Sarcasm detection in online comments using machine learning

Daniel Šandor and Marina Bagić Babac
Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia

Abstract

Purpose – Sarcasm is a linguistic expression that usually carries the opposite meaning of what is being said by words, thus making it difficult for machines to discover the actual meaning. It is mainly distinguished by the inflection with which it is spoken, with an undercurrent of irony, and is largely dependent on context, which makes it a difficult task for computational analysis. Moreover, sarcasm expresses negative sentiments using positive words, allowing it to easily confuse sentiment analysis models. This paper aims to demonstrate the task of sarcasm detection using the approach of machine and deep learning.

Design/methodology/approach – For the purpose of sarcasm detection, machine and deep learning models were used on a data set consisting of 1.3 million social media comments, including both sarcastic and non-sarcastic comments. The data set was pre-processed using natural language processing methods, and additional features were extracted and analysed. Several machine learning models, including logistic regression, ridge regression, linear support vector and support vector machines, along with two deep learning models based on bidirectional long short-term memory and one bidirectional encoder representations from transformers (BERT)-based model, were implemented, evaluated and compared.

Findings – The performance of machine and deep learning models was compared in the task of sarcasm detection, and possible ways of improvement were discussed. Deep learning models showed more promise, performance-wise, for this type of task. Specifically, a state-of-the-art model in natural language processing, namely, BERT-based model, outperformed other machine and deep learning models.

Originality/value – This study compared the performance of the various machine and deep learning models in the task of sarcasm detection using the data set of 1.3 million comments from social media.

Keywords Sarcasm, Natural language processing, Machine learning, Deep learning, BERT, Reddit

Paper type Research paper

1. Introduction

Sarcasm detection is the task of identifying whether a given piece of text is sarcastic or not that has valuable applications in the real world (Davidov *et al.*, 2010). For example, in the world of business and marketing, sarcasm is a type of sentiment analysis task that, among other things, can be used for brand monitoring, customer feedback analysis, opinion mining and market research (Puh and Bagić Babac, 2023). Although sarcasm can often cheat standard sentiment analysis models, sarcasm detection can be used to collect truthful information about the general public's view of a product or brand (Riloff *et al.*, 2013).

The problem of sarcasm detection is challenging because it involves a complex interplay of linguistic, pragmatic and contextual factors (Reyes *et al.*, 2012). Sarcasm can be expressed in many ways, ranging from subtle to overt, and can depend on a range of linguistic and contextual cues (Băroiu and Trăușan-Matu, 2022). For example, sarcasm can be conveyed using exaggeration, understatement, irony or parody and can involve a range of linguistic features such as lexical ambiguity, negation and presupposition (Ashwitha *et al.*, 2021).

A variety of machine learning algorithms have been applied to the task of sarcasm detection, including naive Bayes, support vector machines, random forests, recurrent neural networks and convolutional neural networks (CNNs) (Poria *et al.*, 2016). These algorithms work by learning to recognize patterns in text data that are associated with sarcasm (Zhang and Wallace, 2018). On the other hand, the choice of features is an important factor in sarcasm detection (Arora, 2020). Various features have been used for sarcasm detection, including lexical, syntactic and semantic features. Lexical features involve the frequency of certain words or phrases that are often associated with sarcasm, while syntactic features involve the use of parts of speech and other grammatical structures to detect sarcasm (Ghosh *et al.*, 2018). Semantic features involve the use of word embeddings or other techniques to capture the meaning of the text.

In recent years, *bidirectional encoder representations from transformers* (BERT) has emerged as a state-of-the-art method for various natural language processing tasks (Devlin *et al.*, 2019),

The current issue and full text archive of this journal is available on Emerald Insight at: **https://www.emerald.com/insight/2398-6247.htm**

Information Discovery and Delivery
52/2 (2024) 213–226
Emerald Publishing Limited [ISSN 2398-6247]
[DOI 10.1108/IDD-01-2023-0002]

Received 3 January 2023
Revised 11 March 2023
9 June 2023
Accepted 4 July 2023

including sarcasm detection. However, one major limitation of BERT is its computational cost. BERT is a large-scale deep learning model with millions of parameters, making it computationally expensive to train and use. It is a complex model that can be difficult to interpret. This can make it challenging to understand how the model is making predictions or diagnose errors or biases in the model. Despite these limitations, BERT remains a powerful tool for sarcasm detection. However, it is important to consider these limitations when using BERT and explore simpler machine learning algorithms that may be more appropriate for certain applications.

Each implementation of machine learning algorithms and neural network architectures has its own choice of hyperparameters. Consequently, there exists a research gap regarding the fine-tuning of parameters in existing algorithms, especially considering the inherently complex task of sarcasm detection. Our contribution in addressing this gap lies in our own adjustment of hyperparameters, the selection of neural network layers and the choice of features. This study shows how relatively simple machine learning algorithms can yield comparable results to more complex ones that may be sufficient for sarcasm detection in some cases. We designed, implemented and tested our own simple neural architectures that showed promising results. Although the BERT-based model outperformed other models, this study shows which models can be used in less computationally complex settings. For example, we demonstrated that ridge regression yields promising results, which is less known.

2. Literature overview

Sarcasm detection is a focused research field in natural language processing with the subject approached in different ways. Semi-supervised pattern extraction, hashtag-based supervision and speech-based analysis are some of the methods used in past (Reyes *et al.*, 2012). There are also two opposed approaches concerning the model selection for the task of classification. Standard machine-learning models require manual feature extraction from data (Ghosh and Veale, 2017). This approach leaves more control to the researcher, but the task of choosing the correct, useful features is usually as difficult as choosing the correct machine learning model. In a study by Bouazizi and Otsuki (2016), features were divided into sentiment-related, punctuation-related, syntax-related and pattern-related features. This extensive feature selection proved effective in machine learning classification models. Deep learning models provide automatic feature extraction. They can potentially discover subtle semantic patterns and capture context. They also require large amounts of data and are highly configurable. Finding the correct model proved to be a difficult task on its own.

Sarcasm detection is mostly used as a subcomponent of sentiment analysis. Sentiment analysis refers to the contextual analysis of text which identifies and extracts subjective information in the source material, commonly related to a business problem (Bagić Babac and Podobnik, 2016). Sarcasm detection is a task that requires large data sets. In literature, data is mostly collected from social media websites, like Twitter. In the context of sarcasm detection, the annotation of data is a difficult and time-consuming process. Joshi *et al.* (2016) used two data sets manually labelled by American annotators to analyse how cultural differences impact the quality of annotation. Indian annotators faced difficulties mostly because of a lack of context for the provided data. Either concerning the text itself or the abstraction (e.g. the socio-economic situation at the time) the text refers to. Data bias, unbalance between the amount of sarcastic and non-sarcastic data, also influenced their annotations. The language barrier proved troublesome as well. The alternative to manual annotation is the automatic labelling of data (Musso and Bagić Babac, 2022). Automatic labelling of data can be used, for example, by searching for "#sarcasm" or similar hashtags. This method can provide data sets with more false positives and false negatives. This study used an automatically labelled data set.

Data collected from social media websites contain text that is written in an informal, colloquial language. Some text pre-processing is required. In a study by Kumar and Anand (2020), data is taken from labelled Twitter and Reddit data sets. To remove the noise from the data, they removed unwanted punctuation, multiple spaces, URL tags, changed different abbreviations to their contracted format, etc. Saha *et al.* (2017) proposed a model for sentiment analysis of sarcastic tweets that requires data that has been pre-processed in a certain manner, e.g. stop-words were removed, and text was tokenized before each word was assigned its category, i.e. parts of speech tagging.

When using standard machine learning models, feature extraction plays an important role. To extract more context from the data, a set of features should be defined. In a study by Buschmeier *et al.* (2014), a set of features to determine if a review is ironic or not was created. Some of these features are imbalance, hyperbole and punctuation. They marked a review imbalanced if the given star rating was high, but most words in the text had negative sentiment, and vice versa. The review was marked hyperbolic if there were three or more positive or negative words in a row. The punctuation feature marks the presence of multiple questions or exclamation marks (Nayel *et al.*, 2021).

Deep learning models have also been used for sarcasm detection due to their ability to automatically learn complex patterns and representations from raw text data. In the case of deep learning models, model selection is the difficult part (Gupta *et al.*, 2020). Transformer models are a recent development in deep learning and have been shown to be highly effective for a wide range of NLP tasks.

Hazarika *et al.* (2018) introduced CASCADE, a contextual sarcasm detector, that effectively leveraged content and contextual information. By integrating user profiling and discourse modelling with a CNN-based textual model, CASCADE achieved state-of-the-art results on a large-scale Reddit corpus. This research underscored the potential of deep learning in capturing intricate contextual cues for sarcasm detection. Likewise, Pant and Dadu (2020) asserted the significance of context in refining the performance of modals based on contextual word embedding.

Scola and Segura-Bedmar (2021) extended the application of these models beyond social media data. They investigated the use of bidirectional long short-term memory (BiLSTM) models for sarcasm detection in news headlines. This study bridged the gap between social media and traditional news

texts, demonstrating the capability of deep learning models to detect sarcasm across diverse domains.

