Title	Reading Between the Lines: A NLP approach to Sarcasm Detection
Members	Sion Chun (sc3791), Meyhar Sharma (ms7070), Tejas Chhotulal Badgujar (tcb2145), Ritayan Patra (rp3247)
Course	Applied Machine Learning (COMS4995W032)
Description	Building a NLP based model to detect sarcastic and non-sarcastic news headlines

1. Introduction

Sarcasm is tricky to understand and even harder to identify using computers in Natural Language Processing (NLP). It often relies on special ways of saying things and a deep knowledge of the language, which even humans sometimes struggle with. Detecting sarcasm is especially important in public communication, like news headlines. Knowing if a headline is sarcastic or not can help understand the writer's true intent and what the article might be about. It also affects how many people decide to read the article based on the headline. This project aims to create a machine learning model using NLP to figure out if a news headline is sarcastic or not (a binary classification).

2. About the Data

The JSON data collected comes from an overly sarcastic, satirical news source (The Onion) and a straightforward, likely non-sarcastic news source (HuffPost) and can be found here — https://www.kaggle.com/datasets/rmisra/news-headlines-dataset-for-sarcasm-detection.
Each record has three pieces of information:

- ➔ **article_link**: A link to the original news article. You can use it to find more details.
- ➔ **headline**: The title of the news article.
- ➔ **is_sarcastic**: This shows if the news is sarcastic. It's **1** if sarcastic, and **0** if not.

3. Exploratory Data Analysis

We analyzed the dataset to see patterns and trends. There are no missing values in the data. The dataset has 14,985 (56%) non-sarcastic records and 11,724 (44%) sarcastic records. Both classes have enough data, and the dataset is balanced. The dataset is divided into train and test dataset. In the training set, there are about 12,000 non-sarcastic samples and 9,000 sarcastic samples, indicating a slight imbalance. Both classes are well-represented in both sets, supporting effective model evaluation.

We have applied a number of preprocessing steps. The preprocessing steps aim to clean and simplify text data for better machine learning performance. First, basic cleaning is performed to remove anything that isn't a letter or number, such as special symbols, spaces, or punctuation. Numbers are also removed, and the text is converted to lowercase to ensure uniformity. This step ensures that only meaningful information remains in the data. Next, stopwords like "the," "is," and "and," which don't contribute much meaning to the text, are removed to focus on the more important words. After that, punctuation marks, such as periods, commas, and exclamation points, are stripped away. Finally, lemmatization is applied to reduce words to their base or root forms. For instance, words like "running," "runs," and "ran" are all converted to "run." This process helps the model understand that these variations of the word have the same meaning. Together, these steps simplify the text and make it more consistent.

For data vectorization, Bag-of-Words and TF-IDF with n-grams are compared through analysis of feature selection and performance of XGBoost. Both methods identify "Trump" as the most prominent term, followed by "new," "man," "get," and others. TF-IDF, which emphasizes unique and contextually significant terms by considering their frequency within a document corpus, shows slight differences in importance rankings. For instance, "woman" is assigned higher importance than "find," reflecting its contextual significance. For classical machine learning models, TF-IDF is chosen due to its ability to emphasize unique and contextually relevant terms, making it effective for interpreting prominent words and themes within the dataset. For deep learning models, the texts are tokenized and encoded into

integer sequences, allowing the retention of word order to enable models to capture sequential patterns and contextual relationships within the text.

4. Machine Learning and Deep Learning Models

a. Classical Models

During the classical modelling phase, we explored ten different machine learning algorithms including logistic regression, decision trees, random forest, and AdaBoost, selected to represent a diverse range of approaches for text classification. To identify the best configurations for each model, we conducted a randomized search across carefully defined parameter grids for each algorithm. This process incorporated five-fold cross-validation to ensure robust evaluation, using the F1 score as the primary metric to balance precision and recall. Among the tested models, Extra Trees emerged as the top performer, closely followed by Random Forest. These ensemble methods demonstrated better performance due to their ability to aggregate predictions from multiple decision trees, thereby improving both accuracy and resilience to overfitting. While Random Forests averages the predictions of individual trees, Extra Trees introduces an additional layer of randomness in the splitting criteria during tree construction, which not only reduces variance but also enhances generalization, particularly when the dataset contains subtle patterns or noise. These ensemble methods improve capturing complex, non-linear interactions within the data and handling high-dimensional TF-IDF vectors effectively, making them suitable for sarcasm detection where intricate word combinations and contextual nuances are critical.

