School of Electronic Engineering and Computer Science

Final Report

Programme of study:
BSc Computer Science

Project Title:

Using pre-processing techniques and machine learning models to assess the funniness of edited news headlines

Supervisor:
Julia Ive

Final Year Undergraduate Project 2022/23

Student Name:
Hoi Pui Leung

Queen Mary University of London

Date: 01/05/23

Table of Contents

1. Introduction

Humour is present and vital to our everyday lives both online and off, it is a crucial component of human communication and interaction that helps people in many forms. With the rapid growth of digital media and the ubiquity of user-generated content, the ability to automatically detect and predict humour in text has become an increasingly relevant area of research. In this research paper, I delve into the captivating world of humour by exploring the use of pre-processing techniques and machine learning models to assess the funniness of edited news headlines, aiming to contribute to the growing body of knowledge on humour detection and prediction.

1.1 Motivation

The rapid expansion of social media platforms and online communication channels has led to an enormous increase in user-generated content, with humour being a prevalent component of these interactions. Developing effective techniques to automatically assess and predict humour in text can have wide-ranging applications, such as content recommendation, sentiment analysis, and personalised advertising.

Secondly, humour detection has recently gained particular importance as an integral part of the pipeline for crucial social media tasks, such as hate speech detection (Meaney et al., 2021). Identifying humour is essential for distinguishing between malicious content and innocent jokes. Accurate humour detection can help prevent the misclassification of content and controversy, ensuring that online platforms remain safe and inclusive while preserving the users' freedom of expression.

In summary, the motivation behind this research paper lies in the potential applications and benefits of humour detection and prediction, as well as the desire to contribute to the understanding of humour as a fascinating aspect of human communication. By investigating the funniness of edited news headlines using pre-processing techniques and machine learning models, I hope that the results of this study can potentially serve as a foundation for future research in this domain, contributing to this growing field.

1.2 Aim and Objectives

The aim of this project is to investigate the contribution of different sets of features to the performance of machine learning models, one of them being character level features. This will be done through the investigation of different techniques and their application to different models.

This project has the following objectives:

- Research what has been done on the subject of automatic humour detection, particularly the pre-processing techniques and models used.
- Understand natural language processing (NLP) classification, pre-processing techniques such as lemmatisation and segmentation into subwords/characters, and machine learning models such as linear regression, support vector regression, and random forest regression.
- Analyse and improve the existing techniques and models for humour detection.
- Investigate the importance of the standard sets of features relative to the performance of the task
- Propose different sets of features and investigate their contribution via ablation studies
- Evaluate what the results of this project could be used for.

1.3 Report Structure

The report is structured in a way that is clear for the reader to follow, as well as in order of how I approached the tasks. The project will be split into the following chapters:

Introduction

In this section, I will present an overview of the project I have embarked upon. This will include my initial understanding prior to undertaking the project, the motivations behind choosing this particular topic, and the ways in which this project can address a real-world issue and contribute valuable research.

Literature Review

The literature review is used to analyse what has already been done around the topic of humour detection and prediction with regards to natural language processing and machine learning, this section will delve into a more comprehensive examination of the project's associated areas.

Machine Learning Algorithms

Here I explain in detail the 3 machine learning algorithms I have chosen for this project, how the algorithms work, and why I chose specifically 3 for this task.

Experimental Training

This section contains explanations and analysis of the dataset, pre-processing techniques used, features, as well as the training procedure of my models.

Experimental Results

This section will briefly explain the evaluation metrics used to compare the models, then perform a qualitative assessment and ablation study of the results I have obtained.

Ethics

Here I delve into the potential social and ethical implications of developing and deploying AI systems, as well as possible future improvements on my system.

Conclusion

This section will be used to consolidate all the information and insights gathered from this project. Additionally, I will discuss valuable lessons learned that could be beneficial for future implementations.

References

A comprehensive list of all the references cited in the report, formatted according to the Harvard referencing system.

2. Literature Review

Humour detection is a challenging task in NLP due to the subjective nature of humour and the difficulty in defining it. Researchers have proposed various approaches to detect humour in text, including rule-based, feature-based, and machine learning-based methods. Rule-based methods rely on predefined rules to identify humour, such as the use of puns or sarcasm. Feature-based methods extract linguistic features from text, such as sentiment, syntax, and semantics, and use them to train a classifier to detect humour. Machine learning-based methods use algorithms such as support vector machines (SVMs), decision trees, and neural networks to learn patterns in data and classify text as humorous or not.

In this section, I will examine and review the research I have done on the literature surrounding the topic of humour detection and evaluation. Humour detection as an AI task has been a topic under research for some time, with corpora from Mihalcea and Strapparava (2005) on how humour recognition can be successfully achieved by using automatic classification techniques, to research done in 2021 by Patel et al. (2021) on using a model based on Long Short-Term Memory (LSTM) to identify humour in sentences.

2.1 Humour Detection

There exists a huge range of studies and papers already on the topic due to its large scope and endless sub-topics that can be further explored, the numerous different methodologies and datasets used to conduct these researches combine to result in a large array of conclusions on the subject matter. While humour perception is a psychological phenomenon that is subjective and unpredictable, computerised humour detection research can be said to be the same way.

“Humour detection facilitates the understanding of underlying relations of words and how various combinations of strings can invoke laughter.” (Patel et al., 2021). Patels’ definition of humour detection is also coupled with the context of their understanding of what they believe is the formula to a humorous joke, a general structure that contains a punchline and contextual information leading up to the punchline. This specific paper uses techniques such as deep learning and sentiment analysis to train a Recurrent Neural Networks (RNN) based model - LSTM to ultimately provide a binary outcome of if the given statement is humorous or not (Patel et al., 2021). The impressive part of this research model is not just the excellent accuracy of 94.62%, but also the implementation of the selection of semantic attributes for humour recognition.

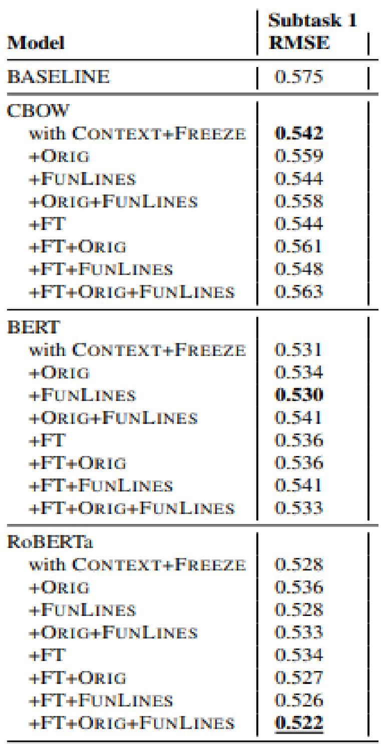
In another paper by Yang, Hooshmand and Hirschberg (2021) more focused on the study of humour on social media, a framework for Collecting Humor Reaction Labels (CHoRaL) was proposed to automatically collect humour reaction labels on Facebook posts, then using the naturally available user reactions to provide both binary labels and continuous humour scores (Yang, Hooshmand and Hirschberg, 2021). After collecting massive amounts of Facebook posts to form an unbiased 785,000 posts dataset and defining a humour score based on Facebooks' built-in reactions, detailed lexico-semantic and affective analysis was then performed to understand the delivery of humour further, and in turn, further optimising humour detection. The CHoRaL framework and dataset lay the groundwork for further development of humour detection models, with the potential for broader applications, such as distinguishing malicious misinformation posts and non-malicious humorous posts (Yang, Hooshmand and Hirschberg, 2021).

Several studies have also explored the use of NLP techniques to detect humour on social media platforms such as Twitter and Reddit. For example, a study by Istaiteh et al. (2020) used a machine learning-based approach to detect racist and sexist hate speech on Twitter. The study used a dataset of tweets labelled as sexist or racist and extracted features such as n-grams, part-of-speech tags, and sentiment to train a classifier to detect hate speech. Another study by Gaikwad et al. (2021) conducted a systematic literature review of online extremism detection and highlighted the need for better datasets and validation techniques to detect extremism on social media platforms.

One of the main inspirations for this paper is the “SemEval-2020 Task 7: Assessing Humor in Edited News Headlines” (Hossain et al., 2020). The primary objective is to automatically compute the humour value of edited news headlines that were produced by humans using a micro-edit, inserting a single-word noun or verb to replace an existing entity or single-word noun or verb in the original headline, the dataset for this task is later explained in section 4.1. Sub-task 1: Regression required participants to predict the mean funniness of the edited headline, given the original and edited headline. While this task focused on sentence-level features like the replacement of a single word, I'm interested in investigating the relationship between character-level features and the humour score.

2.2 Models Used for the SemEval Task

In this section, I will review some of the models provided by the organisers of the task, as well as some of the top-performing models created by participants. The organisers first provided several benchmarks for participating systems to compare against, these benchmark results produced by different models are trained using the edited headlines and then performed on the test set.



Model	Subtask 1 RMSE
BASELINE	0.575
CBOW	
with CONTEXT+FREEZE	**0.542**
+ORIG	0.559
+FUNLINES	0.544
+ORIG+FUNLINES	0.558
+FT	0.544
+FT+ORIG	0.561
+FT+FUNLINES	0.548
+FT+ORIG+FUNLINES	0.563
BERT	
with CONTEXT+FREEZE	0.531
+ORIG	0.534
+FUNLINES	**0.530**
+ORIG+FUNLINES	0.541
+FT	0.536
+FT+ORIG	0.536
+FT+FUNLINES	0.541
+FT+ORIG+FUNLINES	0.533
RoBERTa	
with CONTEXT+FREEZE	0.528
+ORIG	0.536
+FUNLINES	0.528
+ORIG+FUNLINES	0.533
+FT	0.534
+FT+ORIG	0.527
+FT+FUNLINES	0.526
+FT+ORIG+FUNLINES	**<u>0.522</u>**

Table 1: Benchmarks on the test set. The best result in each model is in **bold**, while the overall best is <u>underlined</u> (Hossain et al., 2020)

Baseline: This is the baseline model created by assigning the mean rating from the training set, this simple model is explained later in section 5.2.

