towards a particular ethnicity, gender, sexual orientation, or disability are removed.

Utilizing the NLTK word tokenizer, the comments that are 3 to 30 tokens long, including punctuation, were chosen. Comments at the median token count are We downsample

---

<sup>3</sup><https://huggingface.co/bhadresh-savani/bert-base-go-emotion>



to provide a roughly equal distribution of comment length.

Subreddits with a low representation of positive, negative, ambiguous, or neutral sentiment are deleted, to decrease sentiment bias. Emotion prediction model is used, which was trained on a pilot batch of 2.2k annotated instances, to determine the sentiment of a comment. Subreddits with more than 30% neutral comments or fewer than 20% of critical, affirmative, or unclear comments are excluded.

Using the above-mentioned pilot model, a predicted emotion assigned to each comment. Next, emotion bias is eliminated by downsampling the data with weak labels and capping the number of comments that fall within the median emotion count.

Downsampling and capping by the median subreddit count are utilized to prevent the overrepresentation of prominent subreddits. The remaining comments were randomly chosen from 315k comments (from 482 subreddits) for annotation.

To have the broadest coverage in terms of emotions, a pilot task is performed where raters can add their own emotion labels on top of the pre-defined set after manually labeling a small fraction of the data. To cover as many different emotional expressions as possible, the authors referred to psychological literature (A. Cowen, Sauter, Tracy, & Keltner, 2019; A. S. Cowen & Keltner, 2017) on emotional expression and recognition. Since there hasn't been any research to their knowledge that defines the primary categories for emotion recognition in the domain of text, the emotions that have been deemed fundamental in other domains (video and speech) are taken into consideration.

Too many comparable feelings should be avoided because they make the annotation process more challenging. Additionally, grouping labels that are similar and have great coverage would produce an abundance of annotated labels. Therefore, the number of emotions is limited. Table 4.2 contains a list of chosen emotions with examples.

Given pre-defined emotion definitions and a few sample texts for each emotion, raters were asked to identify the feelings that the text's author expressed. The only emotions that the raters were given to choose from were those for which they had a good amount of confidence that the text had portrayed them. Rating participants were instructed to choose Neutral if they were unsure of any emotion being stated. A checkbox was provided so that raters could indicate whether or not an example was extremely challenging to

classify. In that case, they could leave all of the options blank. All instances where no emotion was chosen were eliminated.

There was no further metadata (such as the author or subreddit) displayed with Reddit comments. A table with all emotion categories aggregated by sentiment and whether that emotion is commonly expressed towards anything (e.g. disapproval) or is more of an intrinsic feeling was shown to raters to help them navigate the vast universe of emotion. Raters were instructed to disregard the category whenever they saw fit because it was made clear in the instructions that it was by no means precise but did reflect basic tendencies. To make it easier to understand emotions that transfer easily to emojis, an emoji was displayed alongside the emotion in the user interface (UI).

Table 4.2: List of chosen emotions with examples representations from dataset.

Label	Text
Admiration	“I appreciate it, that’s good to know. I hope I’ll have to apply that knowledge one day”
Amusement	“That’s crazy; I went to a super [RELIGION] high school and I think I can remember 2 girls the entire 4 years that became teen moms.”
Anger	“Oh, how DARE you discuss the disgustingly unhealthy and dangerous lifestyle I pursue! /s”
Annoyance	“Do you not give your snowmen brooms? I feel like that’s a thing people do”
Approval	“If there’s a pattern, yes.”
Caring	“Yup, not anymore. Keep your blood sugar up! It really helps and DRINK water...”
Confusion	“Because the content creators don’t deserve to be paid, your seconds spent listening to ads are too valuable!”
Curiosity	“Now I’m wondering on what I’ve been missing out. Again thank you for this.”

Continued on next page

Table 4.2 – continued from previous page

Label	Text
Desire	“What’s your source for that? Just curious (and yes I know it sounds like a tired contrarian statement).”
Dissapointment	“Lovely places to buy from but very hard to get a good price as a seller because of their high rents (and bargaining expertise!)”
Dissaproval	“Not really cringey. Its not really embarrassing for you i think people might just feel bad upon reading it.”
Disgust	“This sounds an awful lot like khorne.....”
Embarassment	“209 comments and we don’t get to read any? All that good pasta advice going to waste. Shame. ”
Excitment	“cant wait to go through a rebranded version of this pain again in college too ”
Fear	“Any one of us could be a journalist, and that level of journalism feeding the masses is mildly scary.
Gratitude	“Was she? Oh, never been good at picking up on stuff like that. Thanks for letting me know!”
Grief	“Rip the guy from psych” Disappointment
Joy	“Today’s Recaptchas were fun. I’m very good at identifying store fronts, apparently...”
Love	“Here we go again I hDisapproval *becomes a Dodger* I mean I love [NAME]”
Nervousness	“I don’t understand how her parents were able to take kid when she claims they are horrible alcoholics”
Optimism	“That’s what I’d hope for! Retool the D day 1, if we could find a gem at QB day 2.Excitement”
Pride	“And I taught my room was dirty and that I was not taking care of myself. Good job op”
Continued on next page	

Table 4.2 – continued from previous page

Label	Text
Realization	“I have been abusive.” This was before I even realised his narcissistic characteristics.”
Relief	“I’m glad it only sprayed soda when his thumb went into the can, and not blood everywhere.”
Remorse	“Sorry, but anarcho-anything are bot welcome there ”
Sadness	“No, what’s sad is saying the same message like it’s news. He’s a copy paste machine. No original thought.”
Surprise	“The altcoin thread!? This is the first time I’ve seen it in 6 months.”
Neutral	“It’s the issue with a lot of ‘activists’ who think all’s they have to do is exist.”

#### 4.2.4 Sentiment Detection Pre-Trained Transformer

As a sentiment feature extractor, I used SiEBERT model (Hartmann, Heitmann, Siebert, & Schamp, 2022). It is a fine-tuned on sentiment classification task RoBERTa-large model (Y. Liu et al., 2019). The model was trained on 15 datasets from different text sources (reviews, tweets, etc.). The input text passed through the model and corresponding vector representation of CLS token of the last hidden state was obtained, as well as labels’ distribution.

#### 4.2.5 CNN

I used the CNN architecture following Kim (2014), see Figure 4.3. For processing the input text through CNN, I built a vocabulary dictionary for each dataset separately to create an embedding layer. I used Glove Common Crawl<sup>4</sup> pre-trained vectors (42 billion words

<sup>4</sup><https://nlp.stanford.edu/projects/glove/>

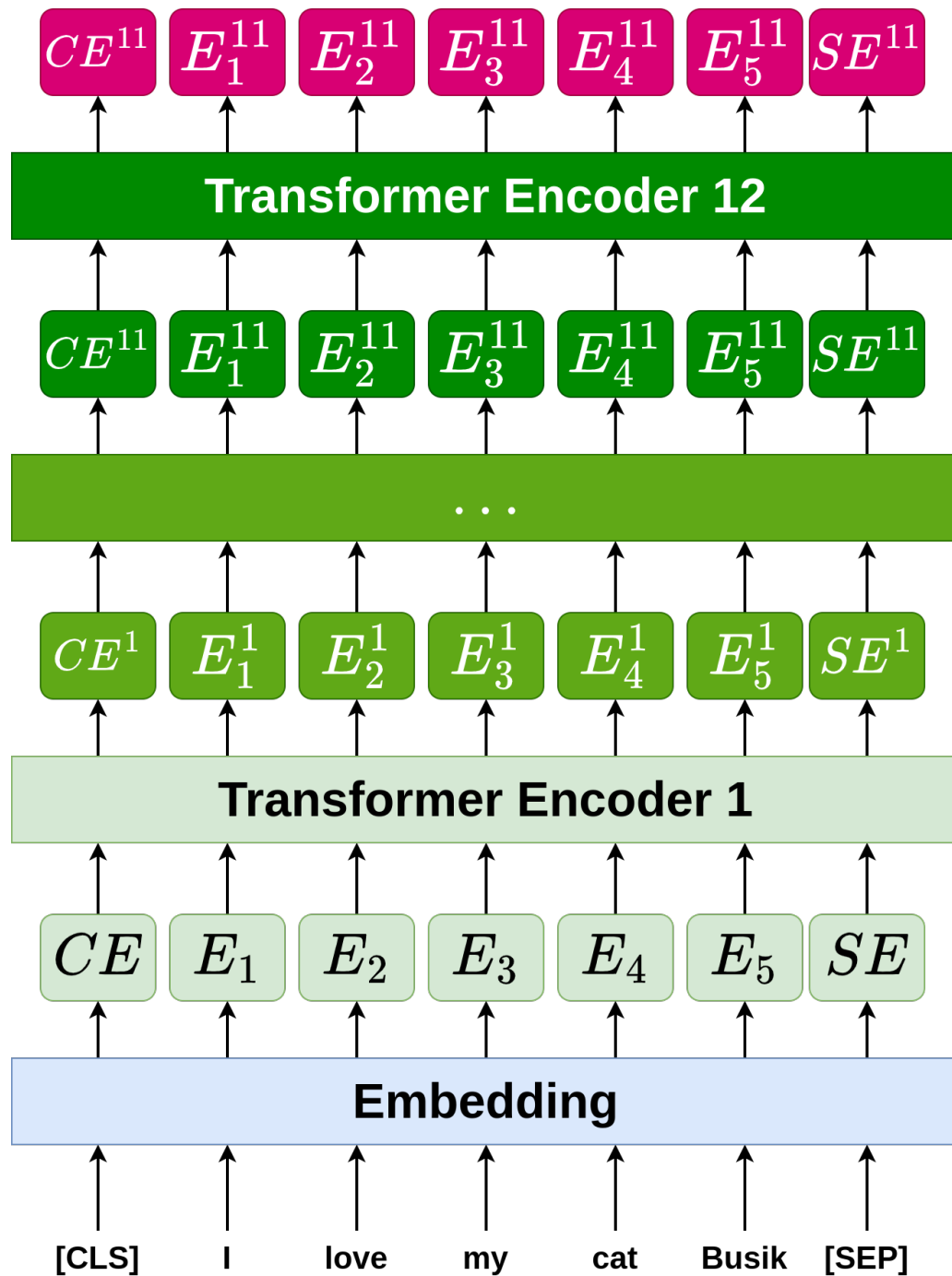


Figure 4.2: BERT architecture.

version). For words with no pre-trained vectors, I checked their stemmed versions. If no pre-trained embedding was found, a random vector was initialized. Then, the input text

was encoded into the embedding matrix of shape  $(N_W, 300)$ , where  $N_W$  is the number of words in the input text and 300 is an embedding dimension.

