UNIVERSITY OF BIRMINGHAM

Classifying Clickbait: A Comparative Analysis of Machine Learning Models for Headline Similarity Detection

Student Name: Kenya Williams

[redacted]

MSc Data Science

School of Computer Science, University of Birmingham

[redacted]

Word Count: 10501

[redacted]

Abstract

This research project investigates the limitations of existing commercial media verification tools and advocates for enhanced NLP model integration to better serve the news consumption habits of the general public. I introduce the concept of "clickbait" and examine user-oriented solutions to combat misinformation online.

Through a systematic comparison, I determine that there is a trade-off between accuracy and computational efficiency when evaluating machine learning models. The methodology of my project demonstrate that efficiency must often be prioritised due to the resource demands of models like BERT and Word2Vec. As a result, I identify TF-IDF vectorisation and as a more accessible embedding-combination approach for commercial media verification tools.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, [redacted] for his continuous, patient support and guidance throughout the various stages of my project. I am also thankful to [redacted] for his valuable feedback during the project demonstration stage. I am particularly grateful to [redacted] for their generous scholarship. I extend my thanks to all the faculty members involved in the MSc Data Science programme t the University of Birmingham, for providing a comprehensive and rigorous curriculum. Without the support and encouragement of these individuals and institutions, I would not have been able to gain as much knowledge and insight as I have. Their contributions have been instrumental in my academic growth and development.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 CONTEXT AND CHALLENGES

Computational social science is a rapidly evolving interdisciplinary academic field that leverages computer science techniques to analyse social phenomena. Emerging in the late twentieth century, it was officially popularised as a term in 2009 by David Lazer, a professor of political science and computer and information science [1]. This growth has been driven by the increasing availability of digital data, including "social media, administrative records, and historical archives [which are used to] develop theories of human behaviour" [2]. The changing landscape of news consumption has also fuelled the need for computational approaches to media analysis, particularly in addressing challenges like misinformation and sensationalism in online news. A significant challenge in this area is the proliferation of 'clickbait' – misleading headlines designed to attract attention at the expense of accuracy. In 2016, clickbait headlines accounted for 25.27% of all online headlines on Facebook [3]. Data science offers promising classification-based solutions to improve media verification by automatically detecting and limiting the spread of such content. This research project will investigate the contributions of data science in forming the foundation for detecting headline similarity, a crucial step toward identifying clickbait in online media sources.

1.2 KEY CONCEPTS

It is useful to clarify some relevant terms. This should refine the specific focus of my research project.

Misinformation: Misinformation occurs when content misleads the reader; contributing to a distorted perception of events that do not align with reality. Even when an article's content is factual, clickbait and dissimilar headlines between news sources can misrepresent the main focus of an event. This can lead readers to incorrect conclusions or cause frustration when the headline obfuscates the information they originally sought out. Misinformation significantly influences public perception and decision-making.

Sensationalism: Sensationalism is a tactic that aims to engage consumers through shocking or exciting headlines. It is misleading because it often exaggerates the significance of events, manipulating emotions in ways that can be considered journalistically irresponsible [4]. For example, sensationalism can use fearmongering language, which may lead to real-world consequences by inciting unnecessary fear or panic.

Bias: Media bias manifests through the selective framing of events, either positively or negatively, often reflecting political leanings or partial perspectives. This complicates media reporting, particularly in relation to headline similarity, as news sources may emphasise certain aspects of an event at the expense of providing an objective overview. Bias becomes more problematic when an event lacks objective or impartial reporting, leaving readers to decide which elements from different sources are trustworthy enough to form an opinion or make a decision. Sensationalism frequently stems from biased reporting.

Fake News: Fake news refers to news that is either fabricated or unsubstantiated. It lacks verification, often omitting references, sources or quotes, making it difficult to trust [5]. Fake news is a major focus of media verification efforts because it poses significant challenges to ensuring accurate and reliable dissemination of information.

Media Verification: Media verification involves fact-checking the claims made in articles. According to the City University of New York's School of Journalism, five key 'pillars' must be determined [6] [7].

1. PROVENANCE: Are you looking at the original account, article, or piece of content?
2. SOURCE: Who created the account or article, or captured the original piece of content?
3. DATE: When was it created?
4. LOCATION: Where was the account established, the website created, or piece of content captured?

5. MOTIVATION: Why was the account established, the website created, or the piece of content captured?

These factors ensure that online news content is accurate and reliable for publication.

Headline vs. Content: This research project focuses on the similarity between headlines alone. Headlines are particularly susceptible to sensationalism due to the emotionally and politically charged language they can contain. It is both possible and necessary to assess their bias, given that:

1. Headlines can be disjointed from the article's actual content (as seen with clickbait).
2. Headlines are often the first point of contact that readers have when encountering news on social media. Therefore, detecting misinformation or limited perspectives in headlines is critical for ensuring accurate reporting.

Similar vs. Dissimilar: In this research project, similar headlines are those that discuss the same theme, event, and event details, while dissimilar headlines report different event details. This aligns with the data source used in this project, which categorises headlines into nine groups of similar themes. This project uses a strict binary classification, mirroring the binary nature of the training examples used to train the models.

- Example: An entry in my training data illustrates this distinction: 'headline_a' reads "EU Parliament grills Zuckerberg - but Facebook CEO slips away without giving solid answers", while 'headline_b' reads "EU data protection regulation comes into force". Although both headlines share the theme of European GDPR reforms, the former focuses on a specific event, while the latter marks a significant regulatory milestone.

Accuracy vs Precision

Accuracy refers to the proportion of correctly classified headline pairs out of the total number of classifications. In this project, accuracy measures how many headline pairs are correctly classified as similar or dissimilar. It is calculated as:

True Positive (TP) – Headline pairs classified as similar (1) that are actually similar (1).