CBOW: CBOW refers to the continuous bag-of-words (CBOW) model architecture proposed by Mikolov et al. (2013). It is a neural network for NLP tasks “based on predicting a target word given the context of the surrounding words.” (Utkarsh, 2023). In this instance, the CBOW architecture is used with the Global Vector (GloVe) model (Pennington, Socher and Manning, 2014), this model generates context-independent word representations as an initialization for the weights of the CBOW model.

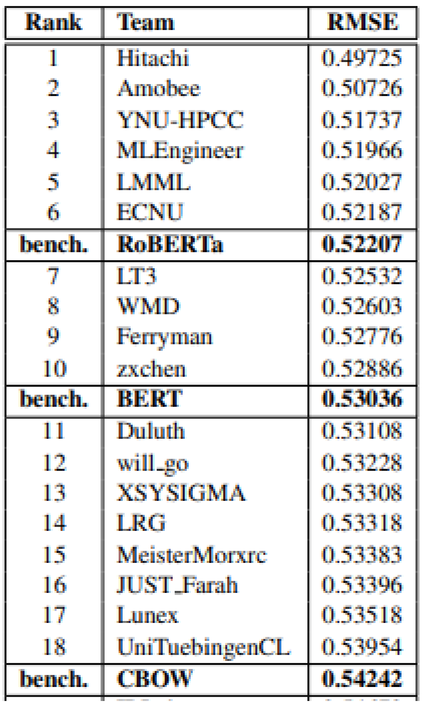
BERT: Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) is a pre-trained language model that can capture the bidirectional context in text, meaning that it is able to understand the meaning of words more accurately. Hossain et al. (2020) used this as a base to create a regressor for the task. BERT has also been used for humour assessment in tweets and jokes (Mao and Liu, 2019).

RoBERTa: A Robustly Optimised BERT Pretraining Approach (RoBERTa) (Liu et al., 2019) is an optimised variant of BERT that improves its performance by modifying the pretraining procedure. It involves training the model for longer, with bigger batches, and over more data. As shown in Table 1, RoBERTa performs slightly better than BERT across the board.

2.2.1 Participating Systems

Results from the competition show that the dominant teams made use of an ensemble of pre-trained language models (PLM) such as BERT, RoBERTa, and GPT-2 (Radford et al., 2019), as well as context-independent word embeddings such as Word2Vec (Mikolov et al., 2013) and GloVe word vectors.

The top-performing system, Hitachi (Morishita et al., 2020) exploited an ensemble of PLMs, with their model utilising 700 base models, each being a different combination of PLMs, sets of hyperparameters and cross-validation folds. Their strategy was to employ Stacking at Scale (SaS), first fine-tuning the numbers of PLMs and then applying a stacking ensemble on top of them.



Rank	Team	RMSE
1	Hitachi	0.49725
2	Amobee	0.50726
3	YNU-HPCC	0.51737
4	MLEngineer	0.51966
5	LMML	0.52027
6	ECNU	0.52187
bench.	**RoBERTa**	**0.52207**
7	LT3	0.52532
8	WMD	0.52603
9	Ferryman	0.52776
10	zxchen	0.52886
bench.	**BERT**	**0.53036**
11	Duluth	0.53108
12	will_go	0.53228
13	XSYSIGMA	0.53308
14	LRG	0.53318
15	MeisterMorxrc	0.53383
16	JUST_Farah	0.53396
17	Lunex	0.53518
18	UniTuebingenCL	0.53954
bench.	**CBOW**	**0.54242**

Table 2: Official results from participating systems and benchmarks (Hossain et al., 2020)

Another participating system, namely Duluth (Jin et al., 2020), approached the task with the hypothesis that “an edited headline is funny if the edited words are semantically distant from the context words or the original words.” (Jin et al., 2020). They then built on this hypothesis by using BERT, RoBERTa and CBOW models, concluding that while BERT and RoBERTa had similar results, CBOW performed slightly worse, showing evidence “that contextual information is essential for humor detection.” (Jin et al., 2020)

While some of the systems proposed by participants outperform the benchmarks produced by existing models and pave the way for future research surrounding humour detection and evaluation. Many of these systems consist of very complex methods and computationally expensive approaches, which leads to long training times and a lack of interpretability, making it challenging to understand the reasons behind their predictions. Therefore, in this investigation I will focus on more simple methods that can be explained and interpreted easily.

2.3 Features

Within the realm of textual humour detection, different studies focus on extracting different features from the corpora for their investigations. These features encompass various aspects of language use, meaning, structure, context, and cognitive processes. In this section, I will provide an overview of the existing literature on these features.

2.3.1 Lexical Features

For a semantic evaluation task titled “SemEval-2021 Task 7: HaHackathon: Detecting and Rating Humor and Offense.” Raha et al. (2021) approached the task with humour detection and rating, they employed a set of lexical features to train their models. Ranging from “the total number of letters, punctuation, upper case letters and numbers within the text.”, to detecting the number of and identifying the kind of personal pronouns in the text. However, the reliance on lexical features alone may not suffice in detecting humour, as it often involves more subtle and complex aspects of language that go beyond simple word usage.

2.3.2 Semantic Features

Semantic features refer to the meanings and relationships between words and phrases in a text. Mihalcea and Strapparava (2005) proposed an approach to humour detection based on semantic similarity and incongruity, leveraging measures like WordNet-based similarity to determine the degree of relatedness between concepts in a text. Semantic features have also been used to capture various forms of humour, such as irony (Veale and Hao, 2010), sarcasm (Barbieri et al., 2014), and satire, which rely on the unexpected or contradictory relationships between words and their intended meanings. Despite their importance, semantic features can be challenging to extract and quantify, given the ambiguity and context dependence of language.

2.3.3 Sentiment Features

Sentiment features involve the emotional content of a text and are particularly relevant to humour detection, as humorous texts often exhibit a positive sentiment or a mix of positive and negative sentiments. Sentiment analysis techniques, such as lexicon-based approaches or machine learning models, have been used to derive sentiment scores from texts and identify humour based on the observed sentiment patterns. Davidov, Tsur, and Rappoport (2010) employed sentiment features in their study on semi-supervised recognition of sarcastic sentences on Twitter and Amazon. Although sentiment features can provide valuable information about the emotional content of a text, their effectiveness in humour detection may be limited by factors such as the subtlety of emotions, linguistic nuances, and cultural differences.

2.4 Pre-processing

As with all NLP tasks, the data needs to be pre-processed before it can be used in any sort of model, in this scenario text normalisation needs to be performed on the dataset. As an example, in a study done by Chandrasekar and Qian (2016), they researched and identified the impact of preprocessing the dataset on the performance of a Naive Bayes Classifier, the conclusion of which shows that by using preprocessing techniques compared to not using them, the prediction accuracy of the classifier can be improved.

Uysal and Gunal (2014) state the four common preprocessing techniques include tokenization, stop-word removal, lowercase conversion, and stemming. **Tokenization** refers to the procedure of separating a text into words or phrases, otherwise known as tokens, typically done by segmenting alphabetic or alphanumeric characters with non-alphanumeric characters such as punctuations, whitespaces, etc.

Stop-word removal refers to the removal of common words which do not carry much meaning and are deemed to be irrelevant, this can help reduce the vocabulary size. However, the context of the scenario is important as a stop-word for one classifier might not be a stop-word for another, stop-word removal must not change the meaning of the sentence/phrase.

Lowercase conversion refers to converting all uppercase characters to their lowercase forms, while this reduces our vocabulary size, it could potentially change the meaning of words such as names. To combat this, Part of Speech (POS) taggers can be used to categorise words as nouns, verbs, adjectives etc. to preserve meaning. However, in our specific case and context which is humour, I believe that leaving the POS taggers out could actually be an advantage.

Stemming is used to obtain the root of words from their derived forms, however, this could often lead to words not looking like words depending on the specific stemming algorithm being used. For example, if the Porter (1980) algorithm is used, single could end up as singl, and accurate could end up as accur.

Many studies have been conducted on the effectiveness of different preprocessing techniques on different datasets for different purposes, Song, Liu and Yang (2005) concluded that while the impact of stop-word removal and stemming is small, the application of it can increase the efficiency of the text classifier. Toman, Tesar and Jezek (2006) stated that in most cases stop-word removal improves the classification accuracy as well as compresses the classification time, they also concluded that “The best preprocessing approach for text classification seems to be only to apply stop-words removal and to omit word normalization.” Their conclusion to omit word normalisation is due to the noticeable decrease in classification accuracy when word normalisation was applied. In this instance, their omission of word normalisation refers to the implementation of lemmatization and stemming algorithms.

2.5 Summary

I have analysed and identified that limited research is devoted to character-level analysis of humour

Humour detection in NLP is a complex and challenging task due to the subjective nature of humour. Various methods have been used to detect humour, including rule-based, feature-based, and machine learning-based approaches. The literature review explored different studies that employed these methods in various settings, such as social media platforms like

Facebook, Twitter, and Reddit. It also discussed the use of PLMs, like BERT and RoBERTa, in humour detection tasks. The review further examines different features used in humour detection, including lexical, semantic, and sentiment features, and the importance of pre-processing in NLP tasks.

Through this literature analysis, I have identified that limited research is devoted to humour detection at the character level, this will form the basis of my investigation.