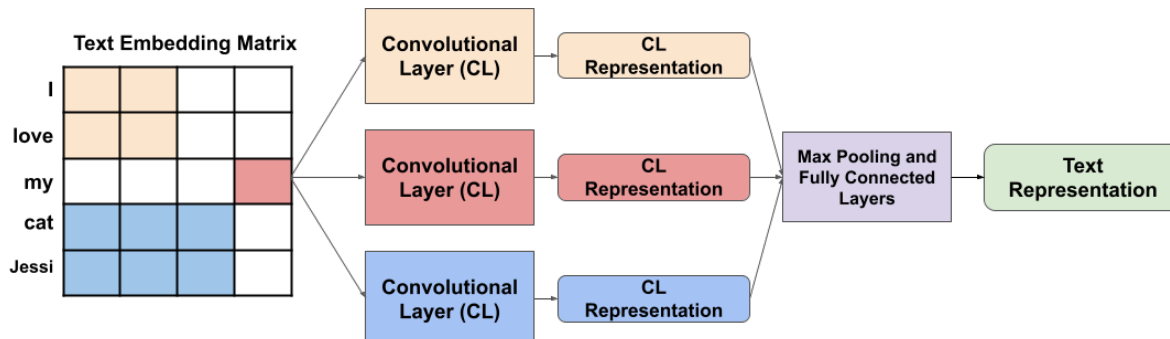


Figure 4.3: CNN block.

I then used convolutions with different filter sizes to extract feature maps from the embedding matrix. Next, we applied the ReLU activation and max-over-time-pooling to reduce each feature map to a single scalar. Then we concatenated these scalars into a vector and obtained vector representations for the texts.

## 4.3 Description of Baselines

Since prior studies lack uniform datasets, I run baseline comparison experiments using our datasets and either open-source code or reproduced code.

### 4.3.1 NBOW

A Bag-of-Words BOW represents text (a sentence, paragraph, or, document) as a vector of word features. Traditional BOW methods utilize frequency of word occurrence in the text and variants of TermFrequency-InverseDocumentFrequency (tf-idf) as the word representation feature (Manning, Raghavan, & Schutze, 2008).

More robust continuous vector representations of words have been developed as a result of advancements in neural network and deep learning-based language processing (Mikolov, Chen, Corrado, & Dean, 2013).

It was proven that these inferential word vector representations work better than count-based word (vector) representations due to capability of capturing syntactic or semantic

features of words and their local context (Mikolov et al., 2013).

Neural bag-of-words (NBOW) (Sheikh, Illina, Fohr, & Linares, 2016) baseline takes an average of word vectors in the given text as sentence representation and feeds it into a standard logistic regression model. The Neural Bag-of-Words (NBOW) model architecture describes as follows. A fully connected network maps sequence of words  $X$ , to one of  $k$  classification labels. The model has  $d$  dimension word vector for each word in  $X$ . For the words  $w \in X$ , corresponding input word vectors  $v_w$  are searched and a vector representation  $z$  is obtained as an average of the word vectors in the given text

$$z = \frac{1}{|X|} \sum_{networks} v_w \quad (4.1)$$

The average vector  $z$  is then fed to fully-connected layer to estimate probabilities for the final label as:

$$\hat{y} = softmax(W_i z + b) \quad (4.2)$$

where  $W_i$  is  $k \times d$  matrix,  $b$  is bias vector and  $softmax(q) = \frac{exp(q)}{\sum_{j=1}^q exp(q_j)}$ . For text classification tasks, the NBOW model is trained to minimise the categorical cross-entropy loss using a stochastic gradient descent algorithm.

### 4.3.2 CNN

I used CNN configuration following Kim (2014). The overall architecture is presented in Figure 4.4. The padded embedded sentences are processed via the CNN cells. Next, the ReLU activation function and Max Pooling are applied. The concatenated outputs from the previous step are processed by linear layers to produce the distribution of the classes.

### 4.3.3 CNN-LSTM-DNN

This model (A. Ghosh & Veale, 2016) is a combination of CNN, LSTM, and DNN. It stacks two layers of convolution and two LSTM layers, then passes the output to a DNN for prediction. The architecture of this model describes as follows. A tweet is considered

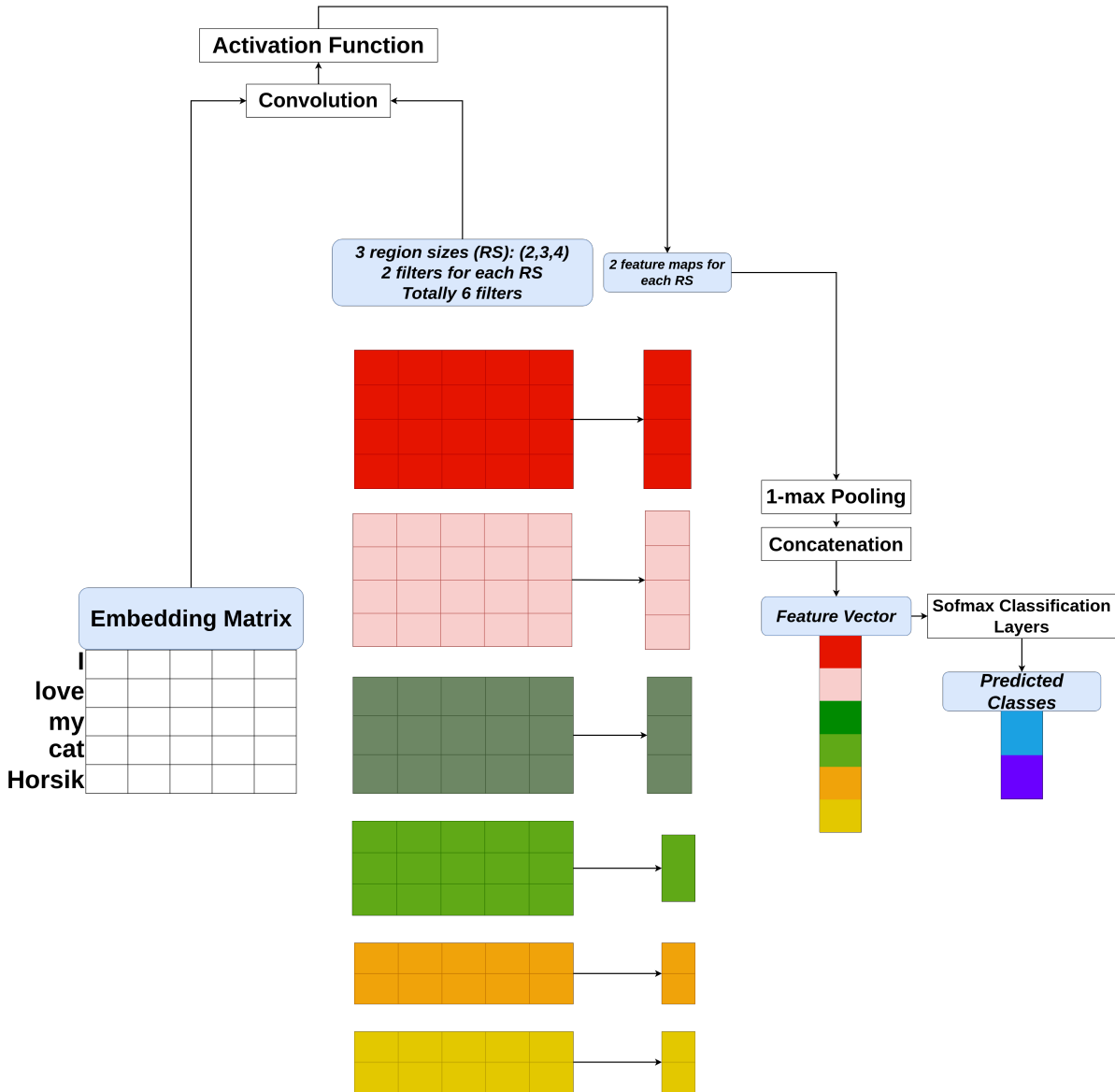


Figure 4.4: CNN baseline.

as an input with  $n$  words. Each word in the tweet is replaced with its dictionary index  $s \in \mathcal{R}^{1 \times n}$  and transformed into a vector. The tweet vector is padded and turned into a matrix  $s \in \mathcal{R}^{1 \times l}$  where  $l$  is the longest tweet in the input corpus, in order to handle varying input lengths. The embedding layer receives the input vector and feeds it through, converting each word into a distributional vector of dimension  $D$ . As a result, the input tweet matrix is transformed to  $s \in \mathcal{R}^{l \times D}$ .

By using convolutional filters and extracting distinguishing word sequences as a com-



posite feature map for the LSTM layer, a convolution network seeks to minimize frequency variation. Using a convolutional filter  $k \in \mathfrak{R}^{\mathcal{D} \times m}$ , the convolution operation converts the input matrix  $s \in \mathfrak{R}^{l \times \mathcal{D}}$  into  $c \in \mathfrak{R}^{l \times m+1}$ . These calculations are made for each component:

$$c_i = (s * k)_i = \sum_{k,j} (S_{i,j-m+1:j} \oplus F)_{k,j} \quad (4.3)$$

A feature map with the dimensions  $c \in \mathfrak{R}^{l \times m+1}$  is produced by a convolution filter with the same dimension  $D$  as the input matrix and sliding along the column dimension of the input matrix. This filter performs an element-wise product between a column slice  $s$  and a filter matrix  $k$ , producing a vector component  $c_i$ .  $C \in \mathfrak{R}^{l \times m+1}$  is a feature map produced by  $f$  filters. For non-linearity, the Sigmoid was chosen. The output of the convolutional network was first sent through a pooling layer, and sizes 2 and 3 are employed with max-pooling. Later, to increase the model's performance, the max-pooling layer was dropped and all of the composite characteristics for sarcasm detection were fed to the LSTM network. The architecture of the model is presented in Figure 4.5

Using feedback cycles in the network architecture, RNN has effectively illustrated the capability of semantic modeling. RNN networks have a temporal memory component that enables the model to directly store the temporal contextual data. It takes into account the hidden state  $h_{t-1}$  and the current input  $x_t$  at each time step. Thus, if the distance between two-time steps is too great, the RNN cannot draw long-term relationships. By defining each memory cell with a set of gates  $\mathfrak{R}^d$ , where  $d$  is the memory dimension of the hidden state of the LSTM, introduced LSTM, which can plot long-term dependencies and does not experience vanishing or erupting gradients while running backpropagation algorithm.