True Negative (TN) - Headline pairs classified as dissimilar (0)that are actually dissimilar (0).

False Positive (FP) - Headline pairs classified as similar (1) that are actually dissimilar (0).

False Negative (FN) - Headline pairs classified as dissimilar (0) that are actually similar (1).

Precision refers to the proportion of all the model's positive classifications that are truly positive. In this project, precision measures how many of the headline pairs classified as similar are actually similar [8].

Both accuracy and precision are important metrics for evaluating the performance of the embedding-model combinations in this project. However, I will prioritise accuracy as it provides an overall measurement of performance. For the purpose of detecting clickbait, identifying dissimilar headlines is just as important as identifying similar ones, thereby making accuracy the more suitable primary metric.

Machine Learning (Supervised vs. Unsupervised, State-of-the-Art):

Machine Learning: Machine learning refers to algorithms that predict outcomes without being explicitly programmed for each task.

Supervised Machine Learning: This project focuses on supervised learning methods, as the dataset is labelled, enabling predictions on headline similarity.

Unsupervised Machine Learning: In contrast, unsupervised learning does not require labelled data. Instead, it clusters or categorises similar training examples without predefined labels. Some studies in Natural Language Understanding (NLU) use unsupervised methods to

group unlabelled news content, such as the work of Laban et al [News Headline Grouping as a Challenging NLU Task (2021)][9].

State-of-the-Art: State-of-the-art is defined as "the highest level of development at a particular time (especially the present time)" [10]. My final approach attempts to utilise one state-of-the-art machine learning method by incorporating novel approaches that differ from traditional techniques.

1.3 MOTIVATION

This research project focuses on applying text similarity techniques to headlines alone, addressing a significant gap in the current landscape of media verification. Headlines, along with eye-catching advertisements and thumbnails, play a pivotal role in shaping online readers' snap judgements. They often feature emotionally charged language and selective details that news sources choose to emphasise. In an environment where critical thinking must be rapid, accurately assessing the similarity between headlines is essential for combatting misinformation and recognising bias.

What distinguishes this project is its focus on identifying the most effective supervised learning models and embedding techniques for detecting similarity between headlines, particularly those covering the same themes or events. While previous studies have explored the alignment of headlines with their content, my research emphasises the headline-to-headline comparison, laying the groundwork for more advanced media verification processes.

Additionally, this project offers potential applications for unsupervised clustering techniques, which could automatically group headlines that share similar perspectives or biases. This would be particularly valuable for identifying clusters of new sources with shared political leanings, thereby enhancing the ability to monitor media integrity. Although websites like Media Bias/Fact Check manually log these biases, my project aims to streamline this process through supervised learning methods, acting as a precursor to further innovations in this area[11]. Ultimately, this systematic comparison serves as a foundational step in the broader

pursuit of evaluating clickbait, facilitating media verification, and ensuring the integrity of journalism in an increasingly digitalised and unregulated information landscape. This research contributes to the evolving field of media integrity as computational verification methods (see: Computational Journalism) become indispensable.

1.4 Research Questions

This project is guided by the following key research questions:

1. Which evaluation metrics, beyond accuracy, are most effective at determining or correlating with correct headline similarity classification?
2. Which supervised learning models perform best in classifying headline pairs?
3. Which popular or readily available embedding techniques facilitate the most accurate classification?
4. Which combinations of embedding techniques and supervised learning models achieve accurate results with low computational cost?
5. What is the impact and role of cosine similarity in facilitating accurate classification?
6. Can a Large Language Model produce better headline similarity classification results while being transparent in its decision-making process?

These questions aim to make it easier for media analysts and non-experts to detect dissimilarity between headlines, thereby enhancing the accessibility and usability of media verification tools.

1.5 Objectives

1. Embedding Techniques Selection
    - Select three popular and accessible embedding or vectorisation techniques.
    - Conduct hyperparameter tuning for each embedding technique to maximise their effectiveness in text classification.

2. Model Selection
    - Narrow down to three supervised learning models that demonstrate strong performance in text classification and are accessible to non-experts in machine learning.
    - Make an informed prediction of the best-performing model based on domain knowledge and initial results.

3. Hyperparameter Tuning
    - Perform hyperparameter tuning on the selected models to optimise their performance.

4. Integration of Models and Embedding
    - Apply the optimised embedding techniques to the selected three supervised learning models.
    - Rank the performance of each embedding-model combination based on accuracy and computational cost, providing a comprehensive comparison.

5. Cosine Similarity Integration
    - Incorporate cosine similarity into the embedding process and assess its impact by creating a baseline model (without cosine similarity) to models using cosine similarity on vectored data.

6. Testing and Validation
    - Test the performance of the optimised models on a separate testing dataset, noting accuracy score and other relevant metrics.

7. Evaluate Model Performance
    - Calculate key evaluation metrics (accuracy, precision, recall, F1-score, and confusion matrix) for each combination of embedding techniques and supervised learning models.

8. Correlation Analysis
    - Identify which evaluation metrics demonstrate a strong correlation with accuracy scores.

- Define criteria for what constitutes a ‘strong correlation’ in this context, based on Pearson correlation standards.

9. Large Language Model Exploration
    - Select a Large Language Model (e.g., ChatGPT) and evaluate its ability to accurately predict headline pair similarity.
    - Compare the performance of the Large Language Model against traditional methods in terms of accuracy, transparency, and computational cost.

1.6 REPORT ORGANISATION

This report is organised into six sections to systematically compare embedding techniques and models for determining headline similarity.

Section 2: Literature Review – This section reviews developments in Natural Language Processing (NLP) within the social sciences, particularly in journalism, and discusses their limitations.

Section 3: Legal, Social and Professional Issues – Here, I consider the legal, social, ethical, and professional implications of the project.