3. Machine Learning Algorithms

3.1 Linear Regression

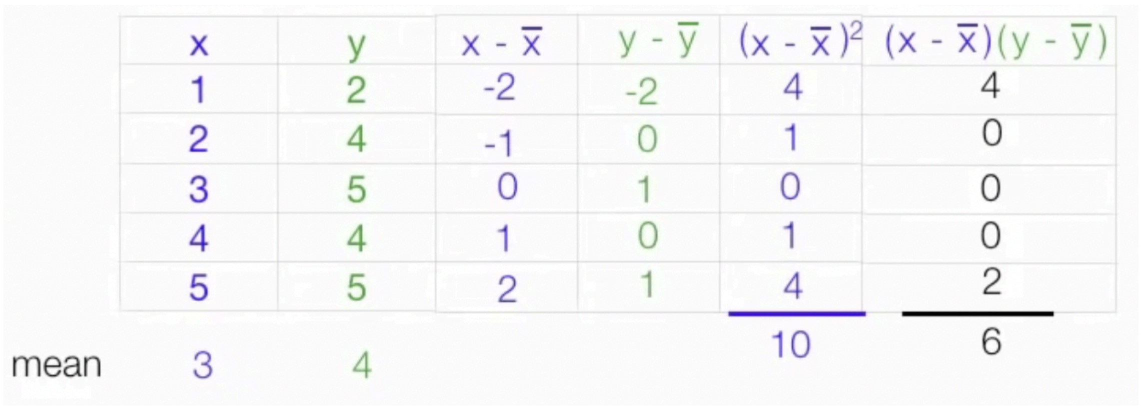
Linear regression is a supervised machine learning algorithm used for regression tasks by identifying relationships between variables and making predictions, it involves predicting a target value based on independent variables. The aim of linear regression is to find a line that can best represent the relationship between the input variables and the output variable. Line of best fit:

$$y = w_1 x + w_0$$

Where:

- **y** is the dependent variable
- **x** is the independent variable
- $\mathbf{w_1}$ and $\mathbf{w_0}$ are coefficients to be determined

Firstly, we must calculate the coefficients of the equation that represents the line of best fit. Once the coefficients are determined, predictions can be made by plugging new input data into the equation for the line of best fit. This can be done by using the squared loss function, for example,



	x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})^2$	$(x - \bar{x})(y - \bar{y})$
	1	2	-2	-2	4	4
	2	4	-1	0	1	0
	3	5	0	1	0	0
	4	4	1	0	1	0
	5	5	2	1	4	2
					10	6
mean	3	4				

Table 3: Visual explanation of the squared loss function (Longstreet, 2012)

Where:

- $\bar{x}$ is the mean of **x**
- $\bar{y}$ is the mean of **y**

The coefficient **w₁** can be found using the last 2 columns of the table:

$$w_0 = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sum(x-\bar{x})^2} = \frac{6}{10} = 0.6$$

The coefficient **w₀** can then be found by plugging in values that we know, using the mean value for **x** and **y**:

$$y = w_1 x + w_0$$

$$4 = 0.6(3) + w_0$$

$$w_0 = 2.2$$

Finally, this gives us a regression line with the equation:

$$y = 0.6x + 2.2$$

Linear regression is a fast and simple algorithm that is easy to understand and implement, often serving as a baseline for more complex algorithms, however, there are certain limitations when it comes to using linear regression. Mainly, it assumes a linear relationship between the input variables and the output variable, so depending on the nature of the problem and data it may not be the most suitable algorithm, it's also sensitive to outliers as it can significantly. impact the line of best fit, leading to inaccurate predictions (Russell and Norvig, 2010).

3.2 Support Vector Regression

Support Vector Regression (SVR) is a variant of the Support Vector Machine (SVM) algorithm used for regression tasks, it is a supervised machine learning algorithm used to predict continuous output variables based on one or more input variables (Cortes and Vapnik, 1995). The objective of SVR is to find a hyperplane in an N-dimensional space that best fits the training data; unlike linear regression, it can find non-linear relationships between variables depending on the kernel function used. This is particularly useful in our case as language data is often non-linear due to its complexity and different variations. One of the factors is ambiguity, words and phrases have multiple meanings depending on the context they are used in, making it difficult for language data to fit into linear models.

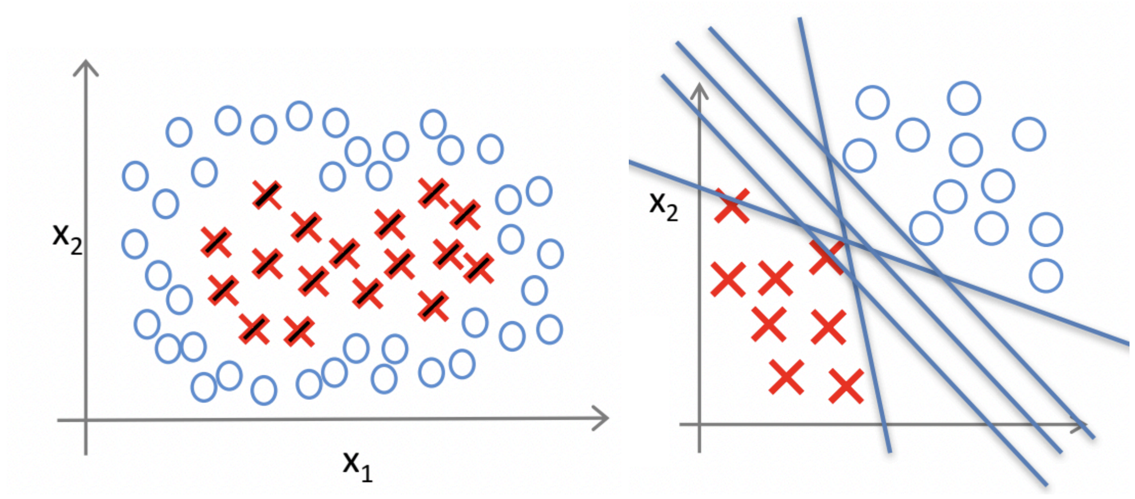


Figure 1: Visualization of SVR (Ive, 2023)

Using SVR requires carefully selecting a kernel function that will transform data into a higher-dimensional space and hyperparameters (HPs), these parameters can be fine-tuned to achieve the best results, the kernel function will also allow the algorithm to learn non-linear relations. SVR is an algorithm that is well-suited for complex problems with small datasets and a large number of features, as well as handling non-linear relationships, however, it is a time-consuming and computationally expensive algorithm, it is also sensitive to outliers in the data, which can affect the shape of the hyperplane and ultimately the results.

3.3 Random Forest Regression

RFR (Breiman, 2001) is a machine learning algorithm based on decision trees, a decision tree can be thought of as a bunch of if-else conditions, starting at the top with one node, the node then splits into a left and right node, these nodes then split into their own respective right and left nodes, splitting as many time as told until they become terminal nodes.

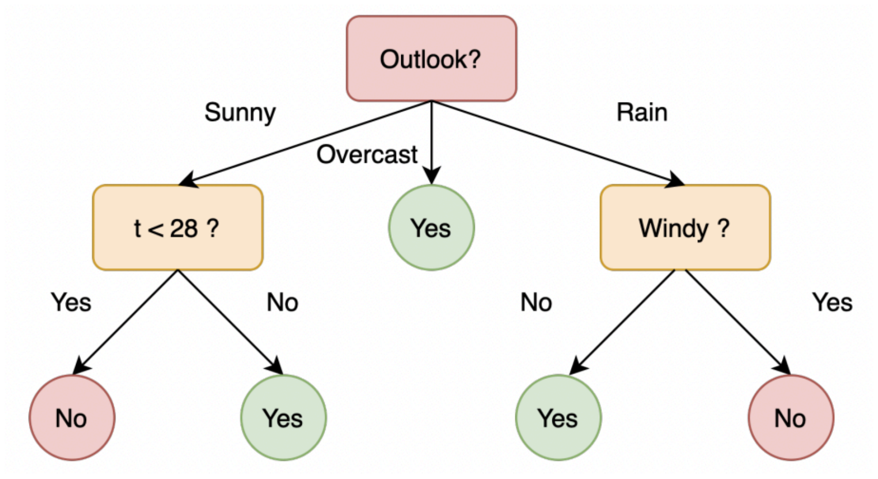


Figure 2: Simple visualization of a decision tree (Ive, 2023)

Random Forest, as the name suggests is an ensemble of decision trees constructed in a random way, each tree is created using a random subset of input features and a subset of the training data, each tree then makes its own individual prediction based on a set of rules learned from the input features (Russell and Norvig, 2010). Lastly, the predictions of all the individual trees are averaged to produce a single result.

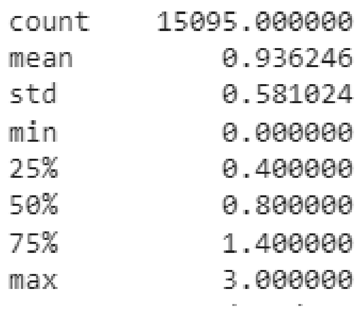
The random selection of features and training data, as well as the averaging, helps reduce overfitting and improves the accuracy of the model. It is a robust algorithm that is great for capturing non-linear relationships between the input and output variables, it works well with a large dataset and can handle a large number of input features. However, the algorithm can take a significant amount of time to train depending on the size of the dataset and the number of trees being used, it is also difficult to tune to get the optimal parameters as it can vary depending on the nature of the data. Furthermore, unlike linear regression, it cannot extrapolate values outside of the training data.

4. Experimental Training

4.1 Dataset

The data for this research project is the Humicroedit (Hossain et al., 2019) dataset created for research in computational humour. The dataset consists of approximately 5,000 original headlines sourced from popular subreddits r/worldnews and r/politics on Reddit (reddit.com) between January 2017 to May 2018, each original headline has three modified versions, resulting in a total of 15,059 edited headlines. The dataset was then annotated by expert editors and judges of humour from Amazon Mechanical Turk, the editors were tasked with making funny micro-edits to headlines, and the judges rated the funniness of the edited headlines on a scale of 0-3.