The input gate  $i_t$ , forget gate  $f_t$ , and output gate  $o_t$  are the three gates that make up an LSTM and are functions of  $x_t$  and  $h_{t-1}$ . The memory updating technique is decided collectively by the gates. The amount of information that should be discarded, stored from, and saved in memory are indicated by equations (3) and (2). The output of the cell  $c_t$  is indicated by equation (5).

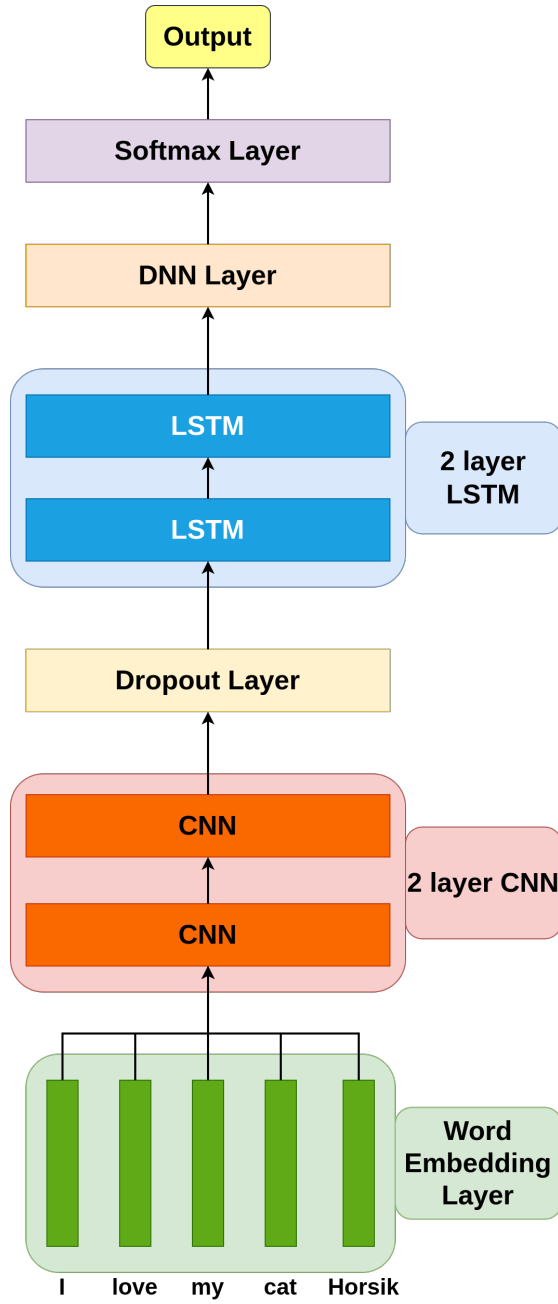


Figure 4.5: CNN-LSTM-DNN baseline

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (4.4)$$

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (4.5)$$

$$q_t = \tanh(W_q[h_{t-1}, x_t] + b_q) \quad (4.6)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (4.7)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot q_t \quad (4.8)$$

$$h_t = o_t \odot \tanh(c_t) \quad (4.9)$$

In order to create a higher level feature set that is easily separable for the necessary number of classes, the output of the LSTM layer is sent to a fully connected DNN layer. The DNN layer is then followed by the addition of a softmax layer. The binary cross-entropy error is minimized during network training. The ADAM optimizer was employed with a learning rate of 0.001 for parameter optimization.

#### 4.3.4 SAWS

SAWS (Pan et al., 2020) model proposes a self-attention mechanism of weighted snippets to model the incongruity between sentence snippets.

The model is constructed of five modules —an input module, a convolution module, a significance weighting module, a self-attention module, and an output module. The overview is shown in Fig. First, an encoding layer transforms the input words into low-dimension representations. The convolution module is then used to obtain representations of sentence fragments based on the encoded words. Following that, the context vector is utilized to compute the weights of the snippets in the importance weighting module; the significant snippets are given large weights, and insignificant snippets are given low weights. Finally, the weighted snippets are subjected to the self-attention module, and the output module generates an output for categorization. See the Figure 4.6

The input module is built as follows. The input text  $X$  is a sequence of words  $X \in R^{l*1}$ , where  $l$  is the total number of words in the text. In order to resolve diverse input lengths

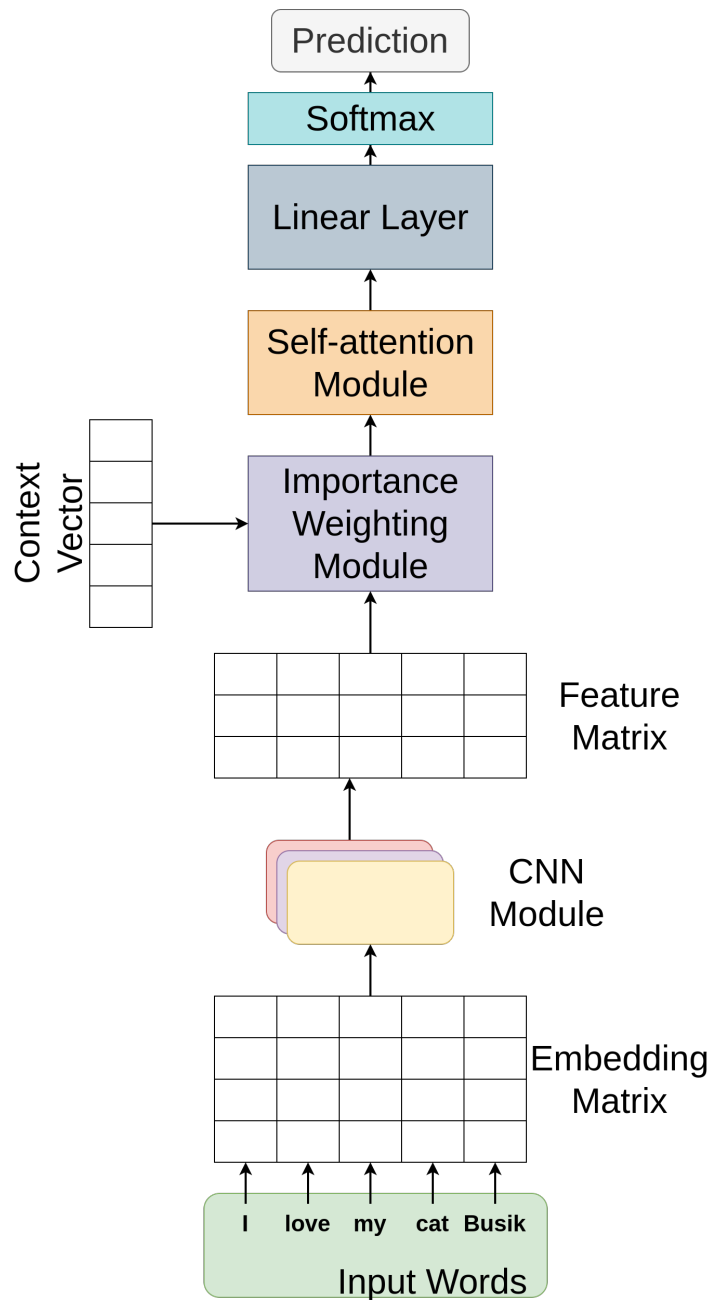


Figure 4.6: SAWS.

$X \in R^{l*1}$  is transformed to  $X' \in R^{n*1}$ , where  $n$  is a pre-defined hyper-parameter. Less than  $n$  token input sequences are padded to  $n$ , and more than  $n$  token input sequences are truncated to  $n$ . Each word is transformed into a low-dimensional vector representation (word embedding) in the input encoding layer using a weighted matrix. The 400,000 most frequent words were used in the publicly accessible GloVe vector (Pennington et al., 2014).

Random initialization is used for words that are not part of the list of pre-trained words. Consequently, the output  $E \in R^{n \times e}$  where  $e$  is the embedding size, is a sequence of word embeddings.

Convolutional neural network (CNN) was used for the first time in text categorization tasks by Kim (2014) and demonstrated excellent performance. By implementing a convolution operation between the convolution kernels and the words in a sequence, convolution layers can be used to capture contextual local information. A convolutional layer appears to be a reasonable option to encode local snippet information because we want to be able to gain representations of sentence snippets, which often consist of many consecutive words.

When applied to a window of  $m$  consecutive words of the input matrix  $E \in R^{n \times e}$ , a convolution filter  $K \in R^{m \times e}$  with the same dimension  $e$  as the input matrix performs element-wise product between the selected windows of  $E$  and filter  $k$  to produce a vector  $c \in R^{n-m+1}$ . A vector named  $c$  is made up of the elements  $c_1, c_2, c_3, \dots, c_{n-m+1}$ . The following formula is determines each  $c_i$ :

$$c_i = \sum E_{i:i+m-1, e} \otimes k_{0:m, e} \quad (4.10)$$

where  $\otimes$  indicates element-wise product. To obtain the snippet representation  $U \in R^{n-m+1} \times e$ , where  $U \in R^e$  is denoted for  $i$ -th snippet in the original input sequence, the same operation is performed using  $e$  filters.

The importance of weighting module to weight the sentence fragments during training and testing is established due to the observation that sentence fragments do not all equally help to the sarcasm detection task. In this module, the context vector  $v \in R^{n-m+1}$  is initialized randomly in an initial stage. After that,  $v$  used to determine each snippet's  $U_i$  attention score  $a_i$ . First,  $b \in R^{n-m+1}$  is added to a multiplication of snippet representation  $U - i \in R^e$  and  $W \in R^{n-m+1} \times e$ , which is then passed through tanh layer to obtain  $u_i$ . The significance of the snippet  $i$  is then determined by calculating the similarity between  $u_i$  and the context vector  $v$ . After that, the weights are normalized using the softmax

function to produce the weight distribution vector  $a$ . Vector  $a$  calculated as follows:

$$u_i = \tanh(W^t U_i + b) \quad (4.11)$$

$$a_i = \frac{\exp(u_i v)}{\sum \exp(u_i, v)} \quad (4.12)$$

$$a = (a_1, a_2, a_3, \dots, a_{n-m+1}) \quad (4.13)$$

where  $a \in R^{n-m+1}$ .