Section 4: Methodology - This section details the methodologies used in the project, including data collection and data preprocessing stages.

Section 5: Evaluation – This section covers the setup of the experiment, the configuration of the models and evaluation of the results.

Section 6: Conclusion – In the concluding section, I assess the successes and limitations of my approach and propose potential directions for future work.

Section 7: Project Management – Finally, this section logs the timelines, milestones, and meeting that contributed to fulfilling the objectives of this research project.

2 LITERATURE REVIEW

2.1 INTRODUCTION

This literature review explores the existing body of research related to headline similarity in Natural Language Processing (NLP), focusing on comparing different approaches and assessing their relevance to media verification and the issue of clickbait. The review is divided into four sections:

1. The Role of NLP in Media Verification: This section presents an overview of current tools used for media and news verification, particularly those that incorporate NLP techniques. It provides a broad context for understanding the landscape of media verification technologies.
2. The Limitations of Existing NLP Tools for Media Verification: This section summarises the key limitations identified in these existing approaches and systems, specifically concerning the issue of clickbait, highlighting gaps that this gap aims to address. I mention how my project builds on existing tools and its significance for advancing the field of media verification.

This review will establish the foundational knowledge necessary to understand the current challenges in headline similarity detection and situate this project's aims and methodology within the broader landscape of NLP and computational social science.

2.2 THE ROLE OF NATURAL LANGUAGE PROCESSING IN MEDIA VERIFICATION

The spread of misinformation and clickbait has necessitated the development of sophisticated media verification systems. These systems are essential for maintaining the integrity of information in an environment where news is rapidly disseminated across multiple platforms. While these systems are increasingly necessary, many remain in their nascent stages and often require significant manual input. For instance, on the social media platform X, formerly

known as Twitter, the success of the ‘Community Notes’ feature – where users append context to posts that are misleading or inaccurate – relies heavily on user contributions [12].

Conversely, Natural Language Processing leverages machine learning to interpret, comprehend and manipulate human language, offering the potential for automated and accurate analysis and classification of textual content. Given its capabilities, it is crucial to explore both the current functionality and future potential of NLP in enhancing the effectiveness of prominent news platforms and social networks in combatting misinformation.

Media verification systems are primarily designed to assess the credibility and accuracy of news content for journalists. Existing commercial systems mainly focus on verifying report details through image verification and detecting original content to avoid plagiarism. For instance, news-specific versions of Google Reverse Image Search compare images with billion-image datasets to detect unauthorised replication of photojournalistic work and corresponding written reports [13]. Tools that specialise in capturing image and video metadata also assist journalists with fact-checking [14].

While these systems employ machine learning for image detection, they do not incorporate Natural Language Processing. Their primary use lies in increasing efficiency for journalists, particularly within their guidelines of professional ethics. Evidently, these tools are largely tailored to specialists in news reporting and do not address the broader challenges posed by misinformation from non-professionals, such as marketers who prioritise engagement over content integrity, or individuals sharing posts with friends and family.

Moreover, these tools do little to alleviate the concerns of readers who faced with multiple news sources covering the same event, as they do not assist in the process of selecting and distinguishing between conflicting reports. In short, they do not facilitate or advise when

exercising critical thinking. Importantly, these tools also miss the added benefits of processing textual data, which remains the predominant form of news dissemination [15].

While images undoubtably play a significant role in generating clickbait-eliciting emotional responses, journalists are more professionally concerned with issues of plagiarism and attribution. Headlines, however, serve as a more explicit form of misleading content where clickbait occurs, as they directly influence the reader's expectations of the article's content. As a result, existing commercial systems fail to appeal to the consumer side and do not capitalise on the potential of NLP technology to process large volumes of textual data. This gap persists despite a wealth of studies in academic spaces that demonstrate the relevance of NLP in media verification [16] [17].

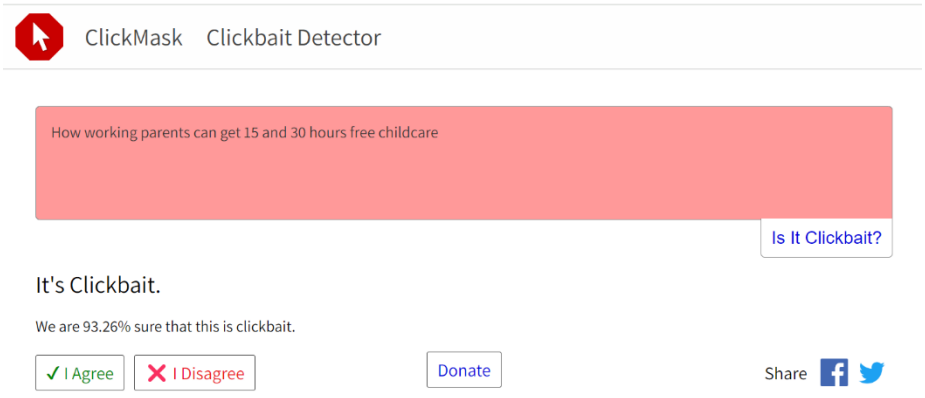
2.3 Limitations of Existing Natural Language Processing Tools for Media Verification

Nevertheless, tools that apply Natural Language Processing (NLP) to media verification and attempt to address challenges posed by clickbait *are* available to users. However, these systems often exhibit limitations that hinder their effectiveness in more dynamic and unregulated spaces, such as social media.

One of the most visible applications is found in news search engines that group stories based on the similarity of their headlines. These systems use NLP classification to sort and display related news articles, making it easier for users to find diverse perspectives on the same event. The widespread use of NLP in search engines is a positive development, given its potential to enhance media literacy by encouraging comparison across sources. However, their application is largely confined to more regulated environments like search engine results pages, leaving significant gaps in the often-unregulated spaces of social media and the spontaneous appearance of advertisements, where clickbait is more prevalent. In these contexts, misinformation can spread unchecked, and the lack of agile NLP-driven verification tools presents a critical weakness.