As deep learning models gained prominence, the introduction of BERT marked a significant breakthrough. Khatri and Pranav (2020) incorporated BERT and GloVe embeddings to detect sarcasm in tweets. By considering both the tweet content and the contextual information, their approach showcased the effectiveness of BERT in capturing the nuanced contextual cues necessary for accurate sarcasm detection. BERT's ability to capture bidirectional dependencies in text revolutionized the field, enabling a more sophisticated understanding of sarcastic language.

Capitalizing on BERT's success, Parameswaran *et al.* (2021) further fine-tuned BERT models, namely, using additional domain-specific data. This approach demonstrated the transferability of BERT's contextual representations across domains, resulting in enhanced performance in sarcasm detection. In addition, Savini and Caragea (2022) investigated the performance improvement brought by BERT-based models, both with and without intermediate task transfer learning, compared to previous works. The study specifically focused on the significance of message content in sarcasm detection, showing that BERT models using only the message content outperformed models leveraging additional information from a writer's history encoded as personality features. This research shed light on the effectiveness of BERT-based models and emphasized the importance of considering message content for accurate sarcasm detection.

Recently, Sharma *et al.* (2023) incorporated word and phrase embeddings, including BERT, and used fuzzy logic evolutionary techniques to refine classification accuracy. This novel approach aimed to overcome the limitations of traditional deterministic models by introducing fuzzy reasoning, enabling improved handling of uncertainty and ambiguity in sarcasm detection.

These advancements underscore the importance of leveraging contextual information, incorporating domain-specific data (Misra and Arora, 2023) and exploring innovative techniques to better understand and interpret the intricate nature of sarcastic language.

Sarcasm detection dependents on data set quality, pre-processing methods and feature engineering, so choosing an appropriate model is not a straightforward process, as it requires careful consideration and evaluation of various factors such as performance metrics, interpretability and computational resources. In addition, the subjective nature of sarcasm makes it challenging to accurately capture and classify in text, further emphasizing the importance of thorough analysis and experimentation in developing effective models for sarcasm detection. This study presents an approach to detect sarcasm in natural language using machine and deep learning models using a data set of 1.3 million Reddit comments by extracting additional features for analysis. The study confirms the potential of using machine and deep learning techniques for sarcasm detection and provides insights for further improvement.

3. Research methodology

3.1 Exploratory data analysis

The data set [1] used in this study was downloaded from *Kaggle*, an online community of data scientists and machine learning practitioners. It contains 1.3 million sarcastic statements from the internet commentary forum-like website Reddit (Khodak *et al.*, 2018). The data was collected by extracting comments from Reddit forums that included a tag denoting sarcasm ("\s"). This tagging convention is commonly used by Reddit users to identify sarcastic remarks. Non-sarcastic comments were also scraped to balance the data set. The data set is divided into the train and test sets. The training set contains 1,010,826 items of data. Additional features were extracted while scraping the data. Each item contains the following information: *comment*, *the parent comment* to which the comment is related, *author* of the comment, *subreddit* (a specific forum), *number of upvotes* (likes) for the comment, *number of downvotes* (dislikes) for the comment, the *score* (calculated as the absolute number of upvotes subtracted from the number of downvotes), the *date* the comment was posted in *YYYY-MM* format and in *UTC* format. Finally, *the label* determines whether the comment is sarcastic or not.

Table 1 shows the last five items of data in *pandas DataFrame* format. The *pandas* is a fast, powerful, flexible open-source data analysis tool built on top of the Python programming language. It is used here for data manipulation and processing.

3.2 Data pre-processing

Text as a representation of language is a formal system that follows certain syntactic and semantic rules. It is complex and difficult to interpret for computational systems. Text pre-processing is an integral part of any natural language processing task (Kampić and Bagić Babac, 2021). It is done to simplify the complex forms of natural text for easier processing by the machine learning model which uses it. The text is cleaned from noise in form of emoticons, punctuation, letters in a different case, stop words and so on.

In this data set, some items with blank comments were observed. Because the comment text is the primary focus of this paper, 53 items with blank comments were removed. Also, some duplicate items were observed. After the removal of these duplicates, 1,010,745 rows of data remained. In addition, certain faulty score calculations were observed. To ensure the correctness of the data, the score was recalculated as the number of downvotes (downs column) subtracted from the number of upvotes (ups column).

The primary point of analysis in this study was the comments and the parent comments they responded to. This text needed to be further processed before it could be supplied to the machine learning models. Firstly, the short forms of words were decontracted (e.g. "won't" was transformed to "will not", "'m" was transformed to "am", "'ve" was transformed to "have", etc.). This was done by applying several regular expressions using the sub-function from the *re* library available in Python. Next, the return symbol ("\r"), the newline symbol ("\n") and the quote symbol (" \" ") were replaced by a single whitespace. Then, the punctuation was removed from the sentences. For this, a constant in the Python string library containing all punctuation signs ("!#$%&'()*+, -./:;<=>?@[\]^_`{|}~") was used. Then, again with the use of the sub-function from Python's *re* library, all non-alphanumerical values were removed.

The sentences were tokenized using the *Natural Language Toolkit* (NLTK) function *word_tokenize*. The NLTK is a

Table 1 Sample data from the used data set

Id.	Label	Comment	Author	Subreddit	Score	Ups	Downs	Date	Created UTC	parent_comment
1010821	1	I'm sure that Iran and N. Korea have the techn. . .	TwarkMain	reddit.com	2	2	0	2009-04	2009-04-25 00:47:52	No one is calling this an engineered pathogen,. . .
1010822	1	Whatever you do, don't vote green	BCHarvey	Climate	1	1	0	2009-05	2009-05-14 22:27:40	In a move typical of their recent do-nothing a. . .
1010823	1	Perhaps this is an atheist conspiracy to make . . .	rebelcommander	Atheism	1	1	0	2009-01	2009-01-11 00:22:57	Screw the Disabled– I've got to get to Church. . .
1010824	1	The Slavs got their own country – it is called. . .	catsi	Worldnews	1	1	0	2009-01	2009-01-23 21:12:49	I've always been unsettled by that. I hear a I. . .
1010825	1	Values, as in capitalism there is good mone. . .	frogking	Politics	2	2	0	2009-01	2009-01-24 06:20:14	Why do the people who make our laws seem unabl. . .

Note: A sample from the public data set on Kaggle.com
Source: The data set is made by Khodak *et al.* (2018); www.kaggle.com/datasets/danofer/sarcasm

platform used for building Python programs that work with human language data for application in statistical natural language processing. It contains text-processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. The function *word_tokenize* transforms text into a list of words. These words were transformed into lowercase. Stop words were removed from the list of words. Stop words are the most common words in any language (like articles, prepositions, pronouns, etc.). They do not add much information to the text and should be removed so the model has less noise in the data to deal with (Kostelej and Bagić Babac, 2022). Examples of a few stop words in English are "the", "a", "an", "so", "what". NLTK library offers English stop words in form of a list. Then, the words were tagged with their language form, i.e. parts of speech tagging.

The process of classifying words into their parts of speech and labelling them accordingly is known as part-of-speech (POS) tagging, grammatical tagging or simply tagging (Bandhakavi *et al.*, 2017). Parts of speech are also known as word classes or lexical categories (Kumawat and Jain, 2015). The collection of tags used for a particular task is known as a tag set (Bird *et al.*, 2009). A part of speech is a grammatical category, commonly including verbs, nouns, adjectives, adverbs, determiners and so on. In this article, a smaller subset of available tags was chosen. Words were tagged with one of the following tags: noun, verb, adjective or adverb. This was done to increase the performance of word lemmatization. Tokenized words were lemmatized using the NLTK class WordNetLemmatizer. The POS tag was supplied to the lemmatize function. Lemmatization refers to the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word (e.g. "am", "are", "is" is transformed into "be"), which is known as the lemma (Manning *et al.*, 2008). Finally, lemmatized words were combined into text. For example, the comment: "Trick or treating, in general, is just weird [. . .]", was transformed into "trick treat general weird".

The author and subreddit columns are also textual values. They are mostly one-word titles that required simpler processing. The values were transformed into lowercase. Whitespaces were removed. Special signs (dash and underscore) were removed. Finally, dots were replaced with the word "dot". For example, the author "Kvetch_22.", was transformed into "kvetch22dot".