Finally, the snippets are connected to their corresponding attention levels.  $v$  can be thought of as a representation of the fixed query “how relevant this chunk is?”. They attention weights are computed using vector  $v$  at both word and sentence levels for document categorization, whereas attention weights are computed across snippets.  $V$  is learnt throughout the training process. Figure 4.7 illustrates this process.

In the self-attention block of the model, the contrast and incongruity between the weighted snippets  $P$  is simulated using the self-attention process. One can think of incongruity as a sarcastic text’s intrinsic trait. Thus, applying self-attention to the weighted phrases results in an output that contains the incongruity information. In order to minimize training loss, the incongruous sentences in particular give each other a high attention value. To create the self-attention matrix  $S$  in this module, the affinity score  $s$  computed between weighted snippets. The following formula is used to determine the affinity score  $s_{i,j}$  between sentence snippets  $P_i$  and  $p_j$ :

$$s_{i,j} = W([P_i; P_j]) + b \quad (4.14)$$

where  $i, j$  where the affinity score between each snippet pair  $P_i, P_j$  is represented by the scalar where  $i, j$  in the self-attention matrix  $S$ . The weighted snippets  $i$  and  $j$  are represented by  $P_i$  and  $P_j$ , respectively. The vector concatenation operation is denoted by  $[\cdot]$ . The affinity scores for each pair of sentence snippets creates the self-attention matrix  $S$ ,

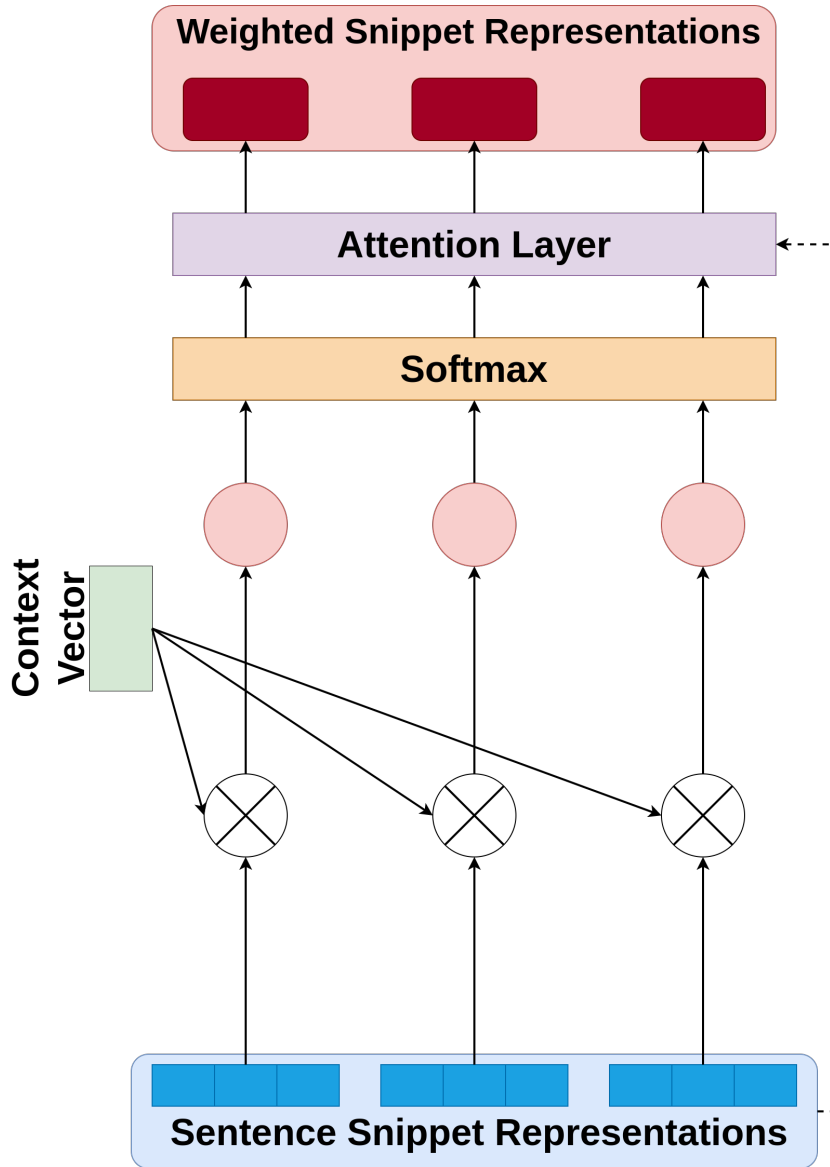


Figure 4.7: Importance weights.

which is represented as follows:

$$S = \begin{bmatrix} S_{1,1} & \dots & S_{1,n-m+1} \\ \vdots & \ddots & \vdots \\ S_{n-m+1} & \dots & S_{n-m+1,n-m+1} \end{bmatrix}$$

Then, by performing a max-pooling operation per each row in the self-attention matrix  $S$ , the attention vector  $a \in R^{n-m+1}$  is computed. In order to prevent affecting the out-

comes, the attention value of a sentence fragment has been masked. The attention vector  $a$  is calculated as follows:

$$a_i = \max(s_{i,1}, s_{i,2}, s_{i,3}, \dots, s_{i,n-m+1}) \quad (4.15)$$

$$a_i = \text{softmax}(a_1, a_2, a_3, \dots, a_{n-m+1}) \quad (4.16)$$

where  $a \in R^{n-m+1}$ . The weighted snippet representations  $P$  are used with the attention vector  $a$  after it has been obtained as:

$$f_a = \sum_{i=1}^{n-m+1} P_i a_i \quad (4.17)$$

where  $f_a \in R^e$  is the weighted snippets' self-attentive representation, which contains the incongruity features and is utilized to make predictions.

The self-attentive representation,  $f_a \in R^e$ , is the input of the output module. The prediction layer is built of linear and a Softmax layer. The linear decrease the dimension of  $f_a$ . Softmax layer classifies the output:

$$\hat{y} = \text{softmax}(W f_a + b) \quad (4.18)$$

Where  $W \in R^{e*2}$  and  $b \in R^2$  are the trainable parameters,  $\hat{y} \in R^2$  is the result of binary classification.

### 4.3.5 ELMo

This model (Ilić et al., 2018b) uses character-level vector representations of words, based on embeddings from ELMo (Peters et al., 2018) language model architecture. Subsequently, word embeddings are passed on to a BiLSTM, and the output hidden states are max-pooled and fed to the 2-layer feed-forward network. The output of this step is then fed to a final layer of decrease, which performs the binary classification.

Starting with a rather complex neural network language model that was greatly in-



fluenced by earlier work on large-scale language models, the ELMo architecture trains language models. The architecture describes as follows.

The language model used for ELMo starts with the following 2-layer bidirectional LSTM backbone as described in Figure4.8

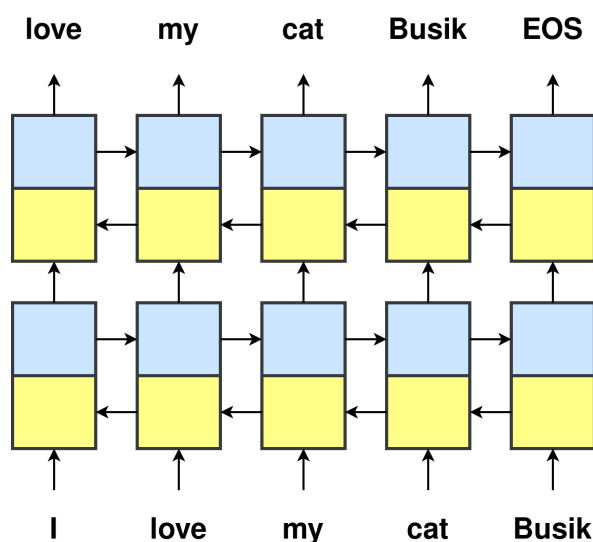


Figure 4.8: A 2-layer bidirectional LSTM. The forward recurrent unit is represented by the red box, and the backward recurrent unit is represented by the blue box.

Now, a remaining link is added between the first and second levels of this two-layer network. High-level intuition holds that the remaining connection contributes to the efficiency of deep model training. After then, the language model appears as follows in Figure4.9.

Each token in the first input layer (in this case, “I love my cat Busik”) is transformed into a fixed-length word embedding before being sent to the recurrent unit in conventional neural language models (NLMs). For each token, either used a pre-trained embedding like GLoVe or initialized a word embedding matrix of size (Vocabulary size) x (Word embedding dimension).

For the ELMo language model, however, it takes a little more involved approach. Character embeddings are initially used to transform each token into the right representation, as opposed to just looking for an embedding in a word embedding matrix. Then, using a certain number of filters, a convolutional layer is applied to this character embedding

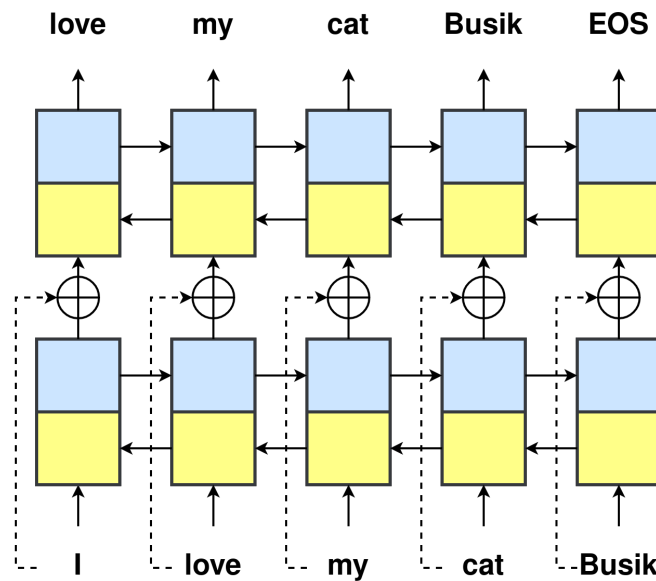


Figure 4.9: The additional connection between the first and second LSTM layers. Prior to being sent on as the input to the second layer, the input to the first layer is combined with its output.

representation, followed by a max-pool layer. This representation is then given as the input to the LSTM layer after passing through a two-layer network. The short summary of process is presented in Figure4.10.