By doing some simple data analysis with Pandas (McKinney, 2010) we can obtain some basic information about the dataset, using the .describe() function in Pandas on the Mean Grade column we obtain the following statistics.



```
count    15095.000000
mean         0.936246
std          0.581024
min          0.000000
25%          0.400000
50%          0.800000
75%          1.400000
max          3.000000
```

Figure 3: Simple statistics on the Mean Grade column

The ‘mean’ of the Mean Grade column only averages out at **0.936** which is considerably low considering the domain ranges from 0 to 3. Further investigation into this shows that only 5 edited headlines achieved a maximum score of 3.0, while a total of 799 edited headlines achieved a minimum score of 0. This is also supported by the fact that the standard deviation of this column is only **0.581**, even the 75th percentile contains a grade that is below half of the range at **1.4**.

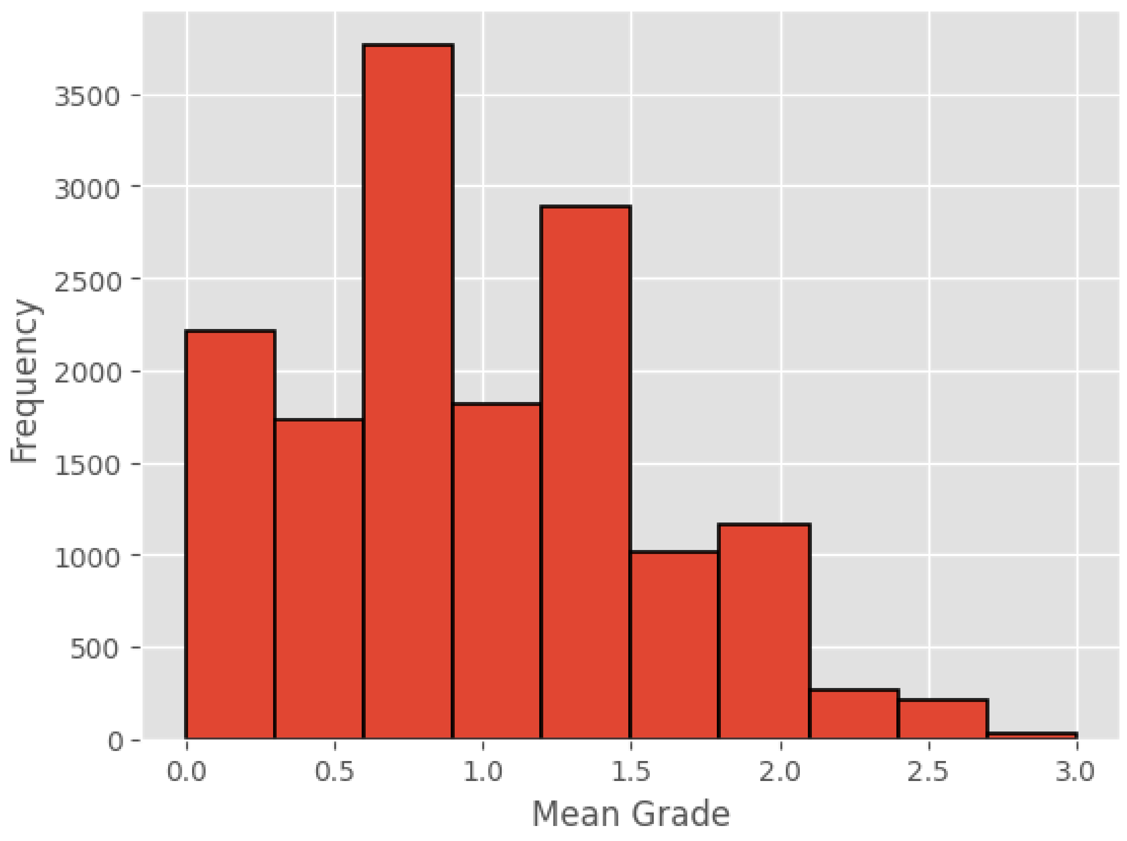
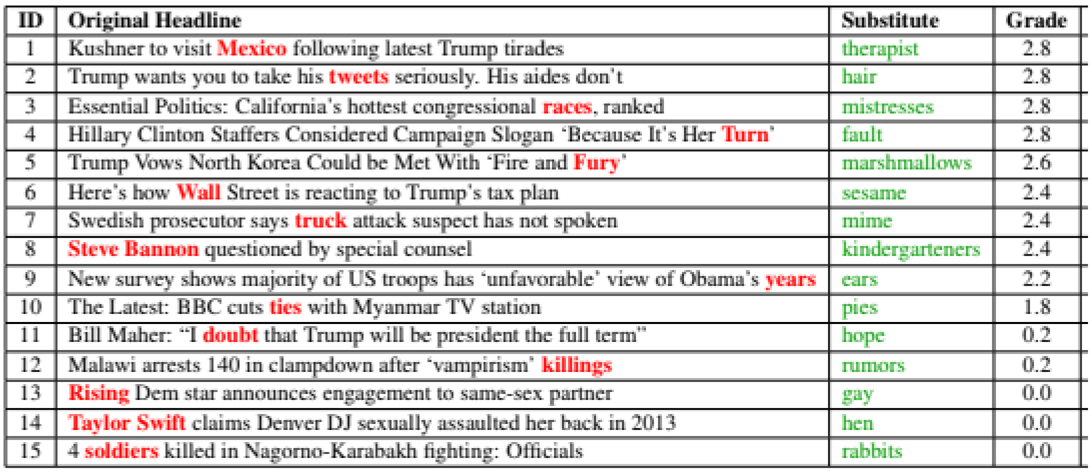


Figure 4: A histogram showing the distribution of Mean Grade

From this histogram we can see that the majority of the mean grade falls between 0 and 1.5, with a spike between 0.5 and 1.0. Data like this means that when the chosen model is trained, it could lead to the algorithm being harsher when predicting the scores of edited headlines.



ID	Original Headline	Substitute	Grade
1	Kushner to visit **Mexico** following latest Trump tirades	therapist	2.8
2	Trump wants you to take his **tweets** seriously. His aides don't	hair	2.8
3	Essential Politics: California's hottest congressional **races**, ranked	mistresses	2.8
4	Hillary Clinton Staffers Considered Campaign Slogan 'Because It's Her **Turn**'	fault	2.8
5	Trump Vows North Korea Could be Met With 'Fire and **Fury**'	marshmallows	2.6
6	Here's how **Wall** Street is reacting to Trump's tax plan	sesame	2.4
7	Swedish prosecutor says **truck** attack suspect has not spoken	mime	2.4
8	**Steve Bannon** questioned by special counsel	kindergarteners	2.4
9	New survey shows majority of US troops has 'unfavorable' view of Obama's **years**	ears	2.2
10	The Latest: BBC cuts **ties** with Myanmar TV station	pies	1.8
11	Bill Maher: "I **doubt** that Trump will be president the full term"	hope	0.2
12	Malawi arrests 140 in clampdown after 'vampirism' **killings**	rumors	0.2
13	**Rising** Dem star announces engagement to same-sex partner	gay	0.0
14	**Taylor Swift** claims Denver DJ sexually assaulted her back in 2013	hen	0.0
15	4 **soldiers** killed in Nagorno-Karabakh fighting: Officials	rabbits	0.0

Table 4: Example headlines, their substitute word, and the mean funniness grades (Hossain et al., 2019)

The above table contains some example headlines in the dataset, it contains the original headline, the word highlighted in red being the word that will be replaced, the substitute word in green, and the mean funniness grade across 5 judges giving their score.

4.2 Pre-Processing

Preprocessing data is an essential step to ensure that machine learning models can learn from the data effectively and make accurate predictions, the quality and suitability of input data can have a significant impact on the performance and accuracy of the models. Preprocessing techniques can differ depending on the type of data and the algorithm being used, below are some of the techniques being employed for this task.

4.2.1 Punctuation, Whitespace, Number, Stop-word Removal

Text data can often be noisy and unstructured, often containing irrelevant information that can slow down the training time and affect the performance of machine learning algorithms. By removing punctuations and whitespace, it will ensure that the focus remains on the actual words and letters, along with preventing inconsistencies in the text data. For numbers, while I am aware that numbers may be a reason for humour in some news headlines, since I will tokenize them into individual characters later on, I believe that individual numbers will not serve a meaningful purpose to the study. Stop-word removal is seen in a similar manner, I believe that removing stop-words from the data will improve the model’s performance overall.

4.2.2 Tokenization

“Tokenization is the process of breaking a stream of text into words, phrases, symbols, or other meaningful elements called tokens.” (Kannan et al., 2014). It typically assumes space as a delimiter unless specified otherwise, meaning in a sentence each token will be separated by a space. In our case, not only do we want to break the headline down into separate words, we also want to further break down the words into individual characters.

4.2.3 Word lemmatization

Lemmatization is a technique that is similar to stemming, however, it also considers the context of the sentence and converts the word to its meaningful base form, resulting in more accurate lemmas. This technique was chosen over stemming as sentence context is a big part of humour, stemming might accidentally remove part of words/letters which could skew the data.