3.3 Data processing

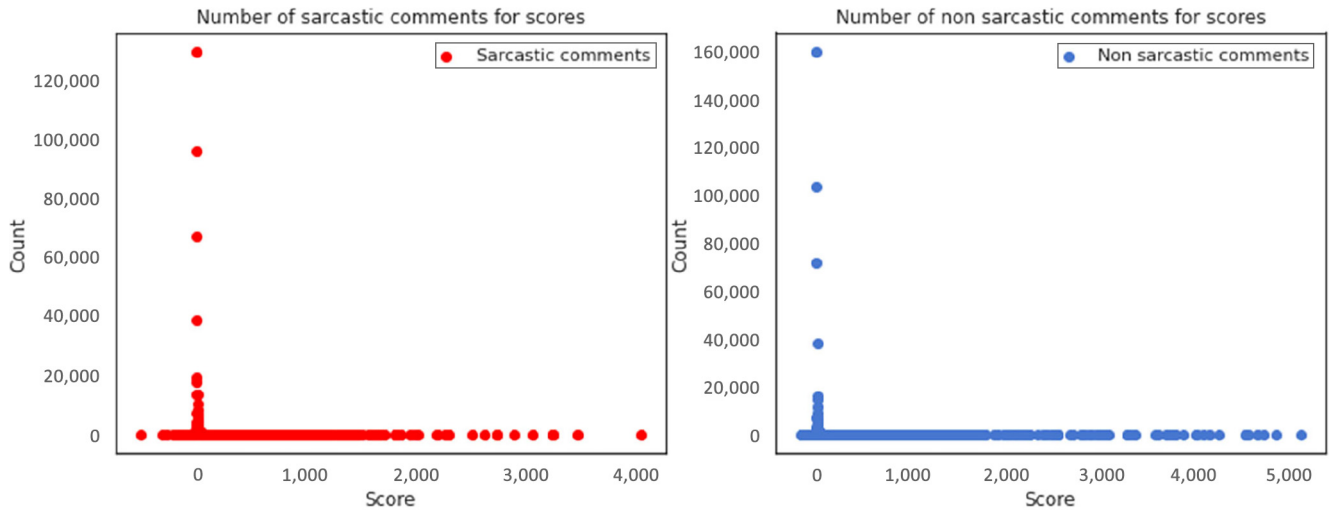
The provided data set is balanced, with 505,340 sarcastic comments and 505,450 non-sarcastic comments. The word "I" is by far the most common word in these comments. Sarcasm is often expressed by contradicting emotions in one sentence, and the word "but" is a conjunction used to connect ideas that contrast. The word "but" does not appear among frequent words in non-sarcastic comments, which could mean the word "but" is a good indicator of sarcasm.

Figure 1 shows the number of comments per score. Both sarcastic and non-sarcastic comments score mostly around 0. Non-sarcastic comments generally have more positive scores. Summed up scores for the sarcastic comments would give a sum of 2,702,923, and the sum of scores for the non-sarcastic rows is 3,002,887.

The top three rated sarcastic comments were "*I think he was really referring to the vuvuzela*", with a score of 4,010; "*Jesus, you wonder why you're still single*!", with a score of 3,444; and "*Yet another thing that men are better at doing*", with a score of 3,220. The top three ranked non-sarcastic comments were "*Getting pushed back by the impact of a bullet.*", with a score of 5,163; "*Ah, a nice guy.*", with a score of 4,909; and "*Does anyone know if Flint has clean water yet?*", with a score of 4,609.

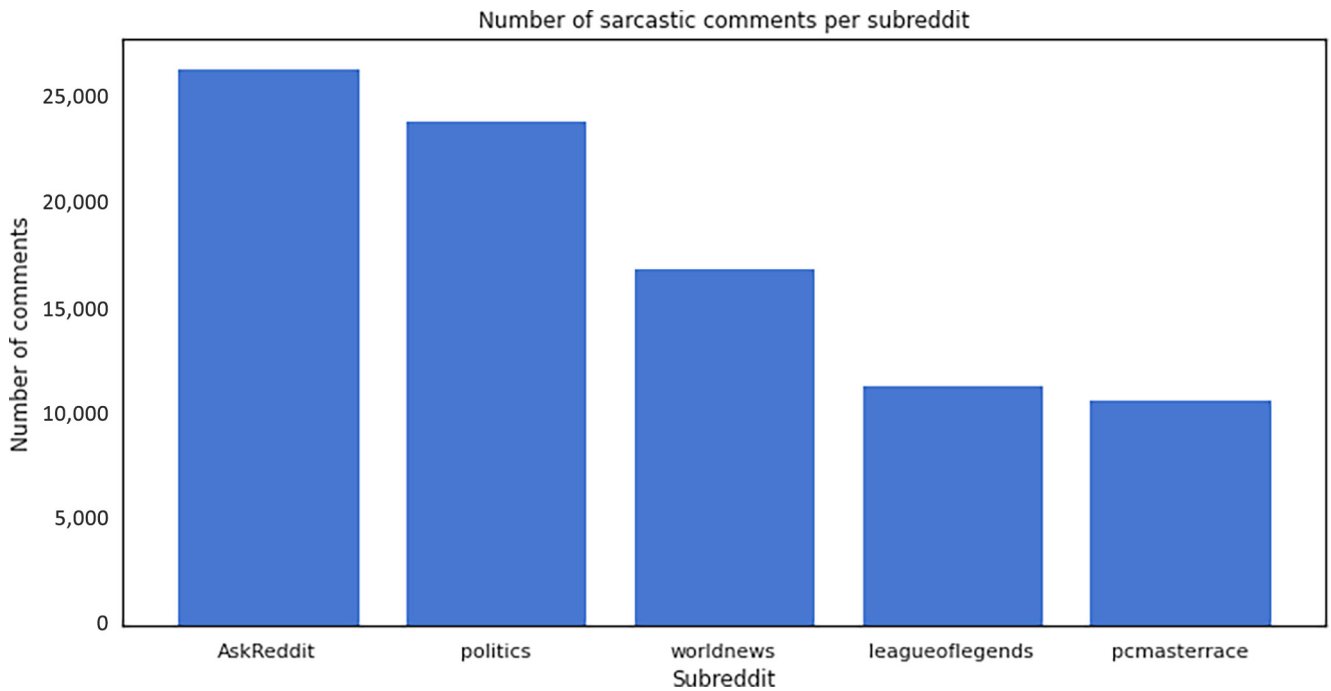
Subreddits, in the context of Reddit, are forums on a specific subject (e.g. data science subreddit). Figure 2 shows the top five subreddits with the most sarcastic comments. Figure 3 shows the top five subreddits with the most non-sarcastic comments. Both sarcastic and non-sarcastic comments are most numerous in the "*AskReddit*" and "*politics*" subreddits. None of the sarcastic

Figure 1 Number of comments per score



Source: Made by the authors

Figure 2 Subreddits with the most sarcastic comments



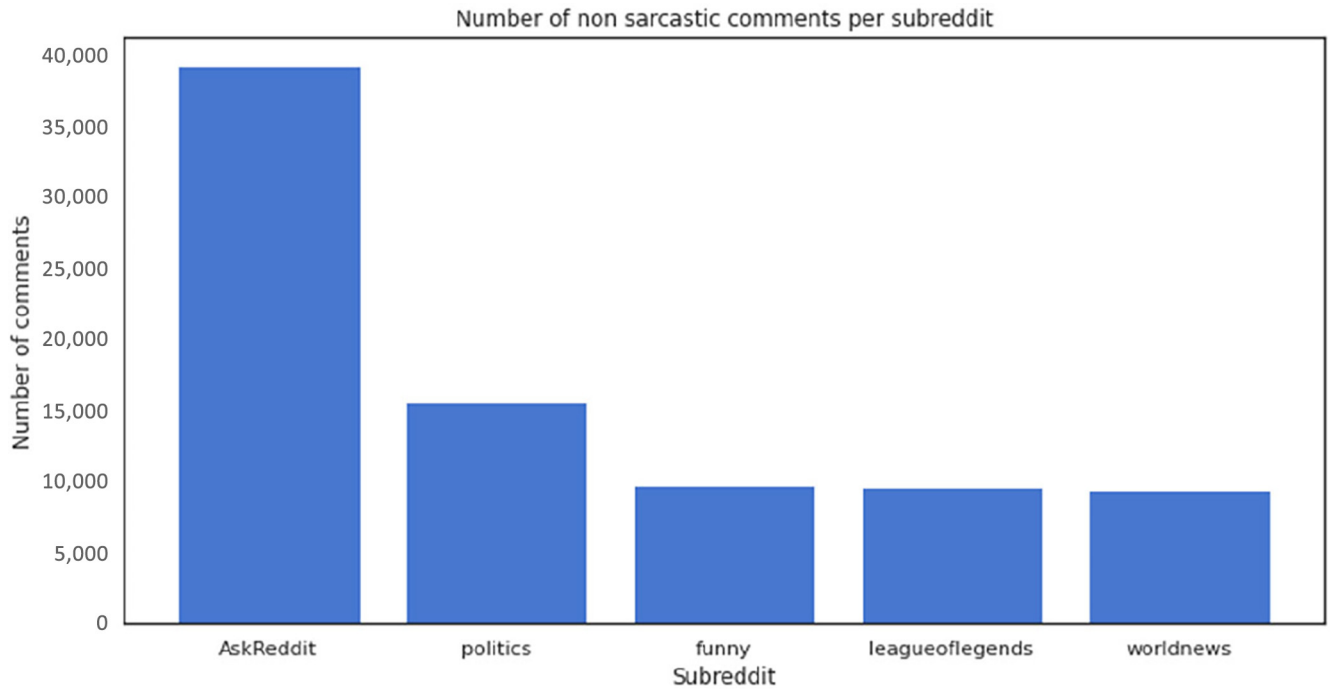
Source: Made by the authors

comments are from the "*funny*" subreddit, which is surprising because sarcasm is often related to humour.