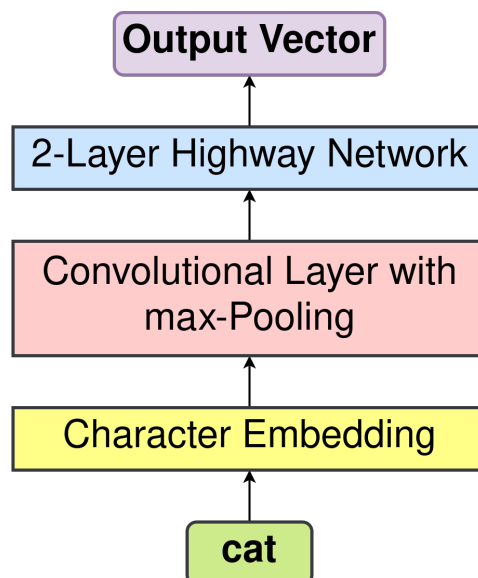


Figure 4.10: Each token is transformed prior to being entered into the first LSTM layer. For simplicity, only the initial token is represented.

There are several benefits to these adjustments made to the input token. First of all, by employing character embeddings, morphological elements may be detected, those that word-level embeddings could overlook. Additionally, character embeddings guarantee that a reliable representation can be created even for words that are not commonly used, which is a major accomplishment.

Then, by employing convolutional filters, n-gram features can be founded that help creates stronger representations. The input can convey information more smoothly thanks to the highway network layers.

The use of the language model after it has been trained is where ELMo makes significant advancements. Let's assume that the input words are  $k^{th}$ . Taking the word representation  $x_k$ , the bidirectional hidden layer representations  $h_{1,k}$  and  $h_{2,k}$ , the model merges them into a new weighted task representation using a trained 2-layer language model. This architecture is represented in Figure4.11.

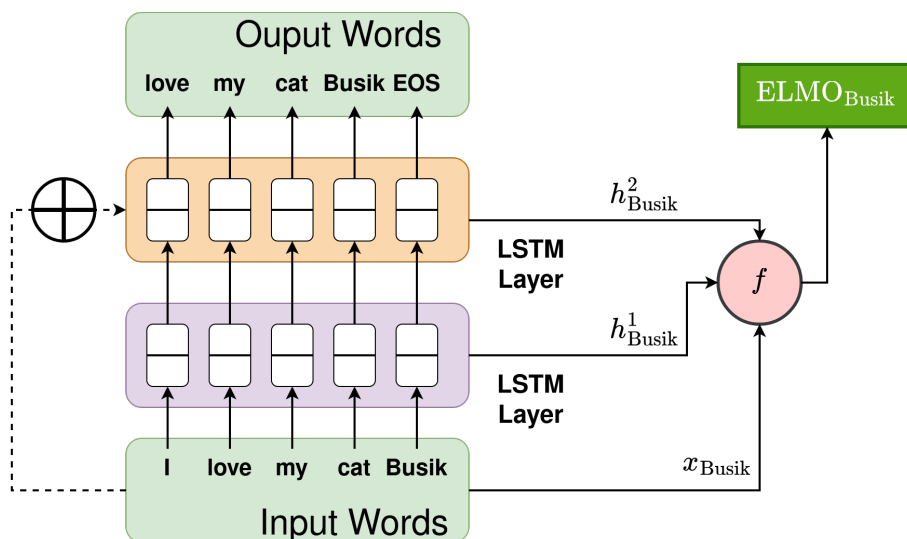


Figure 4.11: An illustration of integrating the word representation for “Busik” with the bidirectional hidden representations to get an ELMo-specific representation.

More specifically, the function  $f$  operates on the input word  $k$  as follows:

$$ELMo_k^t = b_k * (c_0^t * d_k + c_1^t h_{1,k} + c_2^t * h_{2,k}) \quad (4.19)$$

Here,  $b_k$  stands for a weighting factor specific for a certain task.  $c_i$  represents softmax-

normalized weights on the language model’s hidden representations.

For each task (question answering, sentiment analysis, etc.), a unique ELMo representation is learned. To apply ELMo in a task, firstly the the weights of the trained language model have to be freezed. The *ELMo<sub>k</sub>t* must then be concatenated to the input representation of each task-specific model for each token. We then freeze the weights of the trained language model. The training of the task-specific model is then used to learn the weighting parameters  $b_k$  and  $c_k$ . That is basically how ELMo operates. A straightforward concept with enormous power.

A few specifics about the training and application of the ELMo model should be mentioned.

First off, the 1B Word Benchmark dataset is used to train the ELMo language model. The LSTM layers of the language model comprise 4096 units, while the input embedding transform makes use of 2048 convolutional filters.

Another important point is that, where appropriate, fine-tuning the language model on task-specific data resulted in decreased confusion and improved performance on downstream tasks. This outcome is significant because it supports the impact of domain transfer in neural models.

## 4.4 Experiments

I implemented our model using Pytorch Lightning (Falcon & others, 2019) and transformers library (Wolf et al., 2020). For each dataset, we experimented with different sets of hyperparameters. The best-performing hyperparameters are presented in Table 4.3.

Parameter	Datasets			
	SARC/movies	SARC/tech	IAC_V2	Twitter (Ptacek are et al., 2014)
max_length	18	14	16	16
max_epochs	12	30	20	20
lr	1e(-5)	1e(-5)	1e(-5)	1e(-5)
batch_size	multicolumn1c 8	4	32	32

Table 4.3: Best performing hyperparameters for various datasets.

# Chapter 5

## Conclusion and Future Work

### 5.1 Results and Discussion

I used precision, recall, macro-weighted F1, and accuracy scores as the evaluation metrics. I emphasized F1 score as the main metric in my analysis. Table 5.1 reports the results of proposed method as well as baselines.

For each dataset, I trained each baseline model and my model. I observed that the basic models (i.e., NBOW, CNN, and CNN-LSTM-DNN) perform poorly compared to other models. Those models simply encode input text to capture local semantic information, which is insufficient to derive global context or recognize incongruity between words.

However, out of three above-mentioned models, CNN showed higher results on SARC/movies and SARC/technology subreddit. One of the reasons leading to such a result could be the relative shortness of texts in those datasets. The statistics are presented in Table 5.2. Particularly, median and mean for SARC/movies dataset ( $Med = 10, \mu_i = 12.24$ ) are significantly lower than median and mean for IAC-V2 dataset respectively ( $Med = 39, \mu_i = 17.61$ ).

SAWS and ELMo models outperformed basic models. Compared to the previous baselines, SAWS and ELMo models are built to capture more sophisticated patterns, such as text fragments incongruity and complex morpho-syntactic features. However, the ELMo model is almost always a few points ahead of SAWS, showing that purely character-based input and subsequently obtained contextual embeddings capture more useful sarcastic

Datasets	Model	Precision	Recall	Acc.	F1
SARC/movies	NBOW	0.60	0.60	0.60	0.60
	CNN	0.65	0.65	0.64	0.65
	CNN-LSTM-DNN	0.58	0.58	0.59	0.58
	SAWS	0.65	0.65	0.65	0.65
	ELMo	0.72	0.57	0.68	0.64
	Our Model	0.72	0.74	0.73	<b>0.73</b>
SARC/technology	NBOW the	0.63	0.64	0.64	0.64
	CNN	0.67	0.67	0.66	0.66
	CNN-LSTM-DNN	0.65	0.58	0.56	0.61
	SAWS	0.65	0.65	0.66	0.65
	ELMo	0.76	0.81	0.73	0.78
	Our Model	0.77	0.84	0.75	<b>0.80</b>
IAC-V2	NBOW	0.72	0.71	0.72	0.71
	CNN	0.71	0.70	0.70	0.70
	CNN-LSTM-DNN	0.63	0.75	0.65	0.68
	SAWS	0.75	0.75	0.75	0.75
	ELMo	0.78	0.74	0.77	0.76
	Our Model	0.86	0.84	0.85	<b>0.85</b>
Twitter	NBOW	0.75	0.75	0.75	0.75
	CNN	0.80	0.80	0.80	0.80
	CNN-LSTM-DNN	0.77	0.82	0.80	0.80
	SAWS	0.81	0.81	0.81	0.81
	ELMo	0.83	0.86	0.85	0.84
	Our Model	0.93	0.94	0.94	<b>0.93</b>

Table 5.1: Results of the experiments.

Dataets	Metrics				
	Median	Mean	Variance	Max	Min
SARC/movies	10	12.24	8.81	138	1
SARC/technology	12	13.88	9.33	103	1
IAC-V2	39	50.63	36.05	212	10
Twitter	17	17.61	6.26	64	1

Table 5.2: Statistics of the datasets.

information.

For all datasets, my model outperformed all baselines for all metrics and achieved state-of-the-art performance. The best improvement of the F1 metric was achieved on IAC-V2 and Twitter datasets. Specifically, the F1 metric is 9% higher, compared with the previous state-of-the-art version. For SARC/movies dataset, F1 metric is improved by 8%, and for

SARC/technology dataset F1 metric is improved by 2%.

I used different transformer models fitted on emotion and sentiment detection tasks to obtain contextualized and more deep features. This allows the model to learn complex patterns from different perspectives, e.g. “emotional” and “sentiment”.

Furthermore, the CNN block of our model utilizes Glove embeddings in order to capture semantic relationships of words and general context. My model shows that modeling dependencies between emotion, sentiment, and sarcasm is an important feature for sarcasm detection task.

Interestingly, all models show their best results on Twitter dataset, and their performance decrease when the length of the input text is relatively long (IAC-V2) or short (SARC/movies). It suggests that more ideas should be investigated for texts of a particular length.