Another area where NLP has been applied is in the development of individually created plugins and browser extensions designed to detect and block clickbait content. These tools have varied success rates and present a couple of weaknesses, particularly in defining what constitutes clickbait. Consider the use case of 'ClickMask,' a plugin that claims to detect clickbait by analysing headlines [18]. Users input a headline and press a button labelled "Is it clickbait?" to determine whether the headline qualifies as clickbait. In a personal test of this tool, ten headlines from the BBC—a reputable news source known for its relative neutrality—were analysed. The results indicated that the majority of these headlines were not classified as clickbait, with the plugin providing a percentage to quantify its certainty. However, a noticeable trend emerged: headlines containing quotations or newly coined terms were disproportionately flagged as clickbait. This suggests a bias within the tool against newer vocabulary or phrases, raising concerns about the accuracy and reliability of its classifications.

One of the main limitations of tools like ClickMask is their failure to consider the content of the articles associated with the headlines. The plugin requires the URL of the article whereas the webpage does not, leading to instances where it judges the headline in isolation. This approach is problematic because a headline alone cannot definitively indicate whether the article it leads to is clickbait. A headline may be sensational or vague, but if the article content is accurate and informative, the label may be unjustified. This disconnect between headline and content undermines the credibility of the webpage's assessments and highlights a weakness in the tool’s design.



ClickMask claims to use neural networks to determine clickbait, basing its analysis on whether a headline is (i) vague, (ii) emotionally charged, or (iii) misleading. While this may simplify decision-making for users by removing content that meets these criteria, it also limits the opportunity for personal judgment and critical thinking. This becomes particularly concerning when false positives (FP) occur, as the option to review the wrongly categorised article is unavailable. This automatic blocking not only reduces user agency but also raises issues related to press freedom.

The reliance on emotional language as an indicator of clickbait is another point of contention. While emotionally charged language often correlates with sensationalism and, by extension, clickbait, the two are not synonymous. Sensationalism can exist without the intent to mislead, and therefore, should not be the sole criterion for labelling content as clickbait. Bias, for example, may influence readers to consider a specific political perspective, but it is distinct from clickbait. For instance, a headline that evokes a strong reaction may be truthful and correspond with the article's content, challenging the clickbait label.

In summary, while existing systems like news search engines and plugins such as ClickMask offer modern approaches to media verification, they fall short in three areas. These systems (i) lack the ability to operate within the unregulated spaces of social media, (ii) fail to consider the context provided by article content, and (iii) sometimes mislead users themselves by inaccurately categorising content. The implementation of NLP in these tools introduces new challenges, such as potential biases in classification and concerns about press freedom. These gaps highlight the need for more flexible, accurate, and context-aware NLP-driven systems in order to address the complexities of clickbait detection in the digital age.

In my project, I commit to the narrower goal of determining similarity between headlines. This addresses the initial challenge of selecting news sources when presented with multiple options that discuss similar events. I believe this approach is a precursor to the more substantial task of predicting clickbait, which must consider the contributory factors of emotionally loaded language and misleading connections between headlines and content [19]. Given that media verification tools catering to non-journalists are still in their early stages, it

is vital to confirm that the models used to classify similarity are as accurate as possible. Once this accuracy has been established, we can confidently move on to incorporating content analysis and sentiment analysis into the verification process.

3 LEGAL, SOCIAL, ETHICAL AND PROFESSIONAL ISSUES

This research project does not present any serious legal, social, ethical or professional issues. The dataset used is publicly available dataset and sourced from the website Papers With Code. Each headline in the dataset is accompanied by its corresponding URL and publication date, ensuring proper news source attribution.

While some headlines in the dataset address sensitive and distressing topics such as abortion (timeline 2) and fatalities (timeline 9), the systematic comparison conducted in this project does not involve direct engagement with these subjects. The text data is tokenised and converted into vector representations during preprocessing for model training, which abstracts the semantic content and minimises any potential ethical implications.

4 METHODOLOGY

4.1 PROBLEM DEFINITION

This project undertakes a systematic comparison of supervised learning models and their respective embeddings, with the goal of optimising these models for the highest possible accuracy in the context of headline similarity detection. The primary objective is to identify which combinations of models and embeddings can most effectively classify headlines as similar or dissimilar, thereby setting the foundation for application-based tools in media verification.

The outcomes of this investigation are intended to contribute to the development of straightforward models that can be implemented in commercial tools. Such tools would

empower users to apply critical thinking skills more selectively when distinguishing between various news sources, particularly in an era where misinformation and clickbait are prevalent.

Ultimately, this project is motivated by the need to bridge the existing gap between advanced academic research on headline similarity and its practical applications. By focusing on models that are both accurate and accessible, the project aims to provide a stronger foundation for the use of Natural Language Processing in user-oriented media verification systems. This work aims to enhance the reliability of machine learning-drive textual analysis, making it more applicable to real-world scenarios where users must make sense of an increasingly complex media landscape.

4.2 4.2 DATA COLLECTION

The dataset used in this project originates from the study titled '*News Headline Grouping as a challenging NLU Task*' by Philippe Laban, Lucas Bandarkar, and Marti A. Hearst (2021) [9]. This dataset was constructed to develop an unsupervised model capable of surpassing the accuracy of the best-performing supervised learning models. It is particularly relevant to my research as it not only establishes traditional approaches but also envisions future applications for evaluating and comparing different supervised learning approaches.