3.4 Feature extraction and analysis

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set (Wang *et al.*, 2018). It yields better results than applying machine learning directly to the raw data. Standard machine learning methods require manual feature extraction. Manual feature extraction requires a good understanding and insight into the given data. Features should be specific to the studied problem (Brzić *et al.*, 2023). Because standard machine learning models are used in this paper, manual feature extraction had to be done. Feature extraction was done on the unprocessed comments.

Figure 3 Subreddits with the most non-sarcastic comments



Source: Made by the authors

The first feature is the uppercase word count (Jurafsky and Martin, 2000). This feature was chosen because capital letters indicate strong arguments and opinions. An assumption was that uppercase letters would be used to indicate strong, negative, sarcastic expressions (e.g. Yeah, I LOVE to WORK and not have FUNN!). That is, the assumption was that sarcastic texts would have more uppercase letters (Ren *et al.*, 2020).

Sentiment analysis is a tightly related problem to sarcasm detection (Nayel *et al.*, 2021). All words in the comment text were analysed for sentiment using the VADER (*Valance Aware Dictionary for sEntiment Reasoner*) sentiment analysis tool. Elbagir and Yang (2019) used VADER, a sentiment analysis tool, for the task of multi-classification of tweets related to the 2016 US election. It showed good accuracy in this data. VADER is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. It was used in this article because the data used is scraped from a social media website.

A compound sentiment score was calculated for each word in a comment, except stop words. This is the most useful metric if a single unidimensional measure of sentiment for a given sentence is required. It is a normalized, weighted composite score. Values span from −1.0 to 1.0, −1.0 indicating a strong negative sentiment, and 1.0 indicating a strong positive sentiment. Words that scored higher than 0.4 were classified as positive. Words that scored lower than −0.4 were classified as negative. Words that scored higher than −0.4 and lower than 0.4 were classified as neutral. Positive, neutral and negative words were counted for each comment. The assumption was that sarcastic comments would contain more positive and negative words and less neutral words than non-sarcastic comments. The reason for that is that sarcasm is often expressed with strong positive or negative statements (Avdić and Bagić Babac, 2021).

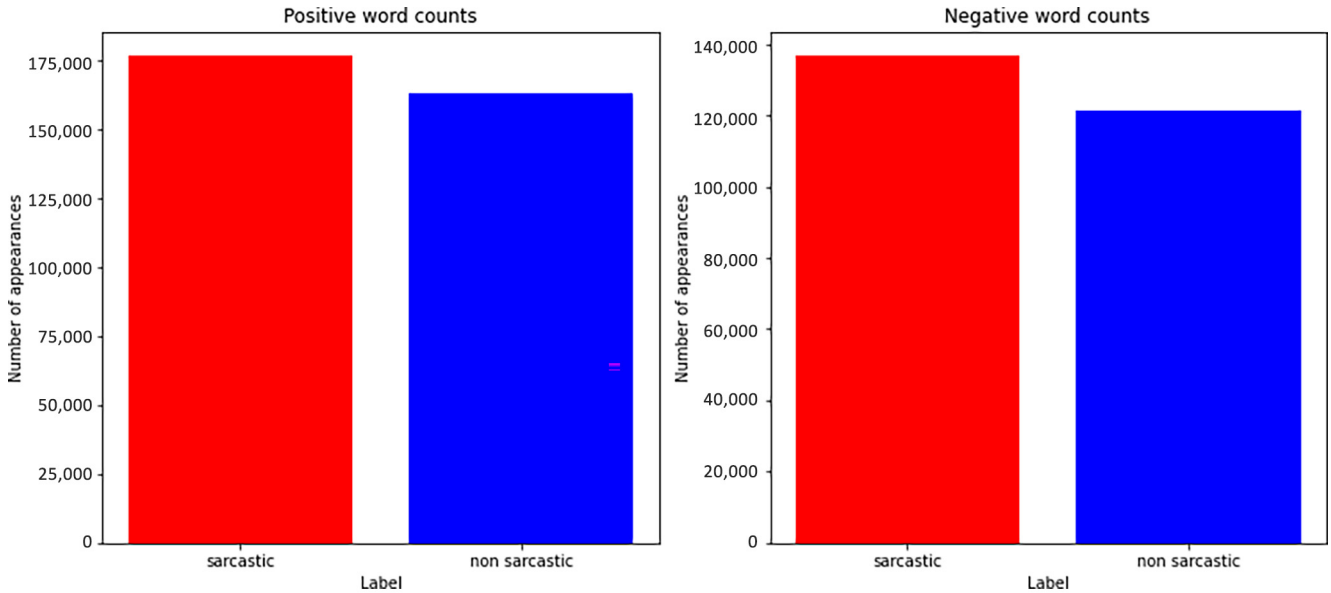
Sarcasm is often portrayed by using contrasting statements. Because of this, the absolute difference in sentiment between the comment and its parent comment was calculated. Compound scores for the whole texts were subtracted. The assumption was that sarcastic comments would have a larger difference in polarity with their parent comment than non-sarcastic comments. The compound score for the whole comment text was also recorded.

Sarcasm can also be expressed using certain syntactic features (Ashwitha *et al.*, 2021). The first syntactic feature extracted was the repeated letter count. Appearances of three or more letters are counted. The assumption is that sarcastic words will have more occurrences of repeated letters (e.g. "Yeaaaah, riiiight"). Again, by using the constant containing punctuation from the string library in Python, punctuations are counted for each comment. The assumption is, sarcastic comments would have more punctuation (e.g. "I adore being bored!!!!"). Dots (e.g. "*I love hard work* [. . .]") and quotation marks (e.g. "I "love" hard work") were the best indicators of sarcasm, so their counts are calculated separately.

By incorporating these feature extraction methods, we aimed to capture distinct linguistic and syntactic patterns associated with sarcasm. These features provide valuable insights into the characteristics of sarcastic comments, enabling effective detection and analysis.

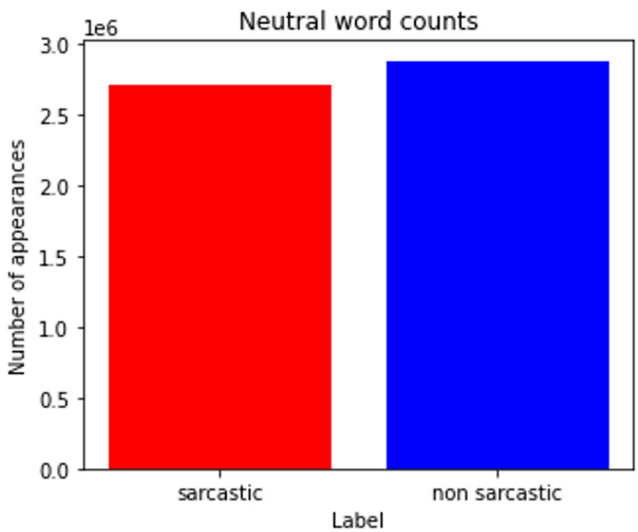
After feature extraction, further data analysis is possible (Cvitanović and Bagić Babac, 2022). Figure 4 shows that sarcastic comments had more of both positive and negative words. Figure 5 shows that there were slightly more neutral

Figure 4 Positive and negative word counts per label



Source: Made by the authors

Figure 5 Neutral word counts per label



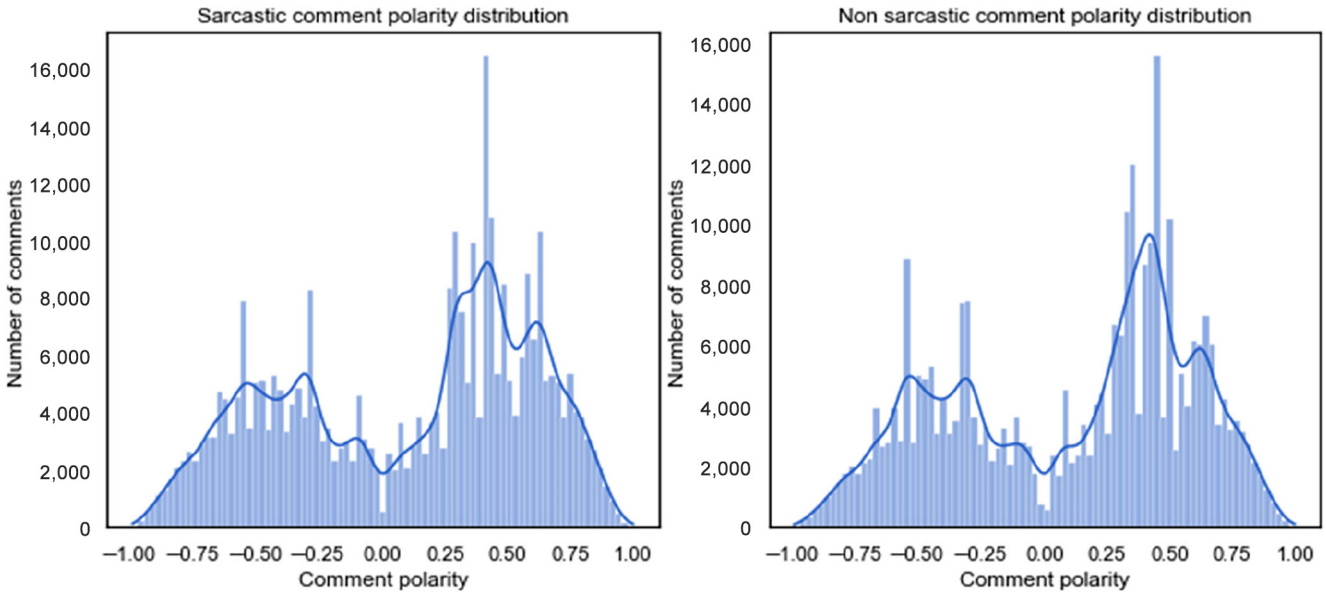
Source: Made by the authors

words in the non-sarcastic comments. Neutral words were more numerous for both comment types. This fits with the assumptions made in the previous paragraph. Sarcastic comments carry slightly more strong opinions.