## 5.2 Conclusion and Future Work

In this paper, I introduced a novel sarcasm detection method, which incorporates both emotion and sentiment features. Proposed approach contains four components: Sarcasm Pre-Trained Transformer (SarcPTT), Emotion Detection Pre-Trained Transformer (datasets), Sentiment Detection Pre-Trained Transformer (SentDPTT), and CNN block.

EmoDPTT and SentDPTT are used as feature extractors and are not trainable during model fitting. These models are used to highlight the parts of the sentence which provide crucial cues for sarcasm detection. CNN module extracts semantic relationship of words and general context features. SarcPTT is pre-trained on train chunks of datasets, so the model learns sarcasm patterns. The output from each block is passed through a fully-connected network and then through a linear layer to get the final classification score.

Experiments are conducted on four datasets: SARC/movies, SARC/technology, IAC-V2, and Twitter. They show significant improvement over the state-of-the-art models using all evaluation metrics. Results show that emotion and sentiment features along with contextual knowledge play a crucial role in the performance of sarcasm detection.

Even though my model outperformed all the baselines, all the methods are struggling

with too short or too long texts. In future works, I am planning to address this challenge.

### 5.3 Scientific Contribution

The present study attempts to address the task of sarcasm detection and in doing so makes important contributions.


First, the study extends the research on related literature and analyses the recent trends in it. Second, novel architecture was proposed, based on a number of different features the model extracts from a text, in particular emotion, sentiment, and context features. Third, the proposed model outperformed existing state-of-the-art results and produces highest scores in sarcasm detection for short texts task.



# References

- Abercrombie, G., & Hovy, D. (2016). Putting sarcasm detection into context: The effects of class imbalance and manual labelling on supervised machine classification of twitter conversations. In *Proceedings of the acl 2016 student research workshop* (pp. 107–113).
- Abulaish, M., & Kamal, A. (2018). Self-deprecating sarcasm detection: an amalgamation of rule-based and machine learning approach. In *2018 ieee/wic/acm international conference on web intelligence (wi)* (pp. 574–579).
- Agrawal, A., & An, A. (2018). Affective representations for sarcasm detection. In *The 41st international acm sigir conference on research & development in information retrieval* (pp. 1029–1032).
- Akula, R., & Garibay, I. (2021). Explainable detection of sarcasm in social media. In *Proceedings of the eleventh workshop on computational approaches to subjectivity, sentiment and social media analysis* (pp. 34–39).
- Alberti, C., Ling, J., Collins, M., & Reitter, D. (2019). Fusion of detected objects in text for visual question answering. *arXiv preprint arXiv:1908.05054*.
- Amir, S., Wallace, B. C., Lyu, H., & Silva, P. C. M. J. (2016). Modelling context with user embeddings for sarcasm detection in social media. *arXiv preprint arXiv:1607.00976*.
- Avvaru, A., Vobilisetty, S., & Mamidi, R. (2020). Detecting sarcasm in conversation context using transformer-based models. In *Proceedings of the second workshop on figurative language processing* (pp. 98–103).
- Baccianella, S., Esuli, A., & Sebastiani, F. (2010). Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the seventh international conference on language resources and evaluation (lrec'10)*.

- Bamman, D., & Smith, N. (2015). Contextualized sarcasm detection on twitter. In *proceedings of the international aaai conference on web and social media* (Vol. 9, pp. 574–577).
- Banerjee, A., Bhattacharjee, M., Ghosh, K., & Chatterjee, S. (2020). Synthetic minority oversampling in addressing imbalanced sarcasm detection in social media. *Multimedia Tools and Applications*, 79(47), 35995–36031.
- Barbieri, F., Saggion, H., & Ronzano, F. (2014). Modelling sarcasm in twitter, a novel approach. In *proceedings of the 5th workshop on computational approaches to subjectivity, sentiment and social media analysis* (pp. 50–58).
- Bharti, S. K., Babu, K. S., & Jena, S. K. (2015). Parsing-based sarcasm sentiment recognition in twitter data. In *2015 ieee/acm international conference on advances in social networks analysis and mining (asonam)* (pp. 1373–1380).
- Bharti, S. K., Pradhan, R., Babu, K. S., & Jena, S. K. (2017a). Sarcasm analysis on twitter data using machine learning approaches. *Trends in Social Network Analysis*, 51–76.
- Bharti, S. K., Pradhan, R., Babu, K. S., & Jena, S. K. (2017b). Sarcastic sentiment detection based on types of sarcasm occurring in twitter data. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 13(4), 89–108.
- Bharti, S. K., Vachha, B., Pradhan, R., Babu, K. S., & Jena, S. K. (2016). Sarcastic sentiment detection in tweets streamed in real time: a big data approach. *Digital Communications and Networks*, 2(3), 108–121.
- Borth, D., Chen, T., Ji, R., & Chang, S.-F. (2013). Sentibank: large-scale ontology and classifiers for detecting sentiment and emotions in visual content. In *Proceedings of the 21st acm international conference on multimedia* (pp. 459–460).
- Bosselut, A., Rashkin, H., Sap, M., Malaviya, C., Celikyilmaz, A., & Choi, Y. (2019). Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*.
- Bouazizi, M., & Ohtsuki, T. O. (2016). A pattern-based approach for sarcasm detection on twitter. *IEEE Access*, 4, 5477–5488.
- Cai, Y., Cai, H., & Wan, X. (2019). Multi-modal sarcasm detection in twitter with

- hierarchical fusion model. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 2506–2515).
- Chakrabarty, T., Ghosh, D., Muresan, S., & Peng, N. (2020).  Reverse, retrieve, and rank for sarcasm generation with commonsense knowledge. *arXiv preprint arXiv:2004.13248*.
- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE), 2493–2537.
- Cowen, A., Sauter, D., Tracy, J. L., & Keltner, D. (2019). Mapping the passions: Toward a high-dimensional taxonomy of emotional experience and expression. *Psychological Science in the Public Interest*, 20(1), 69–90.
- Cowen, A. S., & Keltner, D. (2017). Self-report captures 27 distinct categories of emotion bridged by continuous gradients. *Proceedings of the national academy of sciences*, 114(38), E7900–E7909.
- Das, D., & Clark, A. J. (2018). Sarcasm detection on facebook: A supervised learning approach. In *Proceedings of the 20th international conference on multimodal interaction: Adjunct* (pp. 1–5).
- Davidov, D., Tsur, O., & Rappoport, A. (2010). Semi-supervised recognition of sarcasm in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning* (pp. 107–116).
- Demszky, D., Movshovitz-Attias, D., Ko, J., Cowen, A., Nemade, G., & Ravi, S. (2020). GoEmotions: A Dataset of Fine-Grained Emotions. In *58th annual meeting of the association for computational linguistics (acl)*.
- Dharwal, P., Choudhury, T., Mittal, R., & Kumar, P. (2017). Automatic sarcasm detection using feature selection. In *2017 3rd international conference on applied and theoretical computing and communication technology (icatcct)* (pp. 29–34).
- Diao, Y., Lin, H., Yang, L., Fan, X., Chu, Y., Xu, K., & Wu, D. (2020). A multi-dimension question answering network for sarcasm detection. *IEEE Access*, 8, 135152–135161.
- Di Gangi, M. A., Bosco, G. L., & Pilato, G. (2019). Effectiveness of data-driven induction of semantic spaces and traditional classifiers for sarcasm detection. *Natural Language*

- Engineering*, 25(2), 257–285.
- Dimovska, J., Angelovska, M., Gjorgjevikj, D., & Madjarov, G. (2018). Sarcasm and irony detection in english tweets. In *International conference on telecommunications* (pp. 120–131).
- Dong, X., Li, C., & Choi, J. D. (2020). Transformer-based context-aware sarcasm detection in conversation threads from social media. *arXiv preprint arXiv:2005.11424*.
- Dubey, A., Joshi, A., & Bhattacharyya, P. (2019). Deep models for converting sarcastic utterances into their non sarcastic interpretation. In *Proceedings of the acm india joint international conference on data science and management of data* (pp. 289–292).
- Dubey, A., Kumar, L., Somani, A., Joshi, A., & Bhattacharyya, P. (2019). “when numbers matter!”: Detecting sarcasm in numerical portions of text. In *Proceedings of the tenth workshop on computational approaches to subjectivity, sentiment and social media analysis* (pp. 72–80).
- Eisner, B., Rocktäschel, T., Augenstein, I., Bošnjak, M., & Riedel, S. (2016). emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.
- Eke, C. I., Norman, A., Shuib, L., Fatokun, F. B., & Omame, I. (2020). The significance of global vectors representation in sarcasm analysis. In *2020 international conference in mathematics, computer engineering and computer science (icmcecs)* (pp. 1–7).
- Falcon, W., & others. (2019). Pytorch lightning. *GitHub*. Note: <https://github.com/PyTorchLightning/pytorch-lightning>, 3(6).
- Fan, A., Lavril, T., Grave, E., Joulin, A., & Sukhbaatar, S. (2020). Addressing some limitations of transformers with feedback memory. *arXiv preprint arXiv:2002.09402*.
- Felbo, B., Mislove, A., Søgaard, A., Rahwan, I., & Lehmann, S. (2017). Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.
- Filatova, E. (2012). Irony and sarcasm: Corpus generation and analysis using crowdsourcing. In *Lrec* (pp. 392–398).
- Gao, Z., Feng, A., Song, X., & Wu, X. (2019). Target-dependent sentiment classification

- with bert. *Ieee Access*, 7, 154290–154299.
- Ghosh, A., & Veale, T. (2016). Fracking sarcasm using neural network. In *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis* (pp. 161–169).
- Ghosh, A., & Veale, T. (2017). Magnets for sarcasm: Making sarcasm detection timely, contextual and very personal. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 482–491).
- Ghosh, D., Fabbri, A. R., & Muresan, S. (2017). The role of conversation context for sarcasm detection in online interactions. *arXiv preprint arXiv:1707.06226*.
- Ghosh, D., Fabbri, A. R., & Muresan, S. (2018). Sarcasm analysis using conversation context. *Computational Linguistics*, 44(4), 755–792.
- Ghosh, D., Guo, W., & Muresan, S. (2015). Sarcastic or not: Word embeddings to predict the literal or sarcastic meaning of words. In *proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 1003–1012).
- González-Ibáñez, R., Muresan, S., & Wacholder, N. (2011). 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).
- Gregory, H., Li, S., Mohammadi, P., Tarn, N., Draelos, R., & Rudin, C. (2020). A transformer approach to contextual sarcasm detection in twitter. In *Proceedings of the second workshop on figurative language processing* (pp. 270–275).
- Guo, N., & Shah, R. (n.d.). Finding sarcasm in reddit postings: A deep learning approach.
- Gupta, R., Kumar, J., Agrawal, H., et al. (2020). A statistical approach for sarcasm detection using twitter data. In *2020 4th international conference on intelligent computing and control systems (iciccs)* (pp. 633–638).
- Hartmann, J., Heitmann, M., Siebert, C., & Schamp, C. (2022). More than a feeling: Accuracy and application of sentiment analysis. *International Journal of Research in Marketing*.
- Hazarika, D., Poria, S., Gorantla, S., Cambria, E., Zimmermann, R., & Mihalcea, R. (2018). Cascade: Contextual sarcasm detection in online discussion forums. *arXiv preprint arXiv:1805.06413*.