4.3 Features

4.3.1 TF-IDF

The tf-idf weighting is a statistical measure used to evaluate the importance of a word in a document in a collection of documents, it is the product of two terms, term frequency and inverse document frequency (Jurafsky and Martin, 2022). Term frequency can be found by the following equation:

$$tf_{t,d} = log_{10}(count(t, d) + 1)$$

Where:

- **count** is the raw frequency of the word **t** in document **d**

For example, a word that occurs 0 times in a document would have:

$$tf_{t,d} = log_{10}(1) = 0$$

A word that occurs 5 times in a document:

$$tf_{t,d} = log_{10}(6) = 0.78$$

The second part in tf-idf, inverse document frequency is used to measure the rarity of a term across all documents, it gives a higher weighting to words that only occur in a few documents while giving a lower weighting to words that occur across the entire collection. Document frequency **df$_t$** refers to the number of documents term **t** occurs in, it is defined as

$$idf_t = log_{10}\left(\frac{N}{df_t}\right)$$

Where:

- **N** is the total number of documents in the collection
- **df$_t$** is the number of document term **t** occurs in

For example, a word that occurs in 2 documents out of 10:

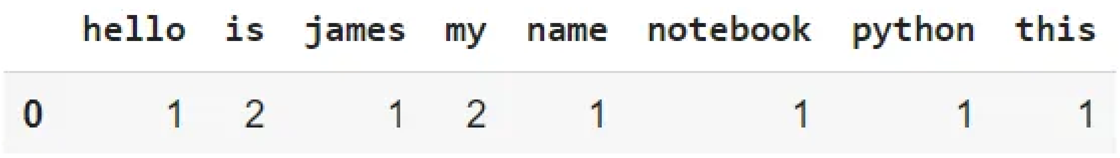
$$idf_t = log_{10}\left(\frac{10}{2}\right) = 0.7$$

Finally, the tf-idf weighted score **$w_{t,d}$** for word **t** in document **d** is calculated by combining tf and idf:

$$w_{t,d} = tf_{t,d} \times idf_t$$

4.3.2 CountVectorizer

CountVectorizer (Pedregosa et al., 2011) from scikit-learn is a method used to transform a text into numerical data that machine learning models can process, it converts a collection of text documents to a matrix of token counts. First, the text is split into individual words or tokens, a process known as tokenization. Then, a list of all the unique words in the corpus is created, and finally, the frequency of each token in each document is counted and represented in a matrix. For example, if we had a string of text "Hello my name is james, this is my python notebook", and we put that string through CountVectorizer, the text would be transformed into the below.



	hello	is	james	my	name	notebook	python	this
0	1	2	1	2	1	1	1	1

Table 5: Sparse matrix of tokens

In our case we are also interested in text down to the character level, instead of CountVectorizer operating on the entire headline and tokenizing each word, it will only operate on the edited word, and tokenize that word into individual characters.

4.3.3 TfidfTransformer

While CountVectorizer provides a count of the frequency of each word in the corpus, it doesn't take into account the importance of each word in the documents, stop words such as "a", "the", "are", should not have the same relevance as other words. How tf-idf works have already been explained previously, here TfidfTransformer (Pedregosa et al., 2011) from scikit-learn is used to compute the tf-idf score for the tokens created with CountVectorizer, it "Transforms a count matrix to a normalized tf or tf-idf representation." (Pedregosa et al., 2011)

4.5 Training procedure (Hyper-parameter tuning)

Aside from just the base algorithm, all of these algorithms can take a number of different parameters to fine-tune and optimise before they are run. These parameters are known as hyper-parameters (HPs) and the process is known as HP tuning. HP tuning is a crucial step in building an effective model as selecting the right HPs can have a significant impact on the accuracy and performance of the model. In this section, I will be explaining some of the HPs available in these models and what strategies I have applied to optimise them to the best of my abilities.

4.5.1 SVR Hyper-parameters

The SVR model in scikit-learn contains numerous HPs available for the user to fine-tune, allowing the user to mould the algorithm to work best for the dataset and purposes they have. Selecting the optimal HPs will allow the model to find the optimal decision boundary that minimises the prediction error while maintaining a certain level of smoothness.

Kernel: The kernel function (Smola and Schölkopf, 2004) is responsible for transforming the input data into a higher-dimensional space, enabling the algorithm to capture complex, nonlinear relationships between the features and the target variable. Common kernel choices include linear, polynomial, radial basis function (RBF), and sigmoid. The choice of kernel greatly affects the models' performance, and it is crucial to select the one that best suits the dataset.

Regularisation parameter (C): The C parameter controls the trade-off between the models' complexity and the smoothness of the decision boundary. A larger value of C encourages the model to fit the training data more closely, potentially capturing intricate patterns but increasing the risk of overfitting. A smaller value of C promotes a smoother decision boundary, potentially reducing overfitting but possibly missing some patterns in the data.

Kernel coefficient (gamma): The gamma parameter is specific to the RBF, polynomial, and sigmoid kernels. It determines the shape and flexibility of the decision boundary, with larger values resulting in more flexible boundaries and smaller values producing smoother boundaries. As with the C parameter, it is important to balance between capturing the data complexity and avoid overfitting.

Epsilon: The epsilon parameter controls the width of the margin or "tube" around the SVR decision boundary, within which errors are tolerated. A larger epsilon value leads to a wider margin and a smoother decision function, while a smaller epsilon results in a narrower margin and a more sensitive model to the training data.

4.5.2 RFR Hyper-parameters

The RFR model in scikit-learn contains a whole host of HPs available for us to make sure our model best fits our dataset and achieves the best possible performance. Here I will discuss some of the key HPs that were used for the model.

Criterion: The criterion parameter refers to the function that measures the quality of a split. For regression tasks, the most common options are Squared Error, which minimises the sum of squared differences between the actual and predicted values, and Absolute Error, which minimises the sum of absolute differences. The choice of criterion can affect the models' performance and sensitivity to outliers.

Number of trees (n_estimators): This parameter determines the number of decision trees in the RFR. Increasing the number of trees typically leads to better performance, as the model can learn more complex relationships and achieve higher stability. However, a larger number of trees also increases the computational cost and can lead to diminishing returns. A common practice is to start with a moderately high number of trees (e.g., 100) and adjust it based on the performance during HP tuning.

Maximum depth of trees (max_depth): The maximum depth of each decision tree influences the models' capacity to learn complex relationships between features and the target variable. A deeper tree can capture more intricate patterns but is more prone to overfitting.

Minimum samples per leaf (min_samples_leaf): This parameter controls the minimum number of samples required to create a leaf node in each decision tree. A higher value increases the models' robustness and reduces the risk of overfitting, while a lower value allows the model to capture finer details in the data.

4.5.3 HP Tuning Method (Grid Search)

There are various different methods to find a good and reliable set of parameters for an algorithm on a given task, such as random search and Bayesian optimization, however, this investigation will focus on grid search. Despite its computational inefficiency, here are some reasons why I chose grid search over the other methods:

Simplicity and Interpretability: Grid search is easy to implement and understand, making it a suitable choice for a first attempt at HP tuning. It provides a clear picture of the search space, allowing for a straightforward interpretation of the results.

Exhaustive Exploration: Grid search guarantees that the best combination of HPs within the predefined search space will be found, as it evaluates all possible combinations. This exhaustive exploration can provide confidence in the obtained results and help identify potential areas for improvement.

Grid search is a method that exhaustively tests all possible combinations of HP values within a predefined search space. Although computationally expensive, it ensures that the optimal combination of HPs is found. Grid search systematically explores all possible combinations of HP values within a predefined search space. It trains and evaluates a model using each combination and selects the one that results in the best performance based on a predefined evaluation metric, such as accuracy or mean squared error.

To perform the grid search I used the GridSearchCV (Pedregosa et al., 2011) class from Scikit-learn. The GridSearchCV function takes multiple different inputs to achieve its goal, below I will list and explain the parameters I have chosen to implement the grid search.

estimator: The estimator parameter refers to the machine learning model I'm tuning. This can be any model that implements the Scikit-learn estimator interface, such as classifiers or regressors from the sklearn library. In this case, it will be RFR and SVR.

param_grid: The param_grid parameter is a dictionary that defines the search space for each HP of the given estimator. Each key in the dictionary corresponds to an HP name, and the corresponding value is a list of possible values for that HP. The grid search will evaluate all possible combinations of these HP values. The keys in the dictionary refer to the HPs mentioned in sections 4.5.1 and 4.5.2.

cv: The cv parameter determines the number of cross-validation folds to use during the grid search process. Cross-validation is a technique for estimating the performance of a model on unseen data by dividing the dataset into a certain number of equally sized folds, training the model on all but one of the folds, and finally evaluating its performance on the remaining fold.

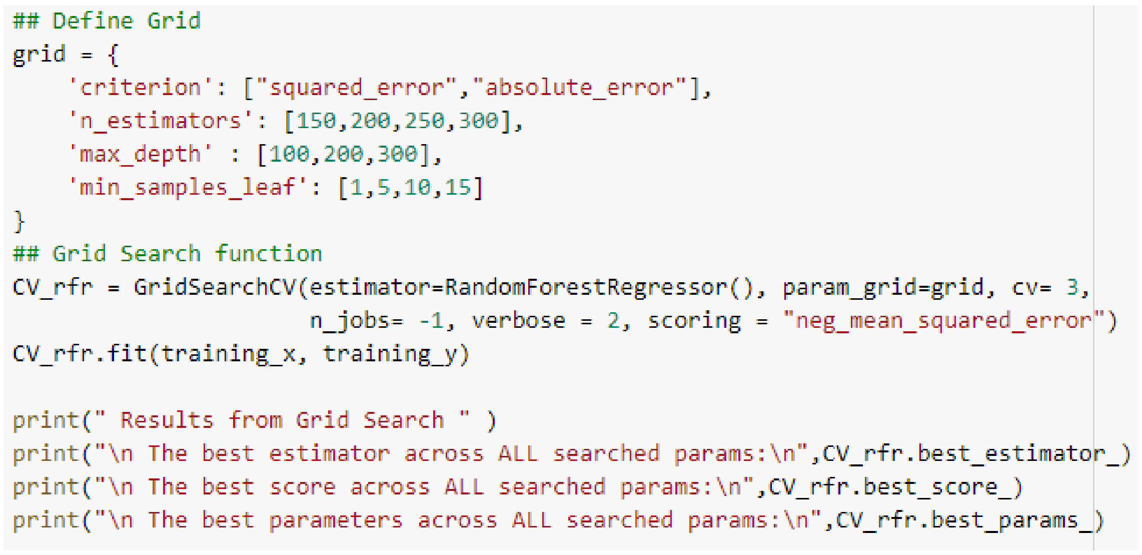
n_jobs: The n_jobs parameter controls the number of CPU cores to use for parallel computation during the grid search process. By setting this parameter to a value greater than 1, or to -1 to use all available cores, I can speed up the grid search by processing multiple hyperparameter combinations simultaneously.

verbose: The verbose parameter controls the verbosity of the output messages during the grid search process. Higher values will produce more detailed messages, while lower values will suppress most messages. This can be helpful for monitoring the progress of the grid search and identifying potential issues.

scoring: The scoring parameter specifies the evaluation metric to optimise during the grid search process. This can be a string representing a built-in Scikit-learn scoring function (e.g., 'accuracy', 'f1' or 'mean_squared_error'), or a custom scoring function. The grid search will attempt to find the hyperparameter combination that maximises or minimises the chosen scoring metric. In this case, I have chosen ‘neg_mean_squared_error’ which is the mean squared error metric explained later in section 5.1.1.