Figure 6 shows the distribution of sarcastic (left) and non-sarcastic (right) comment polarity scores. There is a much higher number of neutral comments, with a compound sentiment score of 0.0, than negative or positive comments. This is to be expected because neutral words are most common in spoken language. Comments with a score of 0.0 were excluded for analysis' sake. It can be observed that both sarcastic and non-sarcastic comment polarity is concentrated around the score of 0.5. The second-highest concentration of polarity is around −0.5.

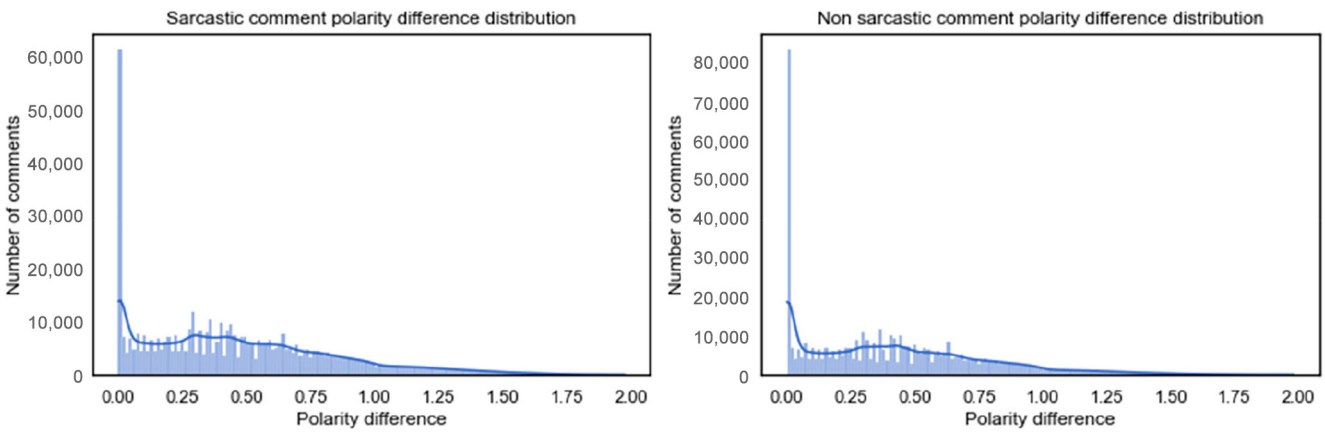
Figure 7 shows the differences between the difference between compound sentiment scores of comments and their respective parent comments. It can be observed that the polarity difference for both types of comments is concentrated at the value of 0.0. That is because most comments and parent comments have a compound sentiment value of 0.0. It can also be observed that sarcastic comments have an overall slightly larger polarity difference from non-sarcastic comments.

Figure 6 Comment polarity distribution



Source: Made by the authors

Figure 7 Polarity difference between comments are parent comments



Source: made by the authors

4. Results from machine learning models

The process of sarcasm detection is essentially a binary classification problem, with the two labels being "sarcastic" and "non-sarcastic". The pre-processed comments at this point were not ready to be used as input in standard machine learning models. Additional data transformation was required.

Comment, parent comments, authors and subreddits are all textual fields. Their pre-processed values were vectorized using the TF-IDF (*Term Frequency Inverse Document Frequency*) Vectorizer available in Python. For the comments and parent comments, features were made of words 1 gram, 2 grams and 3 grams, in hope of preserving context. For the authors and subreddits, features were made of word 1 grams, as they are mostly one-word titles. The vocabulary for comments and parent comments was built considering only the top 5,000 features ordered by term frequency across the corpus. The vocabulary for authors and subreddits is considered the top 1,000 features. When building the vocabulary, terms that have a document frequency lower than 10 were ignored. During the pre-processing step, accents were removed, and another character normalization was done. All numerical features (score, ups, downs, uppercase count, polarity difference, positive word count, negative word count, neutral word count, repeated letters count, punctuation count, dot count and quote count) were added to the corresponding rows.

The data was split into train and test sets (Poch Alonso and Bagić Babac, 2022); 20% of the data went to the testing data set

which made a training set of 808,596 items and a testing set of 202,149 items, and a total of 12,013 features for each data set, that is 10,000 for comments and parent comments, 2,000 for authors and subreddits and 13 for numerical features.

The performance of all models was measured by the average accuracy, F1 score, precision and recall. Cross-validation with 10 stratified folds was used on the training data. Final performance metrics were calculated as the mean of metrics in all folds. The best-performing model, according to accuracy, was used to get predictions on the test set which was not used in cross-validation. The confusion matrix containing the number of true positives, false positives, true negatives and false negatives was plotted for this prediction. For all models, none or simple changes, like regularization strength, were made. The performance of these models is compared to neural network models.

The first classification model used was a logistic regression with the limited-memory Broyden–Fletcher–Goldfarb–Shanno (algorithm) solver and L2 regularization (the inverse of regularization strength C was set to 0.8). Mean accuracy was 63.2%, mean F1 score was 60.4%, mean recall was 56.3% and mean precision was 65.4%. Figure 8(a) shows the confusion matrix. There were 74,465 true negatives, 48,799 false negatives, 26,806 false positives and 52,079 true positives.

The second classification model is the classifier using ridge regression with the solver chosen automatically by the data type. The mean accuracy was 70%, the mean F1 score was 68.4%, the mean recall was 65.6% and the mean precision was 71.4%. Figure 8(b) shows the confusion matrix. There are 74,818 true negatives, 34,760 false negatives, 26,453 false positives and 66,118 true positives. In comparison to logistic regression, the ridge regression classifier outperforms logistic regression by all used performance metrics.

The third classification model is the linear support vector machine (SVM) with Stochastic Gradient Descent (SGD) learning and L2 regularization. The gradient of the loss is estimated for each sample at a time, and the model is updated along the way with a decreasing strength schedule (aka learning rate). Alpha, the factor which multiplies the regularization term and the learning rate, is set to 0.0001. The training was limited to 50,000 iterations. The mean accuracy was 67.6%, the mean F1 score was 63.3%, the mean recall was 56.3% and the mean precision was 73.1%. Figure 8(c) shows the confusion matrix. There are 76,180 true negatives, 38,469 false negatives, 25,091 false positives, and 62,409 true positives. In comparison to the logistic regression classifier, this linear SVM classifier outperforms logistic regression by all used performance metrics. In comparison to the ridge regression classifier, this model underperforms in all performance metrics except precision.

The fourth classification model is the linear support vector classifier with L2 regularization. It is like the regular support vector classifier with the linear kernel, but it should work better on many samples according to the model documentation. The inverse regularization parameter, C, was set to 0.9. The mean accuracy was 70%, the mean F1 score was 68.3%, the mean recall was 65.8% and the mean precision was 71.3%. Figure 8(d) shows the confusion matrix. There are 73,871 true negatives, 33,694 false negatives, 27,400 false positives and 67,184 true positives. In comparison to the ridge classifier, this linear SVM classifier performs equally as well. The ridge regression classifier has slightly better precision.

According to a study done by Li *et al.* (2022), in which an overview of standard machine learning models and deep learning models used for text classification tasks was given, SVMs, usually, perform well on text classification tasks. This trend was noticed here as well. Specifically, the SVM implementation in Python proved better performing than most other models. Ridge regression classification gave competitive results in comparison with the SVM classifier. Our results are summarized in Table 2.