The dataset is organised into three files: Training (train.json), Testing (test.json), Validation (dev.json). These files contain the following key columns:

- **timeline_id:** A unique identifier for each timeline grouping, allowing for the categorisation of related headlines.
- **headline_a:** The first headline in the comparison.
- **headline_b**: The second headline in the comparison.
- **date_a:** The publication date of headline_a.
- **date_b:** The publication date of headline_b.
- **url_a:** The URL where headline_a was published.

- **url_b:** The URL where headline_b was published.
- **label:** A binary label indicating whether the headlines are considered similar (1) or dissimilar (0).

I selected this dataset for the following reasons. First, it was designed with the express aim of improving upon the accuracy of supervised learning methods, which aligns directly with the goals of my project. Although the original study focused on exceeding supervised learning performance through unsupervised learning, the dataset itself provides a basis for evaluating the accuracy of various supervised models. This allows me to focus on the impact of different embeddings and vectorisation techniques on the performance of these models.

Furthermore, the dataset structures itself around a binary classification task that makes it suitable for my investigation. The binary label facilitates a straightforward evaluation of model accuracy, which is critical for my systematic comparison.

Additionally, by organising the dataset by 'timeline_id', it categorises news headlines into common themes. This feature refines the potential performance of the models, as it allows for headlines to be compared that discuss the same event but may vary in temporal context, emphasis, or detail. This categorisation improves accuracy for assessing the similarity between headlines, as it accounts for the nuances in how news stories are reported over time.

Using this dataset ensures that if my results deviate from those expected from the best-performing supervised models, it will be easier for future projects to adjust the focus of the Natural Language Understanding investigations. This could lead to the development of state-of-the-art unsupervised models that build upon my findings.

From a quantitative perspective, the dataset meets other criteria that lend credence to my investigation: it contains over 20000 examples before preprocessing and train-test splitting, has no null entries, and includes reliable human determinations of headline similarity. These factors increase my confidence in using this dataset to train and validate the accuracy of the embedding-model combinations in my study.

4.3 DATA PREPROCESSING

Data preprocessing was critical step in preparing the dataset of news headlines for analysis and model training. The initial objective was to convert the file type into a readable format and prepare the dataset for exploratory data analysis (EDA).

1. **Mounting Google Drive**:
    - **Purpose:** The dataset files were stored in Google Drive, and I selected Google Colab as my chosen notebook environment. To access these files within the Colab environment, it was necessary to mount Google Drive.
    - **Action:** Google Drive was successfully mounted, allowing direct access to the dataset files.

2. **Verification of File Locations**:
    - **Purpose:** It was necessary to verify the presence of the required files—train.json, test.json, and dev.json – before loading them into the notebook.
    - **Action:** The correct directory was specified, and a directory listing command was executed to confirm that the files were located in the correct pathway: "/content/drive/My Drive/MSc Individual Project".

3. **Library Importation**:
    - **Purpose:** Various Python libraries were imported to facilitate data loading and manipulation. These included *pandas* for data handling, *json* for reading JSON files and and matplotlib and seaborn for data visualization
    - **Action:** The necessary libraries were successfully imported into the environment, enabling the subsequent file loading and conversion tasks.

4. **Loading and Converting Datasets**:

    - **Purpose**: The original datasets were provided in JSON format. To streamline data manipulation and analysis and produce a more readable tabular format, these JSON files were converted into CSV format, which is more compatible with *pandas* for data handling and enabling further preprocessing steps.

    - **Action:** A function was implemented to convert the JSON files (train.json, test.json, and dev.json) into CSV files (train.csv, test.csv, and dev.csv), and they were assigned to the following DataFrame variables: train_df, test_df, dev_df. For convenience, the validation dataset (dev_df) was disregarded from this point.

5. **DataFrame Verification**:

    - **Purpose**: To ensure that the datasets were correctly loaded and converted, I confirmed the data types of the resulting DataFrames.

    - **Action**: The type() function was used to verify that the datasets were successfully loaded as pandas DataFrames.

At this stage, the datasets were successfully prepared and loaded for exploratory data analysis, model training, and evaluation phases of the project.

4.4 EXPLORATORY DATA ANALYSIS

To understand the data's structure and ensure its readiness for analysis, I conducted an Exploratory Data Analysis (EDA) using a custom function ('inspect_data(df, name)'). This function inspected the fundamental characteristics of each dataset (train_df and test_df) and identified potential issues relating to data integrity, distribution and label imbalance.

4.4.1 Dataset Structure and Overview

The dataset consists of 15492 rows and 8 columns. A visual inspection of the first five rows confirmed that the data was properly formatted for analysis.

	timeline_id	headline_a	headline_b	date_a	date_b	url_a	url_b	label
0	9	Seven bodies found after dam burst at Brazil m...	Fears rise for 300 missing in Brazil dam disas...	2019-01-25	2019-01-26	https://www.reuters.com/article/us-brazil-vale...	https://timesofindia.indiatimes.com/world/rest...	0
1	9	Dam collapses in Brazil: mud sludge leaves ove...	Brazil dam collapse: At least seven dead and h...	2019-01-26	2019-01-26	http://en.mercopress.com/2019/01/26/dam-collap...	https://www.independent.co.uk/news/world/ameri...	1
2	2	Remember Savita': father's plea for voters to ...	Ireland referendum could lift strict ban on ab...	2018-05-23	2018-05-25	https://www.theguardian.com/world/2018/may/23/...	http://www.foxnews.com/world/2018/05/25/irelan...	0
3	1	Whistleblower Snowden can apply for Russian pa...	Barack Obama commutes sentence of Puerto Rican...	2017-01-18	2017-01-18	http://www.reuters.com/article/us-usa-snowden-...	http://www.independent.co.uk/news/us-president...	0
4	9	Brazilian despair turns to anger as toll from ...	Brazil dam: Startling pictures of Brumadinho c...	2019-01-28	2019-02-01	https://www.reuters.com/article/us-vale-sa-dis...	https://www.bbc.co.uk/news/world-latin-america...	0

4.4.2 Data Integrity Checks

Missing Values: No missing values were detected across any of the datasets, confirming data completeness.