- Ilić, S., Marrese-Taylor, E., Balazs, J., & Matsuo, Y. (2018b, October). Deep contextualized word representations for detecting sarcasm and irony. In *Proceedings of the 9th workshop on computational approaches to subjectivity, sentiment and social media analysis* (pp. 2–7). Brussels, Belgium: Association for Computational Linguistics. Retrieved from <https://aclanthology.org/W18-6202> doi: 10.18653/v1/W18-6202
- Ilić, S., Marrese-Taylor, E., Balazs, J. A., & Matsuo, Y. (2018a). Deep contextualized word representations for detecting sarcasm and irony. *arXiv preprint arXiv:1809.09795*.
- Ivanko, S. L., & Pexman, P. M. (2003). Context incongruity and irony processing. *Discourse processes*, 35(3), 241–279.
- Jaech, A., Zayats, V., Fang, H., Ostendorf, M., & Hajishirzi, H. (2015). Talking to the crowd: What do people react to in online discussions? *arXiv preprint arXiv:1507.02205*.
- Jain, T., Agrawal, N., Goyal, G., & Aggrawal, N. (2017). Sarcasm detection of tweets: A comparative study. In *2017 tenth international conference on contemporary computing (ic3)* (pp. 1–6).
- Jamil, R., Ashraf, I., Rustam, F., Saad, E., Mehmood, A., & Choi, G. S. (2021). Detecting sarcasm in multi-domain datasets using convolutional neural networks and long short term memory network model. *PeerJ Computer Science*, 7, e645.
- Jansi, K., Sajja, P., & Goyal, P. (2018). An extensive survey on sarcasm detection using various classifiers. *International Journal of Pure and Applied Mathematics*, 119(12), 13183–13186.
- Javdan, S., Minaei-Bidgoli, B., et al. (2020). Applying transformers and aspect-based sentiment analysis approaches on sarcasm detection. In *Proceedings of the second workshop on figurative language processing* (pp. 67–71).
- Joshi, A., Bhattacharyya, P., Carman, M., Saraswati, J., & Shukla, R. (2016). How do cultural differences impact the quality of sarcasm annotation?: A case study of indian annotators and american text. In *Proceedings of the 10th sighthum workshop on language technology for cultural heritage, social sciences, and humanities* (pp. 95–99).

- Joshi, A., Goel, P., Bhattacharyya, P., & Carman, M. (2016). Automatic identification of sarcasm target: An introductory approach. *arXiv preprint arXiv:1610.07091*.
- Joshi, A., Jain, P., Bhattacharyya, P., & Carman, M. (2016). Who would have thought of that!?: A hierarchical topic model for extraction of sarcasm-prevalent topics and sarcasm detection. *arXiv preprint arXiv:1611.04326*.
- Joshi, A., Sharma, V., & Bhattacharyya, P. (2015). Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 2: Short papers)* (pp. 757–762).
- Joshi, A., Tripathi, V., Patel, K., Bhattacharyya, P., & Carman, M. (2016). Are word embedding-based features useful for sarcasm detection? *arXiv preprint arXiv:1610.00883*.
- Kalaivani, A., & Thenmozhi, D. (2020). Sarcasm identification and detection in conversation context using bert. In *Proceedings of the second workshop on figurative language processing* (pp. 72–76).
- Kamal, A., & Abulaish, M. (2019). An lstm-based deep learning approach for detecting self-deprecating sarcasm in textual data. In *Proceedings of the 16th international conference on natural language processing* (pp. 201–210).
- Kannangara, S. (2018). Mining twitter for fine-grained political opinion polarity classification, ideology detection and sarcasm detection. In *Proceedings of the eleventh acm international conference on web search and data mining* (pp. 751–752).
- Khatri, A., et al. (2020). Sarcasm detection in tweets with bert and glove embeddings. *arXiv preprint arXiv:2006.11512*.
- Khodak, M., Saunshi, N., & Vodrahalli, K. (2018, May). A large self-annotated corpus for sarcasm. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA). Retrieved from <https://aclanthology.org/L18-1102>
- Khokhlova, M., Patti, V., & Rosso, P. (2016). Distinguishing between irony and sarcasm in social media texts: Linguistic observations. In *2016 international fruct conference*

- on intelligence, social media and web (ismw fruct)* (pp. 1–6).
- Khotijah, S., Tirtawangsa, J., & Suryani, A. A. (2020). Using lstm for context based approach of sarcasm detection in twitter. In *Proceedings of the 11th international conference on advances in information technology* (pp. 1–7).
- Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing, EMNLP 2014, october 25-29, 2014, doha, qatar, A meeting of sigdat, a special interest group of the ACL* (pp. 1746–1751). Retrieved from <http://aclweb.org/anthology/D/D14/D14-1181.pdf>
- Kolchinski, Y. A., & Potts, C. (2018). Representing social media users for sarcasm detection. *arXiv preprint arXiv:1808.08470*.
- Kreuz, R., & Caucci, G. (2007). Lexical influences on the perception of sarcasm. In *Proceedings of the workshop on computational approaches to figurative language* (pp. 1–4).
- Kumar, A., Narapareddy, V. T., Gupta, P., Srikanth, V. A., Neti, L. B. M., & Malapati, A. (2021). Adversarial and auxiliary features-aware bert for sarcasm detection. In *8th acm ikdd cods and 26th comad* (pp. 163–170).
- Kumar, A., Narapareddy, V. T., Srikanth, V. A., Malapati, A., & Neti, L. B. M. (2020). Sarcasm detection using multi-head attention based bidirectional lstm. *Ieee Access*, 8, 6388–6397.
- Kumar, A., Sangwan, S. R., Arora, A., Nayyar, A., Abdel-Basset, M., et al. (2019). Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network. *IEEE access*, 7, 23319–23328.
- Kumar, L., Somani, A., & Bhattacharyya, P. (2017). "having 2 hours to write a paper is fun!": Detecting sarcasm in numerical portions of text. *arXiv preprint arXiv:1709.01950*.
- Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Lee, H., Yu, Y., & Kim, G. (2020). Augmenting data for sarcasm detection with unlabeled



- conversation context. *arXiv preprint arXiv:2006.06259*.
- Lemmens, J., Burtenshaw, B., Lotfi, E., Markov, I., & Daelemans, W. (2020). Sarcasm detection using an ensemble approach. In *proceedings of the second workshop on figurative language processing* (pp. 264–269).
- Lin, R., Xiao, J., & Fan, J. (2018). Nextvlad: An efficient neural network to aggregate frame-level features for large-scale video classification. In *Proceedings of the european conference on computer vision (eccv) workshops* (pp. 0–0).
- Ling, J., & Klinger, R. (2016). An empirical, quantitative analysis of the differences between sarcasm and irony. In *European semantic web conference* (pp. 203–216).
- Liu, B., et al. (2010). Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2(2010), 627–666.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., . . . Stoyanov, V. (2019). RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Liu, Y., Wang, Y., Sun, A., Zhang, Z., Guo, J., & Meng, X. (2021). A dual-channel framework for sarcasm recognition by detecting sentiment conflict. *arXiv preprint arXiv:2109.03587*.
- Lou, C., Liang, B., Gui, L., He, Y., Dang, Y., & Xu, R. (2021). Affective dependency graph for sarcasm detection. In *Proceedings of the 44th international acm sigir conference on research and development in information retrieval* (pp. 1844–1849).
- Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). Vilbert: Pretraining task-agnostic vision-linguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32.
- Lu, J., Goswami, V., Rohrbach, M., Parikh, D., & Lee, S. (2020). 12-in-1: Multi-task vision and language representation learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 10437–10446).
- Majumder, N., Poria, S., Peng, H., Chhaya, N., Cambria, E., & Gelbukh, A. (2019). Sentiment and sarcasm classification with multitask learning. *IEEE Intelligent Systems*, 34(3), 38–43.
- Manning, C., Raghavan, P., & Schütze, H. (2008). Term weighting, and the vector space

- model. *Introduction to information retrieval*, 109–133.
- Maynard, D. G., & Greenwood, M. A. (2014). Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *Lrec 2014 proceedings*.
- Mehndiratta, P., & Soni, D. (2019). Identification of sarcasm using word embeddings and hyperparameters tuning. *Journal of Discrete Mathematical Sciences and Cryptography*, 22(4), 465–489.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Misra, R., & Arora, P. (2019). Sarcasm detection using hybrid neural network. *arXiv preprint arXiv:1908.07414*.
- Mukherjee, S., & Bala, P. K. (2017a). Detecting sarcasm in customer tweets: an nlp based approach. *Industrial Management & Data Systems*.
- Mukherjee, S., & Bala, P. K. (2017b). Sarcasm detection in microblogs using naïve bayes and fuzzy clustering. *Technology in Society*, 48, 19–27.
- Nakassis, C., & Snedeker, J. (2002). Beyond sarcasm: Intonation and context as relational cues in children’s recognition of irony. In *Proceedings of the twenty-sixth boston university conference on language development. cascadilla press, somerville, ma* (pp. 429–440).
- Naseem, U., Razzak, I., Eklund, P., & Musial, K. (2020). Towards improved deep contextual embedding for the identification of irony and sarcasm. In *2020 international joint conference on neural networks (ijcnn)* (pp. 1–7).
- Onan, A. (2019). Topic-enriched word embeddings for sarcasm identification. In *Computer science on-line conference* (pp. 293–304).
- Oprea, S., & Magdy, W. (2019). Exploring author context for detecting intended vs perceived sarcasm. *arXiv preprint arXiv:1910.11932*.
- Oraby, S., Harrison, V., Reed, L., Hernandez, E., Riloff, E., & Walker, M. (2017). Creating and characterizing a diverse corpus of sarcasm in dialogue. *arXiv preprint arXiv:1709.05404*.
- Pan, H., Lin, Z., Fu, P., & Wang, W. (2020). Modeling the incongruity between sentence snippets for sarcasm detection. In *Ecai 2020* (pp. 2132–2139). IOS Press.