5. Results from deep learning models

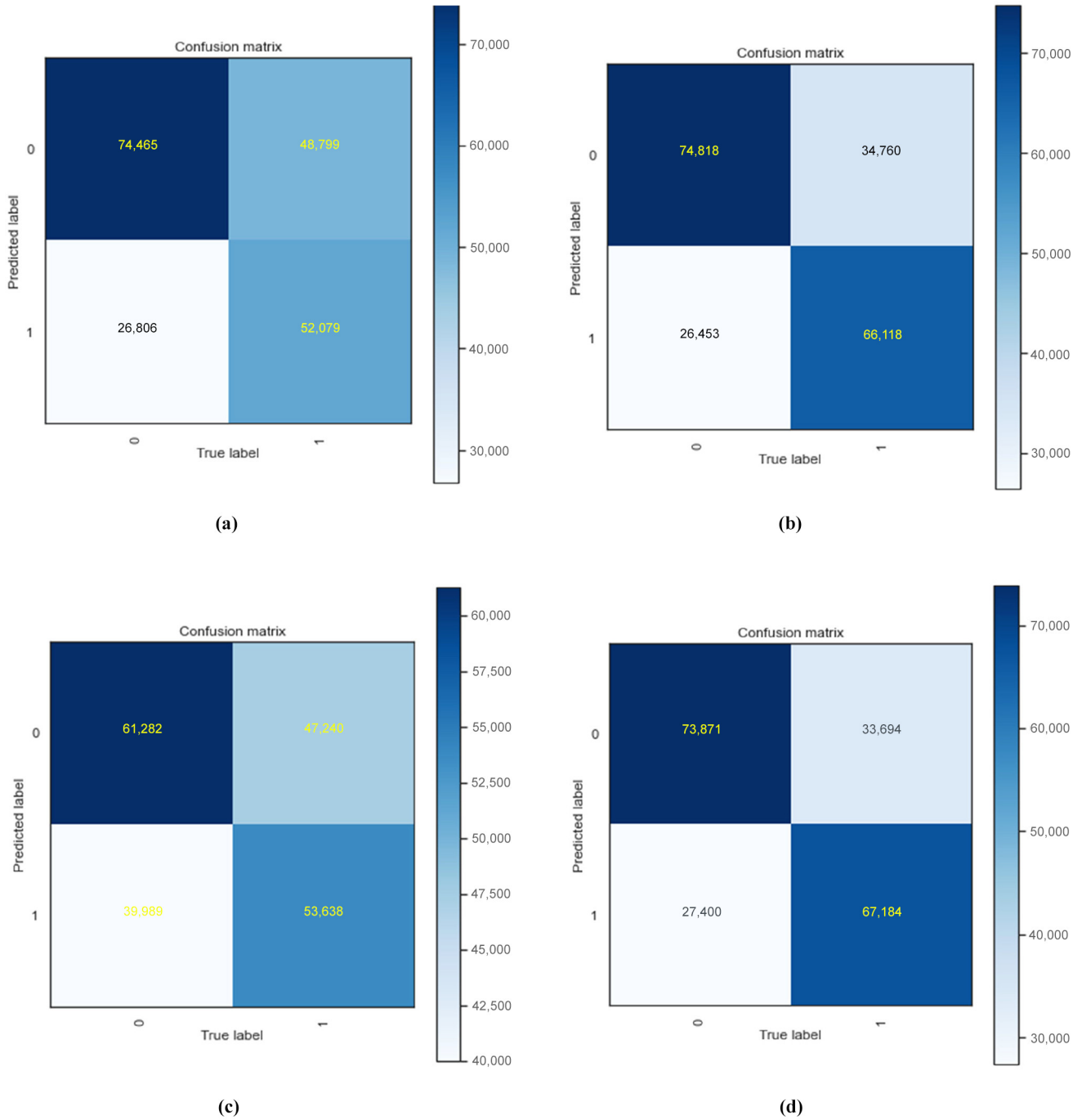
Deep learning models have been showing good results in text classification problems since the early 2010s. They usually do not require feature extraction because they integrate feature extraction and creation into the model fitting process, and they map features directly to outputs using non-linear transformations (Goodfellow *et al.*, 2016). Deep learning models can be a good fit for the task of sarcasm detection because sarcasm is deeply dependent on context. Deep learning models can account for the serial structure and contextual information found in textual data.

Deep learning models required some additional text pre-processing before the data could be used as input. The only textual data used for these models were the comments. The already pre-processed comments were further tokenized using the Keras library *Tokenizer* class in Python. This class enables the vectorization of a whole text corpus. The tokenizer is trained on the comments from the train set. This creates a vocabulary of words with their indices. Each text in the train and test set is then turned into a sequence of integers of the same size. The integers represent the index of the word in the vocabulary. The sequences of integers (tokens) are set to a fixed size of 250. If the sequence is shorter, it is padded with zeros to 250 tokens. If the sequence is longer, it is truncated to 250 tokens.

The first model was created using the Keras Sequential API. The second model was created using the Keras Functional API because it allows multiple inputs. The models were trained for 10 epochs. The performance of both models was measured by the accuracy, F1 score, precision and recall performance metrics on the test set. The neural networks for both models were simple. The assumption was that even without the added complexity, the automatic feature detection ability of neural networks would enable these models to surpass the standard machine learning models.

The first deep learning model used was a five-layer neural network. Firstly, there is the 250-dimensional input layer for the tokenized comments. The second layer is the embedding layer. The vocabulary generated by the process of tokenization is used here to generate the index which will be used to calculate the 16-dimensional embedding of each comment. The following layer is a layer of 16 bidirectional LSTM (*long short-term memory*) cells. It is used here because of the bidirectional LSTM's ability to remember context from the "past" and the "future". The assumption was that the contextual information would capture the nuances of sarcasm in the text. The following layer is the dense layer. This layer contains 24 neurons using the rectified linear unit (*ReLU)* activation function. The final dense layer consists of one

Figure 8 Confusion matrix



Notes: (a) logistic regression; (b) ridge regression; (c) SGD support vector machine; (d) L2 linear support vector
Source: Made by the authors

neuron using the sigmoid activation function. The output is the probability of each label for the input. The loss function used while training the model was binary cross-entropy. The optimization algorithm was Adam. According to the *Keras* documentation, Adam optimization is an SGD method that is based on adaptive estimation of first-order and second-order moments. Figure 9 shows the architecture of the neural network with dimensions of inputs and outputs for each layer.

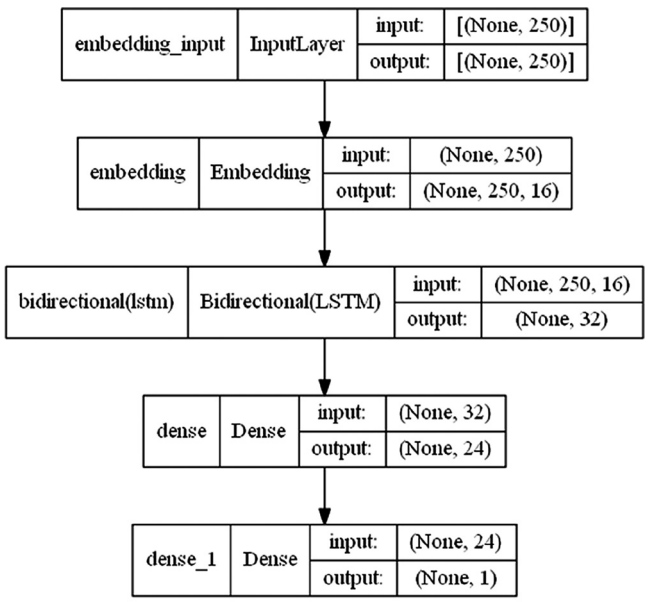
This model had an accuracy score of 68%, an F1 score of 67.7%, a recall score of 68.3% and a precision score of 67.1%. Meaning, it performed better than logistic regression according to all performance metrics, better than the SGD SVM according to accuracy and F1 score and outperformed all models according to recall. The model performed well, even without the manual feature extraction, but it could not be said it is the best model.

Table 2 Summary of results for machine learning models

Model	Accuracy (%)	F1 (%)	Recall (%)	Precision (%)
Logistic regression	63.2	60.4	56.3	65.4
Ridge regression	*70.0*	*68.4*	*65.6*	*71.4*
Support vector machine	67.6	63.3	56.3	73.1
Linear support vector	70.0	68.3	65.8	71.3

Note: Italic are highest values per column
Source: Made by the authors

Figure 9 First deep learning model architecture



Source: Made by the authors

The second deep learning model used was a six-layer neural network. The first layer is the 250-dimensional input layer. The input data is the same as the previous model. Following the input layer is the embedding layer. Words were again embedded in a 16-dimensional feature space. A bidirectional LSTM layer of the same size was used for the comment data. The purpose of this layer is again to capture the contextual nature of the textual data. The following layer differs from the previous model. Here, the manually extracted numerical features are combined with the output of the bidirectional LSTM layer. The next difference is that in this model an extra layer of 12 neurons using the *ReLU* activation function was added. The assumption was that an extra dense layer would better map the added numerical features. The output layer is again a sigmoid function, which gives the probability of both labels. Figure 10 shows the architecture of the neural network with dimensions of inputs and outputs for each layer.

This model had an accuracy score of 67%, an F1 score of 68.1%, a recall score of 67.2% and a precision score of 65.7%. Meaning, adding the manually extracted features and the extra dense layer reduced the performance of the model.

The third deep learning model to test on a classification task of sarcasm detection is BERT, a pre-trained deep learning model that has shown state-of-the-art performance in various natural language processing tasks (Vaswani *et al.*, 2017). In this study, we used the BERT-based neural network model, which consists of 12 layers and 768 hidden units. The model was trained on a large corpus of English texts and was fine-tuned on the specific text classification task at hand (Devlin *et al.*, 2019).

The methodology of implementing BERT for sarcasm detection involves several technical steps (Parameswaran *et al.*, 2021). After the pre-processing of text cleaning and normalization, the BERT tokenizer is used to tokenize the text data into sub-word tokens, which are then converted into numerical representations called input encodings. These input encodings include token IDs, attention masks and segment IDs, which are used to train the BERT-based classification model. The token IDs represent the numerical values of the sub-word tokens, the attention masks indicate which tokens are part of the input sequence and the segment IDs differentiate between two different input sequences. Then, the BERT-based classification model is fine-tuned on the labelled data set. The fine-tuning process involves training the model on the input encodings with a specific loss function and optimizer.