Chosen parameters:

- estimator = RFR() or SVR()
- cv = 3
- n_jobs = -1
- verbose = 2
- scoring = ‘neg_mean_squared_error’
- param_grid = Dictionary set depending on which model



```
## Define Grid
grid = {
    'criterion': ["squared_error","absolute_error"],
    'n_estimators': [150,200,250,300],
    'max_depth' : [100,200,300],
    'min_samples_leaf': [1,5,10,15]
}
## Grid Search function
CV_rfr = GridSearchCV(estimator=RandomForestRegressor(), param_grid=grid, cv= 3,
                      n_jobs= -1, verbose = 2, scoring = "neg_mean_squared_error")
CV_rfr.fit(training_x, training_y)

print(" Results from Grid Search " )
print("\n The best estimator across ALL searched params:\n",CV_rfr.best_estimator_)
print("\n The best score across ALL searched params:\n",CV_rfr.best_score_)
print("\n The best parameters across ALL searched params:\n",CV_rfr.best_params_)
```

Figure 5: Code snippet of the grid search function

Due to the computationally expensive nature of the grid search, limited testing of HPs was conducted as some searches were taking hours on end. These were the best-performing HPs I ended up with.

SVR

- kernel = rbf
- C = 10
- epsilon = 0.2
- gamma = 0.001

RFR

- criterion = squared_error
- max_depth = 300
- min_samples_leaf = 5
- n_estimators = 250

5. Experimental Results

5.1 Evaluation Metrics

Evaluation metrics are used in machine learning to measure and compare the effectiveness and performance of models. Different evaluation metrics are used depending on the type of problem, as this is a regression task, we will be using mean squared error and root mean squared error. These metrics measure how close the predicted values are to actual values while penalising errors made by the algorithm.

5.1.1 Mean Squared Error (MSE)

MSE is a performance metric used to evaluate the performance of regression models, it is the average of the sum of the squared difference between the predicted and actual values, the lower the MSE, the better a model fits a dataset. The difference between the values is squared to put more significance on errors.
The formula for MSE:

$$MSE = \frac{\Sigma(\hat{y}_i - y_i)^2}{n}$$

Where:

- **ŷi** is the predicted value for the ith observation
- **yi** is the observed value for the ith observation
- **n** is the sample size

For example if we have an actual output array of [1,3,5,6] and a predicted output of [2,3,7,8], then our MSE would be as follows:

$$MSE = \frac{(1-2)^2+(3-3)^2+(5-7)^2+(6-8)^2}{4} = \frac{(1+0+4+4)}{4} = 2.25$$

5.1.2 Root Mean Squared Error (RMSE)

RMSE is just the square root of MSE, in this case:

$$RMSE = \sqrt{2.25} = 1.5$$

While MSE is useful in comparing different regression models, RMSE is more intuitive and interpretable as it is in the same units as the dependent variable, it gives an estimate of the average difference of the predictions from the actual output values.

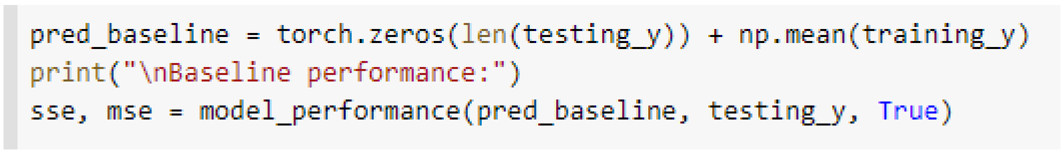
5.1.3 Comparison

Both metrics will be used in the ablation study in section 5.4 to measure the model's performance. While both metrics are used to quantify the difference between predicted and actual values, there are some key differences.

Since RMSE is in the same unit as the original values, it makes it easier to interpret and compare with the original values, while MSE can be more difficult to relate to the original data. Both MSE and RMSE are sensitive to outliers, but MSE, due to squaring, is more sensitive to large errors. This means that a model with a few large errors will have a higher MSE than one with several small errors, even if their RMSE values are similar.

5.2 Baseline model

A baseline model refers to a simple model that provides a reference point for evaluating the performance of more complex models, it is typically the simplest possible model that can be applied to the task. The main purpose of this is to establish a minimum level of performance that any model/algorithm should exceed for it to be useful to this task, if a more complex model cannot outperform the baseline, then it is not considered a viable solution or there may be a problem with the data or training process.



```
pred_baseline = torch.zeros(len(testing_y)) + np.mean(training_y)
print("\nBaseline performance:")
sse, mse = model_performance(pred_baseline, testing_y, True)
```

Figure 6: Code for calculating the baseline from mean

For our case, we simply take the average of the Mean Grade column from the training set, put it against the values of the Mean Grade column from the test set, input this into a function that calculates the evaluation metrics and we end up with the following:

MSE	RMSE
0.3303	0.5747

Table 6: Error scores from the baseline average

5.3 Ablation Study and Qualitative Assessment

In this section, I will discuss the results technical detail and provide a deeper analysis of the models' performance. This may help explain the underlying reasons for the models' success or failure, as well as identify specific components that contribute significantly to its performance.

5.4.1 Base Models

Firstly, these are the results that we got from just processing the training and test data through CountVectorizer and TfidfTransformer, and then putting it against the Mean Grade, no preprocessing, character tokenizing or HP fine-tuning has been done.

Model	MSE	RMSE
LR	509.0080	22.5612
SVR	0.3131	0.5596
RFR	0.3376	0.5811
Baseline	0.3303	0.5747

Table 7: Results from base training and test set

First thing that we notice is the massive error for linear regression (LR), this could be due to the presence of non-linear relationships or interactions between features that the LR model is unable to capture. SVR performs the best with the lowest MSE of 0.3131, this suggests that the SVR model is better at capturing the underlying patterns in our dataset and making accurate predictions compared to the other models. RFR comes closely in second place with a small difference in error score, which makes sense as random forest models are known for their ability to handle non-linear relationships.

5.4.2 With Preprocessing and Character Tokenization

Next, the training and test data were preprocessed by techniques explained in section 4.2, the character tokenization was also performed and added to the stack. Once again, these were all fed into the models with no HPs set.

	MSE			**RMSE**		
set	**LR**	**SVR**	**RFR**	**LR**	**SVR**	**RFR**
all	2.3916	0.3281	0.3160	1.5456	0.5728	0.5621
-test_counts	<u>0.4550</u>	0.3294	<u>0.3157</u>	<u>0.6745</u>	**0.5739**	<u>0.5619</u>
-test_counts_edited	0.8837	**0.3331**	**0.3307**	0.9401	0.5572	**0.5751**
-test_char_counts	**3.9999**	<u>0.3087</u>	0.3228	**2.0000**	<u>0.5556</u>	0.5682
-test_char_counts_edited	3.5021	0.3271	0.3242	1.8714	0.5720	0.5694
Baseline	0.3303			0.5747		

Table 8: Feature ablation experiment with character tokenization and preprocessing, the most contributing set in **bold**, best performing set <u>underlined</u>

In order to understand which features work the best, if we remove one feature and the results drop a lot, it means that feature was really important, said feature is highlighted in bold. “test_counts_edited” is on average the most contributing set in this experiment, causing the biggest increase in error score for 3 different results. However, the biggest contributing feature set overall is “test_char_counts” with the LR model, causing an increase of 1.6083 in MSE.

Interestingly, the SVR models' performance for this feature set improved slightly, while the RFR models' performance increased marginally.

With the exception of LR and some of the results in bold, most of the results from SVR and RFR outperformed the baseline, with the best-performing RMSE score improving by 0.0191, and the best MSE score improving by 0.0216. There is a big difference in results across the board for the LR model, likely due to the models' inability to capture non-linear patterns. Removing "test_counts" had a significant impact on the LR model, improving its performance by an MSE of 1.9366.

In conclusion, the most significant impact on the LR model's performance occurred when removing "test_counts" and "test_counts_edited". For the SVR and RFR models, the performance remained relatively stable across different feature sets, with only minor fluctuations in the MSE values. This suggests that the SVR and RFR models are more robust to changes in the input features, while the LR model is more sensitive to feature selection.

5.4.3 With Hyperparameters

In this experiment, character tokenization, preprocessing, and optimal HPs from fine-tuning in section 4.5 are included.