b. LSTM Model

LSTM can be particularly helpful for sarcasm detection because it is well-suited for analyzing sequences of words in text. Sarcasm often relies on subtle patterns, such as word order, context, and tone, which LSTM can capture by maintaining memory over a sequence. By learning which parts of a headline are relevant to detect sarcasm (e.g., unexpected word combinations or shifts in tone), LSTM can identify nuanced patterns that simpler models might miss. Additionally, its ability to retain context over longer sequences makes it ideal for understanding the meaning of entire headlines. There are two LSTM models that we trained for our objective, a brief description on how these models work is- the sequence of words is fed into the LSTM one word (or token) at a time in the order they appear. LSTM then updates the hidden state after processing each token. In this step both past and current context is captured, enabling the model to understand the sentiment nuances across the headline. After processing the entire sequence, the LSTM's final hidden state represents the sentiment-related features of the text which is passed to a fully connected layer (dense layer) with a sigmoid activation to give the final prediction.

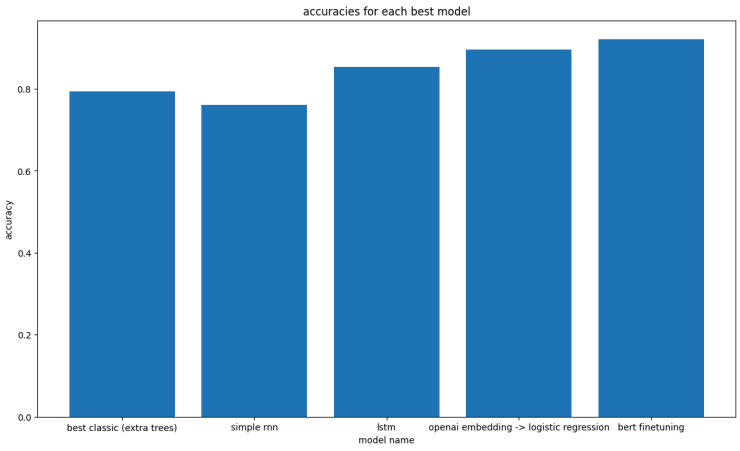
c. Pretrained Models

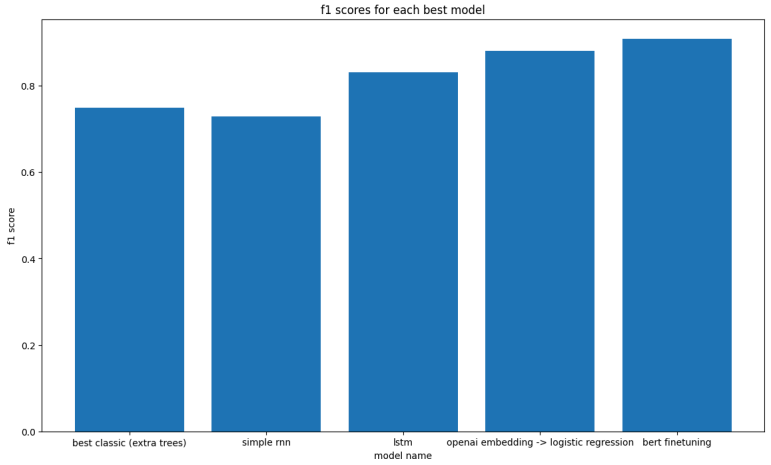
For the pretrained models, we explored two approaches: using a pretrained semantic embedding model (specifically, OpenAI's `text-embedding-3-small` model) connected to a basic Logistic Regression classifier and fine tuning a BERT-based model on our classification problem. The advantages of using such pretrained models lie in the fact that they are usually trained sufficiently on very large and varied datasets with some domain, giving them strong performance. For our objectives, we train a classification head in the semantic embedding, we train a supplementary Logistic Regression classifier, and in the BERT-based case, we train the final linear layer of the transformer model.