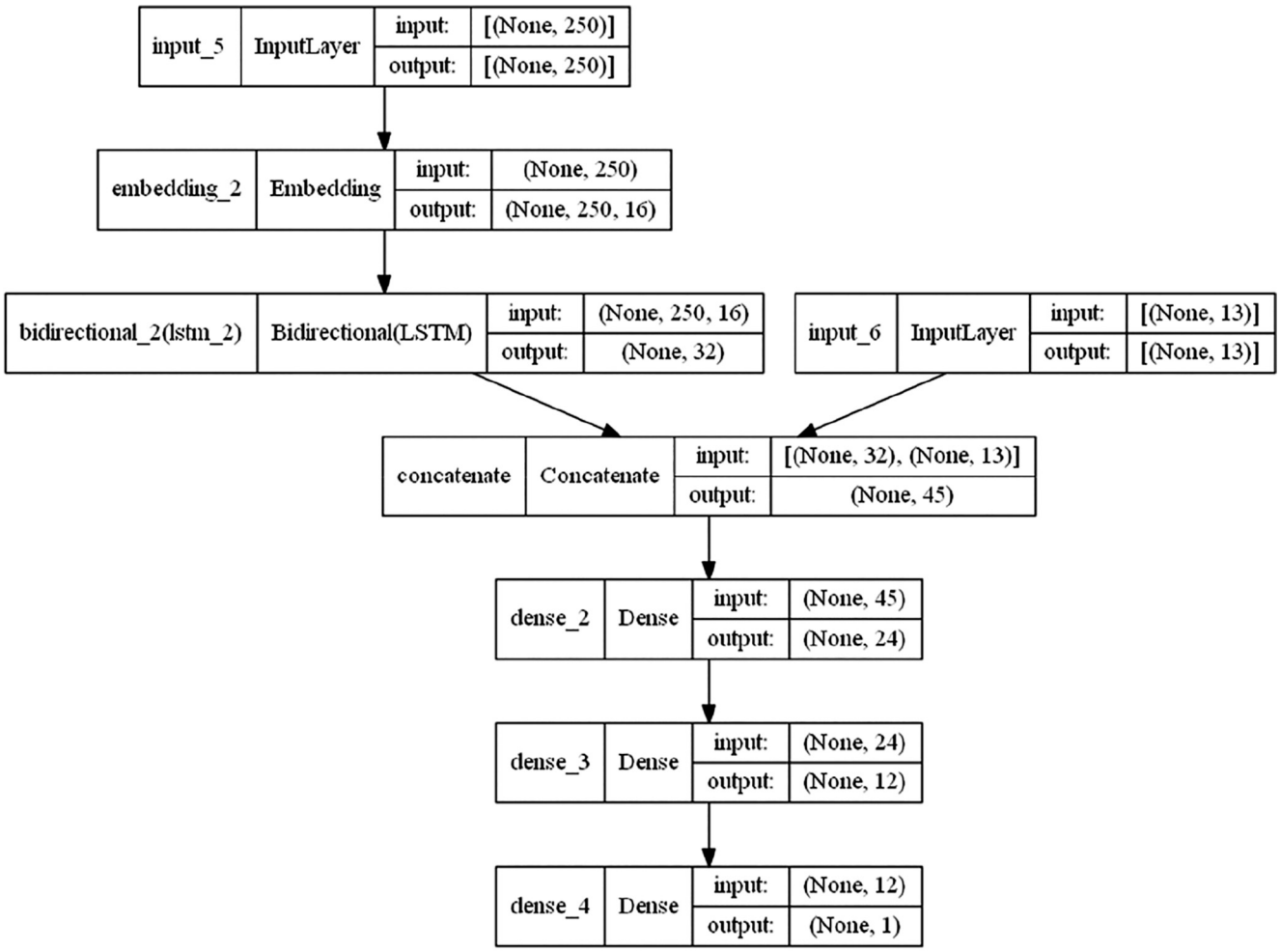
In this study, the loss function used for sarcasm detection here is the sparse categorical cross-entropy loss function, and the optimizer used is the Adam optimizer with a learning rate of 2×10^{-5}. Finally, the fine-tuned BERT-based classification model is evaluated on a separate test data set to assess its performance. The evaluation metrics used to measure the model's performance are shown in Table 3 along with the results of the first and second deep learning models.

The results of our experiments show that the BERT-based model is the most successful in detecting sarcasm, with an accuracy of 73.1%. This model also achieves the highest F1 measure, precision and responsiveness (72.4%, 71.3% and 72.2%, respectively), indicating its ability to recognize sarcastic statements. On the other hand, both BiLSTM-based models score slightly lower compared to BERT, although they are still able to recognize sarcasm with decent accuracy. These results also indicate the great potential of the BERT-based model in detecting sarcasm in the real world; however, it can also be noted that other algorithms gave slightly worse results.

6. Conclusion

Sarcasm, characterized by the deliberate use of words to express the opposite of their literal meaning, represents a complex form of communication. Its predominantly spoken nature poses significant challenges for computational detection. However, this study presents a robust framework for detecting sarcasm in social media comments. The findings have

Figure 10 Second deep learning model architecture



Source: Made by the authors

Table 3 Summary of results for deep learning models

Model	Accuracy (%)	F1 (%)	Recall (%)	Precision (%)
BiLSTM 1	68.0	67.7	68.3	67.1
BiLSTM 2	67.0	68.1	67.2	65.7
BERT-based	*73.1*	*72.4*	*71.3*	*72.2*

Note: Italic are highest values per column
Source: Made by the authors

practical implications in various domains, including sentiment analysis, online reputation management and customer service. By accurately identifying instances of sarcasm, this framework contributes to enhancing the understanding of nuanced communication patterns in online interactions and facilitates more effective decision-making in relevant applications.

This article described two ways to approach this task and the necessary data preparation steps. An automatically annotated data set containing data from an online forum was analysed, pre-processed and used as input for four standard machine learning models and three deep learning models. The performance of several machine learning models with near-default parameters was measured and compared to the performance of deep neural networks. It can be concluded that the performance of both kinds of models depends largely on text cleaning and pre-processing. The performance of machine learning models is also deeply dependent on manual feature extraction, while the performance of deep learning models depends mostly on the architecture itself. The performance of the ridge regression classifier was surprising, as it is not as prominent in literature as other models, in the context of text classification. Logistic regression was the worst-performing model by all performance measures. The fine-tuned BERT-based model outperformed machine learning models as well as both BiLSTM models.

Standard machine learning models could be improved by better feature extraction. For example, the combination of sentiment-related features, punctuation-related features, syntax-related features and pattern-related features yielded good results (Bouazizi and Otsuki, 2016). Deep learning models could be improved with more complex architecture. For example, multiple bidirectional LSTM layers could be added to better capture the contextual information in the text. Although the BERT-based model showed promising results in this study, further research can be conducted to evaluate other

deep learning models and compare their performance with the BERT-based model. This can help to identify the most effective deep learning model for sarcasm detection. The data set is also massive, so more learning epochs could be beneficial for the automatic feature extraction process.

Sarcasm can be domain-specific, meaning that it can differ depending on the context or topic being discussed (Potamias *et al.*, 2020). Future research can explore the development of domain-specific sarcasm detection models that can accurately identify sarcasm in specific domains, such as politics or entertainment (Marijić and Bagić Babac, 2023). Moreover, contextual factors such as cultural disparities, social norms and the speaker's intention play a crucial role in the interpretation of sarcasm. Investigating how these contextual elements influence the performance of machine and deep learning models in sarcasm detection would be an essential avenue for future research. Understanding the impact of context on the effectiveness of these models can help enhance their robustness and applicability in real-world scenarios.

Note

1 www.kaggle.com/datasets/danofer/sarcasm

References

Arora, A. (2020), "Sarcasm detection in social media: a review", Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2021", https://ssrn.com/abstract=3749018 or doi: 10.2139/ssrn.3749018

Ashwitha, A.S.G.S.H.R., Upadhyaya, A.P. and Ray, P.M.T.C. (2021), "Sarcasm detection in natural language processing", *Materials Today: Proceedings*, Vol. 37, pp. 3324-3331, doi: 10.1016/j.matpr.2020.09.124.

Avdić, D. and Bagić Babac, M. (2021), "Application of affective lexicons in sports text mining: a case study of FIFA World Cup 2018", *South Eastern European Journal of Communication*, Vol. 3 No. 2, pp. 23-33.

Bagić Babac, M. (2022), "Emotion analysis of user reactions to online news", *Information Discovery and Delivery*, doi: 10.1108/IDD-04-2022-0027.

Bagić Babac, M. and Podobnik, V. (2016), "A sentiment analysis of who participates, how and why, at social media sports websites: how differently men and women write about football", *Online Information Review*, Vol. 40 No. 6, pp. 814-833, doi: 10.1108/OIR-02-2016-0050.

Bandhakavi, A., Wiratunga, N., Massie, S. and Padmanabhan, D. (2017), "Lexicon generation for emotion detection from text", *IEEE Intelligent Systems*, Vol. 32 No. 1, pp. 102-108.