Duplicate Entries: No duplicate rows were identified, confirming data integrity.

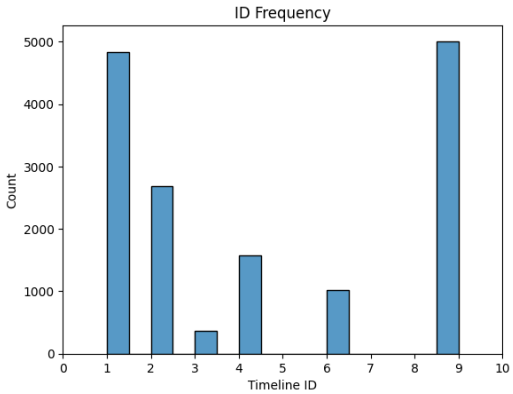
Unique Values: The number of unique entries in each column was calculated to understand the variability in the data.

- ‘timeline_id’: 6 unique values, which is expected as the training dataset contains 6 timelines, with the remaining 3 allocated to the testing and validation files.
- ‘headline_a’: 936 unique entries
- ‘headline_b’: 940 unique entries
- ‘date_a’: 150 unique entries
- ‘date_b’: 158 unique entries
- ‘url_a’: 951 unique entries
- ‘url_b’ : 955 unique entries
- ‘label’: 2 unique values (0 and 1), as expected in a binary classification

Initially, the lower number of unique entries in the headline and corresponding URL and date columns was concerning , as it suggested limited variability. However, this is resolved by reflecting that the variety arises from combining unique headlines with each other, providing sufficient diversity for model training. The extent of the variation is to be assessed in the ‘Assessment of Duplicates’ section.

Timeline Distribution

The ‘timeline_id’ feature which categorises headlines into common themes, showed that the most common timeline is “9: Brazil Dam Collapse (Brumadinho, 25th January 2019)” with 5011 entries, while “3: Bird Flu Outbreak (USA, March 2017)” had the least with 366 counts.



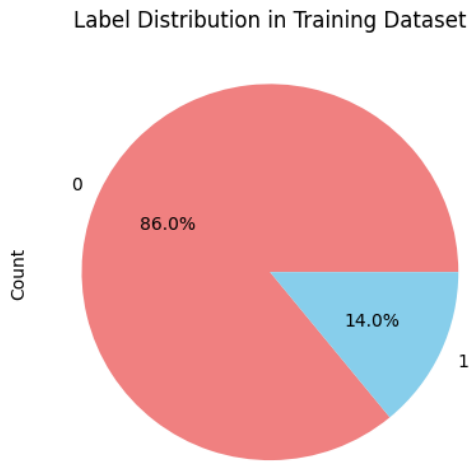
Correlation Matrix

A correlation matrix was generated to show the correlation coefficients between the numerical features – ‘timeline_id’ and ‘label’. The low correlation coefficient (0.005) between these features indicate minimal linear relationship, suggesting that no timeline disproportionately contains more similar or dissimilar headlines compared to others.

	timeline_id	label
timeline_id	1.000000	0.005051
label	0.005051	1.000000

Label Imbalance Analysis

The dataset revealed significant class imbalance, with 85.97% of the entries classified as dissimilar (0) and only 14.03% as similar (1), resulting in an imbalance ratio of 6.13. This imbalance could introduce bias in each model’s predictions and this balance could be exaggerated by the binary nature of the dataset’s output [20].



4.3.3 Assessment of Duplicates

To avoid redundancy, I conducted several checks to identify and handle duplicate rows, identical headline pairs, and swapped headline pairs.

Identical Rows

During the initial Data Integrity Checks, I confirmed that no duplicate rows were present in the dataset. However, I performed an additional verification to ensure accuracy. Identical rows are rows where all column values, including both headline pairs, match exactly with another row.

- **Action:** I used the duplicated() function to identify any fully duplicated rows.
- **Findings:** No fully duplicated rows were found, indicating that each row in the dataset is unique in terms of both headlines and other attributes like URLs and dates.

Identical Headlines in 'headline_a' and 'headline_b'

Next, I checked for instances where the headlines in a pair were identical. In these cases, both ‘headline_a’ and ‘headline_b’ are the same, providing no useful information for model training, as the model would not be required to differentiate between distinct headlines.

- **Action:** I flagged and removed rows where ‘headline_a’ was identical to ‘headline_b’.
- **Findings:** A total of 23 rows contained identical headlines in both columns, and these were successfully removed from the dataset.

Swapped Headlines

In some cases, the same headline pair may appear in reverse order—where headline_a of one row is the same as headline_b in another row, and vice versa. These "swapped" duplicates do not contribute new information and should also be removed.

- **Action:** I created a temporary column called combined_headline, where the two headlines were concatenated in alphabetical order to identify swapped pairs. Duplicates in this column were removed, while retaining instances labelled as "similar" (1) to avoid losing important information.
- **Findings:** After removing 361 swapped headline pairs and retaining at least one instance of each combination, the dataset was further refined, significantly reducing redundancies.

Final Cleanup and Validation

After handling the duplicate and swapped headline pairs, the final step was to verify the uniqueness of all headline combinations.

- **Action:** I grouped headline pairs and verified that no combination appeared more than once, followed by removing the the combined_headline column to restore the dataset to its original structure.
- **Findings:** The dataset was confirmed to contain only unique headline pairs, ensuring that duplicate or redundant entries remained.