- Pandey, R., Kumar, A., Singh, J. P., & Tripathi, S. (2021). Hybrid attention-based long short-term memory network for sarcasm identification. *Applied Soft Computing*, *106*, 107348.
- Parameswaran, P., Trotman, A., Liesaputra, V., & Eysers, D. (2021). Bert's the word: Sarcasm target detection using bert. In *Proceedings of the the 19th annual workshop of the australasian language technology association* (pp. 185–191).
- Parde, N., & Nielsen, R. (2018). Detecting sarcasm is extremely easy;- . In *Proceedings of the workshop on computational semantics beyond events and roles* (pp. 21–26).
- Peng, W., Adikari, A., Alahakoon, D., & Gero, J. (2019). Discovering the influence of sarcasm in social media responses. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *9*(6), e1331.
- Pennebaker, J. W., Francis, M. E., & Booth, R. J. (2001). Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, *71*(2001), 2001.
- Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1532–1543).
- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018, June). Deep contextualized word representations. In *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)* (pp. 2227–2237). New Orleans, Louisiana: Association for Computational Linguistics. Retrieved from <https://aclanthology.org/N18-1202> doi: 10.18653/v1/N18-1202
- Plepi, J., & Flek, L. (2021). Perceived and intended sarcasm detection with graph attention networks. *arXiv preprint arXiv:2110.04001*.
- Poria, S., Cambria, E., Hazarika, D., & Vij, P. (2016). A deeper look into sarcastic tweets using deep convolutional neural networks. *arXiv preprint arXiv:1610.08815*.
- Porwal, S., Ostwal, G., Phadtare, A., Pandey, M., & Marathe, M. V. (2018). Sarcasm detection using recurrent neural network. In *2018 second international conference on intelligent computing and control systems (iciccs)* (pp. 746–748).
- Potamias, R.-A., Siolas, G., & Stafylopatis, A. (2019). A robust deep ensemble classifier for

- figurative language detection. In *International conference on engineering applications of neural networks* (pp. 164–175).
- Potamias, R. A., Siolas, G., & Stafylopatis, A.-G. (2020). A transformer-based approach to irony and sarcasm detection. *Neural Computing and Applications*, *32*(23), 17309–17320.
- Prasad, A. G., Sanjana, S., Bhat, S. M., & Harish, B. (2017). Sentiment analysis for sarcasm detection on streaming short text data. In *2017 2nd international conference on knowledge engineering and applications (icke)* (pp. 1–5).
- Ptáček, T., Habernal, I., & Hong, J. (2014, August). Sarcasm detection on Czech and English Twitter. In *Proceedings of COLING 2014, the 25th international conference on computational linguistics: Technical papers* (pp. 213–223). Dublin, Ireland: Dublin City University and Association for Computational Linguistics. Retrieved from <https://aclanthology.org/C14-1022>
- Rajadesingan, A., Zafarani, R., & Liu, H. (2015). Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the eighth acm international conference on web search and data mining* (pp. 97–106).
- Rakov, R., & Rosenberg, A. (2013). "sure, i did the right thing": a system for sarcasm detection in speech. In *Interspeech* (pp. 842–846).
- Ren, Y., Ji, D., & Ren, H. (2018). Context-augmented convolutional neural networks for twitter sarcasm detection. *Neurocomputing*, *308*, 1–7.
- Rendalkar, S., & Chandankhede, C. (2018). Sarcasm detection of online comments using emotion detection. In *2018 international conference on inventive research in computing applications (icirca)* (pp. 1244–1249).
- Reyes, A., & Saldívar, R. (2022). Linguistic-based approach for recognizing implicit language in hate speech: Exploratory insights. *Computación y Sistemas*, *26*(1).
- Riloff, E., Qadir, A., Surve, P., De Silva, L., Gilbert, N., & Huang, R. (2013). Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 704–714).
- Salim, S. S., Ghanshyam, A. N., Ashok, D. M., Mazahir, D. B., & Thakare, B. S. (2020). Deep lstm-rnn with word embedding for sarcasm detection on twitter. In *2020*

- international conference for emerging technology (incet)* (pp. 1–4).
- Sarsam, S. M. (2019). Reinforcing the decision-making process in chemometrics: Feature selection and algorithm optimization. In *Proceedings of the 2019 8th international conference on software and computer applications* (pp. 11–16).
- Savini, E., & Caragea, C. (2020). A multi-task learning approach to sarcasm detection (student abstract). In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 13907–13908).
- Schifanella, R., De Juan, P., Tetreault, J., & Cao, L. (2016). Detecting sarcasm in multimodal social platforms. In *Proceedings of the 24th acm international conference on multimedia* (pp. 1136–1145).
- Sheikh, I., Illina, I., Fohr, D., & Linarès, G. (2016, August). Learning word importance with the neural bag-of-words model. In *Proceedings of the 1st workshop on representation learning for NLP* (pp. 222–229). Berlin, Germany: Association for Computational Linguistics. Retrieved from <https://aclanthology.org/W16-1626> doi: 10.18653/v1/W16-1626
- Shmueli, B., Ku, L.-W., & Ray, S. (2020). Reactive supervision: A new method for collecting sarcasm data. *arXiv preprint arXiv:2009.13080*.
- Song, Y., Wang, J., Jiang, T., Liu, Z., & Rao, Y. (2019). Attentional encoder network for targeted sentiment classification. *arXiv preprint arXiv:1902.09314*.
- Sreelakshmi, K., & Rafeeqe, P. (2018). An effective approach for detection of sarcasm in tweets. In *2018 international cet conference on control, communication, and computing (ic4)* (pp. 377–382).
- Sridhar, D., Foulds, J., Huang, B., Getoor, L., & Walker, M. (2015, July). Joint models of disagreement and stance in online debate. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)* (pp. 116–125). Beijing, China: Association for Computational Linguistics. Retrieved from <https://aclanthology.org/P15-1012> doi: 10.3115/v1/P15-1012
- Srivastava, H., Varshney, V., Kumari, S., & Srivastava, S. (2020). A novel hierarchical bert architecture for sarcasm detection. In *Proceedings of the second workshop on*

*figurative language processing* (pp. 93–97).

- Subramanian, J., Sridharan, V., Shu, K., & Liu, H. (2019). Exploiting emojis for sarcasm detection. In *International conference on social computing, behavioral-cultural modeling and prediction and behavior representation in modeling and simulation* (pp. 70–80).
- Sundararajan, K., & Palanisamy, A. (2020). Multi-rule based ensemble feature selection model for sarcasm type detection in twitter. *Computational intelligence and neuroscience, 2020*.
- Sundararajan, K., Saravana, J. V., & Palanisamy, A. (2021). Textual feature ensemble-based sarcasm detection in twitter data. In *Intelligence in big data technologies—beyond the hype* (pp. 443–450). Springer.
- Tay, Y., Tuan, L. A., Hui, S. C., & Su, J. (2018). Reasoning with sarcasm by reading in-between. *arXiv preprint arXiv:1805.02856*.
- Tepperman, J., Traum, D., & Narayanan, S. (2006). " yeah right ": sarcasm recognition for spoken dialogue systems. In *Ninth international conference on spoken language processing*.
- Thakur, S., Singh, S., & Singh, M. (2018). Detecting sarcasm in text. In *International conference on intelligent systems design and applications* (pp. 996–1005).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems, 30*.
- Veale, T., & Hao, Y. (2010). Detecting ironic intent in creative comparisons. In *Ecai 2010* (pp. 765–770). IOS Press.
- Walker, M., Tree, J. E. F., Anand, P., Abbott, R., & King, J. (2012). A corpus for research on deliberation and debate. In *Proceedings of the eighth international conference on language resources and evaluation (lrec'12)* (pp. 812–817).
- Walker, M. A., Anand, P., Abbott, R., Tree, J. E. F., Martell, C., & King, J. (2012). That is your evidence?: Classifying stance in online political debate. *Decision Support Systems, 53*(4), 719–729.
- Wallace, B. C., Kertz, L., Charniak, E., et al. (2014). Humans require context to infer

- ironic intent (so computers probably do, too). In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers)* (pp. 512–516).
- Wang, Z., Wu, Z., Wang, R., & Ren, Y. (2015). Twitter sarcasm detection exploiting a context-based model. In *international conference on web information systems engineering* (pp. 77–91).
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... Rush, A. M. (2020, October). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations* (pp. 38–45). Online: Association for Computational Linguistics. Retrieved from <https://www.aclweb.org/anthology/2020.emnlp-demos.6>
- Wu, C., Wu, F., Wu, S., Liu, J., Yuan, Z., & Huang, Y. (2018). Thu\_ngn at semeval-2018 task 3: Tweet irony detection with densely connected lstm and multi-task learning. In *Proceedings of the 12th international workshop on semantic evaluation* (pp. 51–56).
- Xiong, T., Zhang, P., Zhu, H., & Yang, Y. (2019). Sarcasm detection with self-matching networks and low-rank bilinear pooling. In *The world wide web conference* (pp. 2115–2124).
- Xu, N., Zeng, Z., & Mao, W. (2020). Reasoning with multimodal sarcastic tweets via modeling cross-modality contrast and semantic association. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 3777–3786).
- Zeng, B., Yang, H., Xu, R., Zhou, W., & Han, X. (2019). Lcf: A local context focus mechanism for aspect-based sentiment classification. *Applied Sciences*, 9(16), 3389.
- Zhang, M., Zhang, Y., & Fu, G. (2016). Tweet sarcasm detection using deep neural network. In *Proceedings of coling 2016, the 26th international conference on computational linguistics: technical papers* (pp. 2449–2460).