	MSE (HPs)		**MSE (No HPs)**	
set	**SVR**	**RFR**	**SVR**	**RFR**
all	0.3170	0.3187	0.3281	0.3160
-test_counts	0.3207	<u>0.3183</u>	0.3294	<u>0.3157</u>
-test_counts_edited	**0.3301**	**0.3240**	**0.3331**	**0.3307**
-test_char_counts	<u>0.3104</u>	<u>0.3183</u>	<u>0.3087</u>	0.3228
-test_char_counts_ edited	0.3156	0.3224	0.3271	0.3242
Baseline	0.3303			

Table 9: MSE Feature ablation experiment with HPs compared with Table 8

	RMSE (HPs)		**RMSE (No HPs)**	
set	**SVR**	**RFR**	**SVR**	**RFR**
all	0.5630	0.5645	0.5728	0.5621
-test_counts	0.5663	<u>0.5642</u>	**0.5739**	<u>0.5619</u>
-test_counts_edited	**0.5745**	**0.5692**	0.5572	**0.5751**
-test_char_counts	<u>0.5571</u>	<u>0.5642</u>	<u>0.5556</u>	0.5682
-test_char_counts_ edited	0.5618	0.5678	0.5720	0.5694
Baseline	0.5747			

Table 10: RMSE Feature ablation experiment with HPs compared with Table 8

This time all the results in Table 9 and 10 outperform the baseline, with the best performing MSE beating the baseline by 0.0199 and the best RMSE beating the baseline by 0.0176, I have chosen to omit the LR model here as no fine-tuning was done and no HPs were implemented. Although the results vary minimally, the best performing feature set is when “test_char_counts” was excluded. Surprisingly in Table 9, while the SVR model with HPs outperformed the majority of the results from the SVR model without HPs, the best performing features set with HPs performed slightly worse than its non-HP counterpart.

It is also very clear in both tables with the results in bold that “test_counts_edited” is the most contributing set once again with an overwhelming majority. While the improvement is not massive, it is still good to see that fine-tuning the HPs has led to more optimal results.

5.4.4 Qualitative analysis

Through looking at the Tfidf scores of words after preprocessing, words that do not appear a lot such as “cane”, “trucks” and “launches” have a maximum idf weight of 9.482, while the lowest weighting in the list goes to “trump” at 1.974, meaning that this word occurs in a lot of headlines.

Taking the best performing model along with the best set of features: SVR with no HPs and “-test_char_counts”, we can go into the model and find specifically which headlines are performing the best and worst. I can identify that one of the worst headlines is “US Ambassador to Netherlands discovers own words as ‘fake news’”, while one of the best performing ones is “Michigan mannequin held captive, sexually assaulted for 3 days in a house of horrors”. However, these results may not be 100% accurate; the indices in the code may not match directly with the ordering of headlines in the dataset. Between RFR and SVR, it is not clear to say which one is better for this task, regardless of the presence of HPs or not, it is highly dependent on the quality and pattern of data.

5.4.5 Further Evaluation

In this section, I will evaluate the limitations of these findings.

Subjective nature of humour annotation - One of the inherent challenges in studying humour is the subjectivity involved in its annotation. People have diverse senses of humour, which makes it difficult to establish a universally agreed-upon definition of what is funny. Hence, the humour annotations in our dataset may not accurately represent the wide range of humour preferences within the general population.

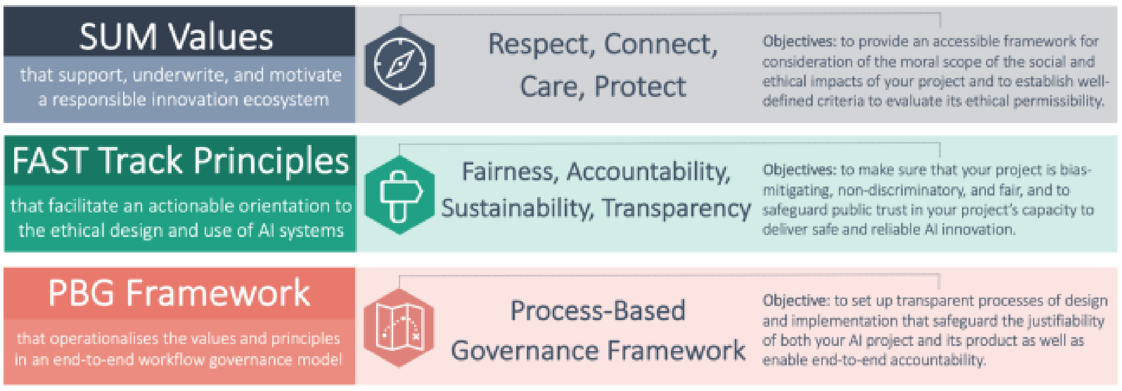
This subjectivity may lead to biases in the data, as the humour annotations could be influenced by the personal preferences and cultural backgrounds of the annotators. As a result, our analysis may not adequately capture the complexities of humour, limiting our findings to broader contexts.

Data limitation - The analysis is based on a single dataset, which presents certain limitations. Relying on one dataset restricts the diversity and scope of the data, potentially leading to biassed or incomplete conclusions. The models were trained on a single dataset annotated by 5 judges, essentially meaning the predictions that the model makes are limited to what those 5 judges perceive as humorous.

Relatively simple algorithms - The algorithms employed in our analysis are relatively simple, while straightforward to implement and interpretable, it may limit their ability to accurately model the complexities of humour. More advanced machine learning techniques, such as deep learning models or those that incorporate additional features and contextual information, could potentially offer a deeper understanding of humour. Exploring these advanced approaches may help to address the limitations of our current results and provide further insights into humour detection and prediction.

6. Ethics

As we delve into the realm of NLP, humour detection, and machine learning, it is crucial to address the ethical implications of developing and deploying artificial intelligence (AI) systems in these domains. While many of the ethical concerns may arise depending on how the results of this research will be used in further tasks, it is important to address them all nonetheless. This section will highlight key ethical concerns, principles, and recommendations for designing and implementing humour detection systems.


SUM Values
that support, underwrite, and motivate a responsible innovation ecosystem
Respect, Connect, Care, Protect
FAST Track Principles
that facilitate an actionable orientation to the ethical design and use of AI systems
Fairness, Accountability, Sustainability, Transparency
PBG Framework
that operationalises the values and principles in an end-to-end workflow governance model
Process-Based Governance Framework

Objectives: to provide an accessible framework for consideration of the moral scope of the social and ethical impacts of your project and to establish well-defined criteria to evaluate its ethical permissibility.

Objectives: to make sure that your project is bias-mitigating, non-discriminatory, and fair, and to safeguard public trust in your project's capacity to deliver safe and reliable AI innovation.

Objective: to set up transparent processes of design and implementation that safeguard the justifiability of both your AI project and its product as well as enable end-to-end accountability.

Figure 7: Ethical platform for the responsible delivery of an AI project (Leslie, 2019)

6.1 Bias, Discrimination, and Fairness

Because AI systems derive their insights from the existing structures and dynamics of the societies they analyse, they can “reproduce, reinforce, and amplify the patterns of marginalisation, inequality, and discrimination that exist in these societies” (Leslie, 2019), especially when trained on large datasets containing biassed language data. In humour detection, this can result in prejudiced or offensive content being flagged as humorous or vice versa.

The dataset used to train and test AI systems is also insufficiently representative of the demographic they were drawn from. This creates possibilities of biassed and discriminatory outcomes, as the data the system is trained with is biassed from the start. For example, as mentioned in section 1.1, investigations on this subject can be used as a pipeline for hate speech detection

Ideally, for a dataset written entirely by men compared to a dataset written entirely by women, the score should be the same, however, this will most likely not be the case.

6.2 Lack of Transparent, Explainable, or Justifiable Outcomes

AI systems, particularly those using deep learning, can be complex and opaque, making it challenging to understand and justify their decision-making processes. Many systems produce results by “operating on high dimensional correlations that are beyond the interpretive capabilities of human scale reasoning” (Leslie, 2019). Predictions/decisions that are difficult to interpret or rationalise can be especially problematic in situations such as medical diagnosis or criminal justice. Unjustifiable outcomes occur when AI systems produce decisions or predictions that are biassed, discriminatory, or otherwise unfair. This can be due to bias in the training data or flawed algorithmic design.

In our case, while the models used are simple and easy to understand to a certain degree, it may still be hard to justify the decision-making process and understand the reasoning behind the outcomes. Even if the algorithm training and testing itself do not have any issues, ethical concerns will arise depending on how the outcome of this research is used in the future.

6.3 Accountability

Accountability is a fundamental ethical consideration in the development and deployment of AI systems. Identifying responsible parties when AI systems make mistakes or cause harm (intentional or not) is essential for addressing negative consequences and promoting trust in this technology. Establishing accountability will ensure that AI systems are developed and implemented in a manner that is ethically sound. It can help foster trust between developers, users, and stakeholders, it can help identify and address potential risks, preventing harm and promoting responsible innovation.

For the context of this project, stakeholders can range from the people who created the dataset, and developers who created the machine learning models, to me or anyone who uses the results of this investigation in any capacity.

6.4 Future Improvements

In light of the limitations identified and the ethical concerns discussed above, several future improvements can be considered to enhance the performance, interpretability, and ethical robustness of my system.

Incorporating accountability mechanisms - Improving accountability can be achieved by integrating methods for monitoring, auditing, and evaluating the system's decisions. Engaging a diverse group of stakeholders and domain experts in the development process can help identify and address potential biases and ethical concerns.

Mitigating biases and promoting fairness - Investigating and addressing potential biases in the dataset and models is required to ensure a fair representation of different groups. Techniques such as re-sampling, re-weighting, or adversarial training can be employed to promote fairness in the AI system.

Exploring multiple datasets - To enhance the robustness and generalizability of my system, future work should involve analysing multiple datasets that encompass diverse sources of humour and different cultural contexts.

7. Conclusion

In conclusion, this project investigated the contribution of different sets of features and preprocessing techniques to the funniness of edited news headlines, one of which being character level features. By utilising machine learning models, such as linear regression, support vector regression and random forest regression, the results of this study have the potential to serve as a foundation for future research in character level humour detection and prediction. Furthermore, this project has explored the potential social and ethical implications of developing and deploying AI systems.

Our study shows that preprocessing techniques and HP fine-tuning are both good ways to substantially increase the performance of the models, we can also see that there exists a relationship between character level features and the humour level of headlines, but this requires further investigation to confirm. We have also identified a number of limitations with regards to our results and experimental process.