This thorough cleaning process, which was also applied to the testing DataFrame, resulted in a significantly cleaner and more reliable dataset, now ready for the final stage of preprocessing and model training. Significantly, this process alleviated concerns about the scarcity of unique headlines, as the updated dataset structure (15286 rows and 8 columns) now exclusively represents unique headline combinations.

4.4.3 Tokenisation, Stop Word Removal and Lemmatization

Tokenisation

Tokenisation is the process that splits text down into individual words or subwords based on boundaries determined by tokenisation algorithms. These algorithms defined where one word or subword starts and ends, making the text more suitable for machine learning in NLP by representing it in a structured format. In my constructed function, I apply the 'punkt' variable from the NLTK toolkit [21]:

words = word_tokenize(text)

Tokens create new features that models can be trained on, enhancing the text's suitability for machine learning applications.

Stop Word Removal

Stop word removal is a preprocessing techniques that eliminates common words, known as stop words, which have little or no semantic meaning. Words typically usually considered stopwords include [22]:

- **Articles** – e.g., a, an, the, this, these, those
- **Conjunctions** – e.g. with, and, or, but
- **Pronouns** – my, your, his, her, their

In my function, I used the standard collection of English stop words provided by the NLTK library:

stop_words = set(stopwords.words('english'))

By removing these low-value words, I can improve performance of models that trained with BERT and Word2Vec embeddings. This allows these embeddings to focus on more informative semantic and contextual relationships between words. Additionally, stop word removal can improve computational efficiency by reducing the vocabulary size, which in turn speeds up training and prediction times.

Lemmatisation

Lemmatisation is the process of reducing words to their root or base form, standardising similar words with different grammatical variations, such as plural forms or verb tenses. In my function, I used the WordNetLemmatizer from the NLTK toolkit:

lemmatized_words = [lemmatizer.lemmatize(word) for word in filtered_words]

The lemmatisation algorithm analyses the word's structure and context to identify its morphological features [23], effectively reducing words to their base form. This process further reduces computational cost by decreasing the dataset's vocabulary size. It also helps capture the meaning of words, which is particularly beneficial for models using TF-IDF vectorisation, where understanding the root meaning of words can enhance the accuracy of classification.

4.5 SELECTING AND FINETUNING MODELS

In this section, I will present the different models, embeddings and measures I use to design my experiment.

Synthetic Minority Oversampling Technique (SMOTE)

SMOTE is a technique that addresses imbalance in datasets. It generates new data points for my minority class (similar headlines - 0) by generating synthetic data points that are equidistant between existing minority data points.

Cosine Similarity

Cosine similarity is used to compare the vector constructed by the embeddings and vectorisations. It calculates the cosine of the angle between the vectors, with a value closer to 1 indicating that two vectors are similar.

Grid Search

Grid Search searches through my specified grid of hyperparameter values and trains each model with each combination of hyperparameters and selects the best-performing set based on my preferred metric of accuracy.

4.5.1 Support Vector Machines (SVM)

Support Vector Machines are a supervised learning algorithm commonly used for classification and regression tasks on labelled data. They function by identifying an optimal 'hyperplane' that separates data points into distinct classes. The process can be segmented into the following stages:

1. **Kernel Trick and Transformation**: The data points are mapped into a higher-dimensional space using the kernel trick. This transformation makes it easier to separate the data points into different classes.
2. **Optimal Hyperplane Calculation**: The algorithm calculates the optimal position of the hyperplane position, which is defined as the one that maximises either the margin or the distance between the classes. The aim is to find the hyperplane that has the largest margin, thus providing the best separation between similar and dissimilar classes.

3. **Classification:** The transformed data points are classified based on their distance from the optimal hyperplane, thereby assigning them to their respective classes.

Technical Considerations

The mathematical concepts that that SVM are composed of include Lagrange multipliers, support vectors and the decision function. The decision function is represented as [24]:

$$f(x)=w^Tx+b$$

Where w is the weight vector, x is the input feature vector, and b is the bias term.

Rationale for Selection

I have selected SVM for this project because it is well-suited to handling classification tasks, which is central to the objective of accurately predicting similar and dissimilar headline pairs.

The margin maximisation characteristic helps withstand the risk of overfitting – a common problem where a model becomes too closely tailored to the training data and its noise, leading to poor generalisation on unseen datasets. Given that my systematic comparison aims to produce results that are applicable beyond the NLU dataset, and to be useful for commercial clickbait detection tools, SVM is a promising option.

To optimise the model's performance while managing its computational cost, I will approach hyperparameter tuning with careful consideration. Although I will begin with common starting points, significant attention will be given to parameters such as regularisation and kernel type, as these have a significant impact on the model's performance.

Hyperparameter Tuning

```
svm_param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear'],
```

```
    'gamma': ['scale'],
}
```

I intend to apply Grid Search iteratively to each of model's parameter inputs to lessen the computational load. For SVM, I start with the following modest ranges and then refine them progressively with GridSearch until they cannot be refined further.

'C', regularisation: This parameter penalises the model for misclassifying headlines, where larger values impose harsher penalties. I chose a broad range of values commonly used during NLP training to ensure the model is adequately tuned.

kernel: This parameter determines the type of function used by the Support Vector Machine. I start with the 'linear' kernel, which maps simpler relationships between features, and then introduce 'rbf' (Radial Basis Function) in later iterations, as it is well-suited for mapping non-linear relationships.

'gamma': This parameter determines the coefficient for non-linear kernels, like 'rbf'. I include both 'scale' and 'auto' but initially select scale because it considers the dataset's variance when calculating the coefficient.

After completing the iterative Grid Search for the best parameter inputs using TF-IDF, the refined parameter grid looks like this:

```
svm_param_grid_final = {
    'C': [0.75],
    'kernel': ['rbf'],
    'gamma': ['scale'],
}
```

4.5.2 Random Forest

Random Forest is a machine learning method that combines the power of multiple decision trees to make predictions. A decision tree is a model that splits data into branches based on feature values, ultimately leading to a classification. In a Random Forest, these decision trees

are independent of one another; each tree is trained on a random subset of the data and its features. This approach helps to reduce the risk of overfitting.