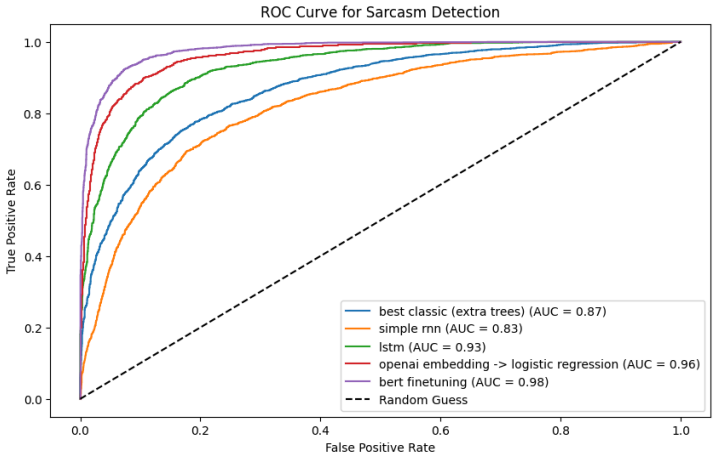
As a summary for how the semantic embedding model works, it takes in a string of text as input and then returns a 1536-length float vector corresponding to its semantic meaning. In general, if two strings of text are very close to each other in meaning, then their corresponding semantic vector representation as part of this model will also be close to each other (in terms of their cosine similarities). For our purpose, we embedded all of the raw headlines into semantic vectors and then trained a logistic regression classifier for sarcasm detection. As a summary for how the BERT based model works, it is based on the transformer architecture. Similar to the aforementioned OpenAI model, BERT is able to, given a string of text, encode a representation of the text as a whole as an embedding. However, contrasting to the semantic representation found through the OpenAI model, we hypothesize that the BERT-based representation is a bit more general which may lead to better performance on our downstream task compared to using a model specifically tuned for

semantic-based objectives. For our purpose, we train BERT on our specific dataset by freezing all of the weights except for the last linear layer of the neural network.

5. Results








```
HEADLINE: women in business q&a: laura tenison, founder and managing director, jojo maman
IS LABELED SARCASTIC?: NO
BERT MODEL PREDICTION: NO

HEADLINE: native american activists create spoof website to call for redskins name change
IS LABELED SARCASTIC?: NO
BERT MODEL PREDICTION: NO

HEADLINE: sixth-grader's family tree fails to hold up to scrutiny
IS LABELED SARCASTIC?: YES
BERT MODEL PREDICTION: YES

HEADLINE: harry reid: republicans are 'acting as puppets for the nra'
IS LABELED SARCASTIC?: NO
BERT MODEL PREDICTION: NO

HEADLINE: alderman has that zoning dream again
IS LABELED SARCASTIC?: YES
BERT MODEL PREDICTION: YES
```

The performance of the best models in each category (classification trees, LSTM and pre-trained) was compared to discover both the pretrained models outperformed the classical model and neural nets with the best model performance shown by BERT. From the top left and top right charts, accuracy and F1 score of the BERT model is noted close to 0.93 with an AUC score of 0.98 (bottom left chart), this is ideal since it is close to 1, indicating the model has an excellent balance between sensitivity (true positive rate) and specificity (true negative rate). The bottom right image shows a small example set of predictions made by the BERT model.

6. Conclusion

In summary, our project explores a variety of possible approaches for solving a natural language-based objective in machine learning. Specifically, we explored and assessed the use of bag-of-words and TF-IDF featurization schemes with the use of classical machine learning classification models, the use of tokenized-based, from-scratch neural network approaches, and the use of various pretrained models commonly found in the natural language space. In terms of our specific objective of sarcasm detection, we were able to successfully apply these methodologies to our dataset to achieve satisfactory results.

In the future, our work can be extended to combine text with more modalities such as images, videos, or audio to detect sarcasm in news in memes and video formats as well. The models can also be extended to work with multiple languages, as sarcasm often involves cultural and linguistic nuances, to tackle this more data for various languages and cultures can be incorporated in training.