Băroiu, A.-C. and Trăușan-Matu, Ș. (2022), "Automatic sarcasm detection: systematic literature review", *Information*, Vol. 13 No. 8, p. 399, doi: 10.3390/info13080399.

Bird, S., Klein, E. and Loper, E. (2009), *Natural Language Processing with Python*, O'Reilly Media.

Bouazizi, M. and Otsuki, T. (2016), "A pattern-based approach for sarcasm detection on Twitter", *IEEE Access*, Vol. 4, pp. 5477-5488.

Brzić, B., Botički, I.B. and Babac, M. (2023), "Detecting deception using natural language processing and machine learning in datasets on COVID-19 and climate change", *Algorithms*, Vol. 16 No. 5, p. 221, doi: 10.3390/a16050221.

Buschmeier, K., Cimiano, P. and Klinger, R. (2014), "An impact analysis of features in a classification approach to irony detection in product reviews", *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis. Baltimore, MD*, pp. 42-49.

Cvitanović, I. and Bagić Babac, M. (2022), "Deep learning with self-attention mechanism for fake news detection", in Lahby, M., Pathan, A.-S., Maleh, Y.K. and Yafooz, W.M.S. (Eds), *Combating Fake News with Computational Intelligence Techniques*, Springer, Switzerland, pp. 205-229, doi: 10.1007/978-3-030-90087-8_10.

Davidov, D., Tsur, O. and Rappoport, A. (2010), "*Semi-supervised recognition of sarcastic sentences in twitter and amazon*", *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pp. 107-116.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019), "*BERT: pre-training of deep bidirectional transformers for language understanding*", *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, MN, USA, pp. 2-7, June 2019.

Elbagir, S. and Yang, J. (2019), "Analysis using natural language toolkit and VADER sentiment", *Proceedings of the International MultiConference of Engineers and Computer Scientists 2019*, China, Hong Kong.

Ghosh, D. and Veale, T. (2017), "*Fracking sarcasm using neural network*", *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 439-444.

Ghosh, D., Fabbri, A.R. and Muresan, S. (2018), "Sarcasm analysis using conversation context", *Computational Linguistics*, Vol. 44 No. 4, pp. 755-792, doi: 10.1162/coli_a_00336.

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep Learning, Adaptive Computation and Machine Learning Series*, MIT Press, London, England.

Gupta, R., Kumar, J. and Agrawal, H. (2020), "A statistical approach for sarcasm detection using twitter data", *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, *IEEE*, pp. 633-638.

Hazarika, D., Poria, S., Gorantla, S., Cambria, E., Zimmermann, R. and Mihalcea, R. (2018), "Cascade: contextual sarcasm detection in online discussion forums", *Proceedings of the 27th International Conference on Computational Linguistics*, Association for Computational Linguistics, Santa Fe, NM, pp. 1837-1848.

Joshi, A., Bhattacharyya, P., Carman, M.J., Saraswati, J. and Shukla, R. (2016), "How do cultural differences impact the quality of sarcasm annotation? A case study of Indian annotators and American text", LaTeCH@ACL.

Jurafsky, D. and Martin, J.H. (2000), *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice-Hall, Upper Saddle River, NJ.

Kampić, M. and Bagić Babac, M. (2021), "Sentiment analysis of president trump's tweets: from winning the election to the fight against COVID-19", *Communication Management Review*, Vol. 6 No. 2, pp. 90-111, doi: 10.22522/cmr20210272.

Khatri, A. and Pranav, P. (2020), "Sarcasm detection in tweets with BERT and GloVe embeddings", *Proceedings of the Second Workshop on Figurative Language Processing*, pp. 56-60, *Online. Association for Computational Linguistics*.

Khodak, M., Saunshi, N. and Vodrahalli, K. (2018), "A large self-annotated corpus for sarcasm", *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan.

Kostelej, M. and Bagić Babac, M. (2022), "Text analysis of the harry potter book series", *South Eastern European Journal of Communication*, Vol. 4 No. 1, pp. 17-30.

Kumar, A. and Anand, V. (2020), "Transformers on sarcasm detection with context", *Proceedings of the Second Workshop on Figurative Language Processing. Virtual event*, pp. 88-92.

Kumawat, D. and Jain, V. (2015), "POS tagging approaches: a comparison", *International Journal of Computer Applications*, Vol. 118 No. 6, pp. 32-38.

Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P.S. and He, L. (2022), "A survey on text classification: from traditional to deep learning", *ACM Transactions on Intelligent Systems and Technology*, Vol. 13 No. 2, Article 31, p. 41, doi: 10.1145/3495162.

Manning, C.D., Raghavan, P. and Schütze, H. (2008), *Introduction to Information Retrieval*, Cambridge University Press, Cambridge, England.

Marijić, A. and Bagić Babac, M. (2023), "Predicting song genre with deep learning", *Global Knowledge, Memory and Communication*, doi: 10.1108/GKMC-08-2022-0187.

Misra, R. and Arora, P. (2023), "Sarcasm detection using news headlines dataset", *AI Open*, Vol. 4, pp. 13-18.

Musso, I.M.-B. and Bagić Babac, M. (2022), "Opinion mining of online product reviews using a lexicon-based algorithm", *International Journal of Data Analysis Techniques and Strategies*, Vol. 14 No. 4, pp. 283-301.

Nayel, H., Amer, E., Allam, A. and Abdallah, H. (2021), "Machine learning-based model for sentiment and sarcasm detection", *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, Association for Computational Linguistics, Kyiv, Ukraine (Virtual), pp. 386-389.

Pant, K. and Dadu, T. (2020), "Sarcasm detection using context separators in online discourse", arXiv preprint arXiv:2006.00850.

Parameswaran, P., Trotman, A., Liesaputra, V. and Eyers, D. (2021), "BERT's the word: sarcasm target detection using BERT", *Proceedings of the 19th Annual Workshop of the Australasian Language Technology Association*, Online. Australasian Language Technology Association, pp. 185-191.

Poch Alonso, R. and Bagić Babac, M. (2022), "Machine learning approach to predicting a basketball game outcome", *International Journal of Data Science*, Vol. 7 No. 1, pp. 60-77.

Poria, S., Cambria, E., Hazarika, D., Vij, P. and Hussain, A. (2016), "*A deeper look into sarcastic tweets using deep convolution neural networks*", *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1969-1980.

Potamias, R.A., Siolas, G. and Stafylopatis, A. (2020), "A transformer-based approach to irony and sarcasm detection", *Neural Computing and Applications*, Vol. 32 No. 23, pp. 17309-17320.

Puh, K. and Bagić Babac, M. (2023), "Predicting stock market using natural language processing", *American Journal of Business*, Vol. 38 No. 2, pp. 41-61, doi: 10.1108/AJB-08-2022-0124.

Ren, L., Xu, B., Lin, H., Liu, X. and Yang, L. (2020), "Sarcasm detection with sentiment semantics enhanced multi-level memory network", *Neurocomputing*, Vol. 401, pp. 320-326.

Reyes, A., Rosso, P. and Buscaldi, D. (2012), "From humor recognition to irony detection: the figurative language of social media", *Data & Knowledge Engineering*, Vol. 74, pp. 1-12.

Riloff, E., Qadir, A., Surve, P., De Silva, L., Gilbert, N. and Huang, R. (2013), "Sarcasm as contrast between a positive sentiment and negative situation", in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, *Association for Computational Linguistics*, *Seattle*, *Washington*, pp. 704-714.

Saha, S., Yadav, J. and Ranjan, P. (2017), "Proposed approach for sarcasm detection in twitter", *Indian Journal of Science and Technology*, Vol. 10 No. 25, pp. 1-8.

Savini, E. and Caragea, C. (2022), "Intermediate-task transfer learning with BERT for sarcasm detection", *Mathematics*, Vol. 10 No. 5, p. 844, doi: 10.3390/math10050844.

Scola, E. and Segura-Bedmar, I. (2021), "Sarcasm detection with BERT", *Procesamiento Del Lenguaje Natura*, Vol. 67, pp. 13-25.

Sharma, D.K., Singh, B., Agarwal, S., Pachauri, N., Alhussan, A.A. and Abdallah, H.A. (2023), "Sarcasm detection over social media platforms using hybrid ensemble model with fuzzy logic", *Electronics*, Vol. 12 No. 4, p. 937, doi: 10.3390/electronics12040937.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser. and Polosukhin, I.Ł (2017), "Attention is all you need", *Advances in Neural Information Processing Systems*, Vol. 30, pp. 5998-6008.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. and Bowman, S.R. (2018), "*GLUE: a multi-task benchmark and analysis platform for natural language understanding*", *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Vol. 1, pp. 353-355.

Zhang, Y. and Wallace, B. (2018), "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification", arXiv preprint arXiv:1510.03820.

Further reading

Puh, K. and Bagić Babac, M. (2022), "Predicting sentiment and rating of tourist reviews using machine learning", *Journal of Hospitality and Tourism Insights*, doi: 10.1108/JHTI-02-2022-0078.

Corresponding author

Marina Bagić Babac can be contacted at: marina.bagic@fer.hr