8. Acknowledgements

I would like to sincerely thank my supervisor Dr. Julia Ive for her invaluable support throughout this project.

References

- Barbieri, F., Saggion, H. and Ronzano, F. (2014). Modelling Sarcasm in Twitter, a Novel Approach. *Meeting of the Association for Computational Linguistics*. doi:https://doi.org/10.3115/v1/w14-2609.
- Beheshti, N. (2022). *Random Forest Regression*. [online] Medium. Available at: https://towardsdatascience.com/random-forest-regression-5f605132d19d [Accessed 10 Mar. 2023].
- Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), pp.5–32. doi:https://doi.org/10.1023/a:1010933404324.
- Chandrasekar, P. and Qian, K. (2016). The Impact of Data Preprocessing on the Performance of a Naive Bayes Classifier. *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. doi:https://doi.org/10.1109/compsac.2016.205.
- Chy, A.N., Seddiqui, M.H. and Das, S. (2014). *Bangla news classification using naive Bayes classifier*. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICCITechn.2014.6997369.
- Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), pp.273–297. doi:https://doi.org/10.1007/bf00994018.
- Davidov, D., Tsur, O. and Rappoport, A. (2010). Semi-supervised recognition of sarcastic sentences in Twitter and Amazon. *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*. [online] Available at: https://arxiv.org/pdf/1810.04805.pdf.
- Gaikwad, M., Ahirrao, S., Phansalkar, S. and Kotecha, K. (2021). Online Extremism Detection: A Systematic Literature Review With Emphasis on Datasets, Classification Techniques, Validation Methods, and Tools. *IEEE Access*, 9, pp.48364–48404. doi:https://doi.org/10.1109/access.2021.3068313.
- Granik, M. and Mesyura, V. (2017). Fake news detection using naive Bayes classifier. *2017 IEEE First Ukraine Conference on Electrical and*

Computer Engineering (UKRCON). [online] doi:https://doi.org/10.1109/ukrcon.2017.8100379.

- Hossain, N., Krumm, J. and Gamon, M. (2019). ‘President Vows to Cut Hair’: Dataset and Analysis of Creative Text Editing for Humorous Headlines. *Proceedings of the 2019 Conference of the North*, 1. doi:https://doi.org/10.18653/v1/n19-1012.
- Hossain, N., Krumm, J., Gamon, M. and Kautz, H. (2020). SemEval-2020 Task 7: Assessing Humor in Edited News Headlines. *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. doi:https://doi.org/10.18653/v1/2020.semeval-1.98.
- Istaiteh, O., Al-Omoush, R. and Tedmori, S. (2020). *Racist and Sexist Hate Speech Detection: Literature Review*. [online] IEEE Xplore. doi:https://doi.org/10.1109/IDSTA50958.2020.9264052.
- Jin, S., Yin, Y., Tang, X. and Pedersen, T. (2020). *Duluth at SemEval-2020 Task 7: Using Surprise as a Key to Unlock Humorous Headlines*. [online] pp.986–994. Available at: https://aclanthology.org/2020.semeval-1.128.pdf [Accessed 1 May 2023].
- Jurafsky, D. and Martin, J. (2022). *Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition Third Edition draft*. [online] Available at: https://web.stanford.edu/~jurafsky/slp3/ed3book_jan122022.pdf.
- Kannan, S., Gurusamy, V., Vijayarani, S., Ilamathi, J., Nithya, M., Kannan, S. and Gurusamy, V., 2014. Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks*, *5*(1), pp.7-16.
- K, S., S, T.I., Malik, A. and R, P.J.I. (2022). Career Prediction Using Naive Bayes. *2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT)*. doi:https://doi.org/10.1109/icicict54557.2022.9917745.
- Leslie, D. (2019). Understanding artificial intelligence ethics and safety: A guide for the responsible design and implementation of AI systems in the public sector. The Alan Turing Institute. doi:https://doi.org/10.5281/zenodo.3240529

- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V. and Allen, P. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. [online] Available at: https://arxiv.org/pdf/1907.11692.pdf.
- Longstreet, D. (2012). *How to calculate linear regression using least square method*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=JvS2triCgOY&ab_channel=statisticsf un.
- Mao, J. and Liu, W. (2019). A BERT-based Approach for Automatic Humor Detection and Scoring. *IberLEF@SEPLN*, pp.197–202.
- McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, pp.56–61. doi:https://doi.org/10.25080/majora-92bf1922-00a.
- Meaney, J., Wilson, S., Chiruzzo, L., Lopez, A. and Magdy, W. (2021). *SemEval-2021 Task 7: HaHackathon, Detecting and Rating Humor and Offense*. [online] pp.105–119. Available at: https://aclanthology.org/2021.semeval-1.9.pdf [Accessed 2 May 2023].
- Metsis, V., Androutsopoulos, I. and Paliouras, G., 2006, July. Spam filtering with naive bayes-which naive bayes?. In *CEAS* (Vol. 17, pp. 28-69).
- Mihalcea, R. and Strapparava, C. (2005). *Making Computers Laugh: Investigations in Automatic Humor Recognition*. [online] *ACL Anthology*. Association for Computational Linguistics. Available at: https://aclanthology.org/H05-1067.pdf.
- Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. [online] Available at: https://arxiv.org/pdf/1301.3781.pdf.
- Morishita, T., Morio, G., Ozaki, H. and Miyoshi, T. (2020). Hitachi at SemEval-2020 Task 7: Stacking at Scale with Heterogeneous Language Models for Humor Recognition. *International Conference on Computational Linguistics*. doi:https://doi.org/10.18653/v1/2020.semeval-1.101.
- Patel, K., Mathkar, M., Maniar, S., Mehta, A. and Natu, Prof.S. (2021). To laugh or not to laugh – LSTM based humor detection approach. *2021*

12th International Conference on Computing Communication and Networking Technologies (ICCCNT). [online] doi:https://doi.org/10.1109/icccnt51525.2021.9580124.

- Pedregosa, F., Buitinck, L., Louppe, G., Grisel, O., Varoquaux, G. and Mueller, A. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85), pp.2825–2830.
- Pennington, J., Socher, R. and Manning, C. (2014). *GloVe: Global Vectors for Word Representation*. [online] Association for Computational Linguistics, pp.1532–1543. Available at: https://aclanthology.org/D14-1162.pdf.
- Probst, P., Boulesteix, A.L. and Bischl, B., 2019. Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, *20*(1), pp.1934-1965.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. *OpenAI blog*, [online] 1(8), p.9. Available at: https://life-extension.github.io/2020/05/27/GPT%E6%8A%80%E6%9C%AF%E5%88%9D%E6%8E%A2/language-models.pdf [Accessed 23 Aug. 2022].
- Raha, T., Upadhyay, I.S., Mamidi, R. and Varma, V. (2021). *IIITH at SemEval-2021 Task 7: Leveraging transformer-based humourous and offensive text detection architectures using lexical and hurtlex features and task adaptive pretraining*. [online] ACLWeb. doi:https://doi.org/10.18653/v1/2021.semeval-1.173.
- Raskin, V. (1979). Semantic Mechanisms of Humor. *Annual Meeting of the Berkeley Linguistics Society*, 5, p.325. doi:https://doi.org/10.3765/bls.v5i0.2164.
- Riggio, C. (2019). *What's the deal with Accuracy, Precision, Recall and F1?* [online] Medium. Available at: https://towardsdatascience.com/whats-the-deal-with-accuracy-precision-recall-and-f1-f5d8b4db1021 [Accessed 13 Dec. 2022].
- Russell, S. and Norvig, P. (2010). *Artificial intelligence: a modern approach*. 3rd ed. New Jersey: Pearson.

- Smola, A.J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, [online] 14(3), pp.199–222. doi:https://doi.org/10.1023/b:stco.0000035301.49549.88.
- Song, F., Liu, S. and Yang, J. (2005). A comparative study on text representation schemes in text categorization. *Pattern Analysis and Applications*, 8(1-2), pp.199–209. doi:https://doi.org/10.1007/s10044-005-0256-3.
- Toman, M., Tesar, R. and Jezek, K. (2006). *Influence of Word Normalization on Text Classification*. [online] *ResearchGate*. Available at: https://www.researchgate.net/publication/250030718_Influence_of_Word_Normalization_on_Text_Classification.
- Utkarsh (2023). *Continuous Bag of Words (CBOW) Model in NLP*. [online] Scaler Topics. Available at: https://www.scaler.com/topics/nlp/cbow/ [Accessed 1 May 2023].
- Uysal, A.K. and Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management*, [online] 50(1), pp.104–112. doi:https://doi.org/10.1016/j.ipm.2013.08.006.
- Veale, T. and Hao, Y. (2010). Detecting Ironic Intent in Creative Comparisons. *ECAI 2010*, [online] pp.765–770. doi:https://doi.org/10.3233/978-1-60750-606-5-765.
- Yang, Z., Hooshmand, S. and Hirschberg, J. (2021). CHoRaL: Collecting Humor Reaction Labels from Millions of Social Media Users. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. doi:https://doi.org/10.18653/v1/2021.emnlp-main.364.
- Yse, D.L. (2021). *Text Normalization for Natural Language Processing (NLP)*. [online] Medium. Available at: https://towardsdatascience.com/text-normalization-for-natural-language-processing-nlp-70a314bfa646.
- Zach (2021). *MSE vs. RMSE: Which Metric Should You Use?* [online] Statology. Available at: https://www.statology.org/mse-vs-rmse/ [Accessed 10 Mar. 2023].
- Zhang, T.-T., Chen, Z. and Lan, M. (2020). ECNU at SemEval-2020 Task 7: Assessing Humor in Edited News Headlines Using BiLSTM with

Attention. *International Conference on Computational Linguistics*. doi:https://doi.org/10.18653/v1/2020.semeval-1.129.

●