The process of Random Forest involves the following stages:

1. **Generation of Decision Trees:** Multiple decision trees are generated, each trained on a random subset of the data and its features. This randomised element introduces diversity into the trees.
2. **Individual Tree Predictions:** Each tree makes its prediction of whether the headline pairs are similar or dissimilar based on the data it has been given. Since the trees are trained on different subsets, their predictions may initially vary.
3. **Ensemble Voting:** The final classification is decided through an 'ensemble' method, usually a majority vote among the individual decision trees. The prediction with the most votes is selected as the final output.

Technical Calculations

Random Forest is calculated by the averaging of the results of its individual trees. This can be mathematically represented with the following formula:

$$y\hat{} = N1 \sum i=1 N fi(x)$$

where *y* is the predicted output, *N* is the number of trees and f*i*(x) represents the prediction of the *i*th tree.

Rationale for Selection

I have selected Random Forest for this project because of its ability to mitigate the risk of overfitting, similar to the advantages offered by SVM. Random Forest also provides insights into feature importance, which is especially valuable for refining the models based on which features contribute most to accurate classification. This feedback mechanism supports my goal of producing models that can consistently and accurately group similar headlines, thereby enabling users to make more informed decisions about the trustworthiness of different news sources.

Given the computational demand that generating multiple trees presents, I will consider the model's computational cost by initially referring to common hyperparameter settings. I will then focus on tuning the most impactful parameters to achieve optimal performance.

Hyperparameter Tuning

My initial Random Forest parameter grid appears as:

```
rf_param_grid = {
    'n_estimators': [150],
    'max_depth': [10, 20, 30],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [2],
    'bootstrap': [True],
    'max_features': ['sqrt'],
}
```

After completing the iterative Grid Search for the best parameter inputs using TF-IDF, the parameter grid has the following inputs:

```
rf_param_grid_final = {
    'n_estimators': [120],
    'max_depth': [15],
    'min_samples_split': [2],
    'min_samples_leaf': [2],
    'bootstrap': [True],
    'max_features': ['sqrt'],
}
```

'n_estimators': This parameter specifies that the ideal number of decision trees in a forest are 120.

'max_depth': This parameter permits that trees can have a maximum depth of 15, which is considered relatively deep.

‘**min_samples_split’**: This parameter dictates that the minimum number of samples requires to split an internal node is 2. This means a node can be split if it has at least 2 samples but not when only a single sample is present.

4.5.3 Gradient Boosting

Gradient Boosting, similar to Random Forest, is an ensemble learning technique with a key difference: it builds models sequentially, with each new model correcting the errors of the previous one [25]. This method is effective classification tasks where complex relationships between data points exist. In this project, Gradient Boosting is used to predict the classes oof headline pairs, aiming for improved accuracy with each iteration.

Data points are separated by a loss function, which is calculated and refined with each model that is passed through.

The process of Gradient Boosting involves the following stages:

1. **Initial Model Creation**: The process begins by creating a simple model, such as a decision tree, to make initial predictions.
2. **Sequential Model Fitting**: New models are fitted to the errors, known as residuals, of the previous model, with each new model attempting to correct these errors.
3. **Ensemble Combination**: The predictions of all models are combined using a weighted sum. The contribution of each model is determined by its performance, resulting in a final prediction.

Technical Considerations

Gradient Boosting relies on two concepts: loss function and gradient descent. The loss function measures the error between predicted and actual values and the gradient descent minimises this error iteratively by updating the model’s parameters. The updates occur so that each new model reduces the final model’s overall prediction error, ensuring that classifications become progressively more accurate.

Rationale for Selection

I have selected Gradient Boosting for this project because it offers an improvement on Random Forest by employing sequential learning, where each model is dependent on the corrections of the previous one. This approach is better suited for capturing complex relationships between textual data, such as the nuances when headlines may refer to the same event but with different temporal or contextual details. For this reason, Gradient Boosting's precision in classification is particularly advantageous.

However, this precise interdependence between models introduces the risk of overfitting, especially if too many trees are used. To mitigate this risk, I will limit the number of trees and carefully tune other parameters that have the most significant impact on the model's performance, such as learning rate and maximum depth, while managing computational costs effectively.

I will use XGBoost, as it is a highly efficient variation of the Gradient Boosting and is known for its performance and scalability.

Hyperparameter Tuning

My initial XGBoost parameter grid appears as:

```
xgb_param_grid = {
    'n_estimators': [50, 100, 200], #
    'learning_rate': [0.01, 0.1], #
    'max_depth': [3, 5, 7], #
    'min_child_weight': [3],
    'subsample': [0.8],
    'colsample_bytree': [0.8],
    'gamma': [0.1],
    'reg_alpha': [0.1],
    'reg_lambda': [0.01],
}
```

After completing the iterative Grid Search for the best parameter inputs using TF-IDF, the parameter grid has the following inputs:

```
xgb_param_grid_final = {
    'n_estimators': [160], #
    'learning_rate': [0.1], #
    'max_depth': [5], #
    'min_child_weight': [3],
    'subsample': [0.8],
    'colsample_bytree': [0.8],
    'gamma': [0.1],
    'reg_alpha': [0.1],
    'reg_lambda': [0.01],
}
```

'n_estimators': This parameter specifies that the ensemble model will use 160 trees.

'learning_rate': This parameter sets the learning rate to 0.1.

'max_depth': This parameter limits the maximum depth of each tree to 5.

4.5.4 TF-IDF Vectorisation

TF-IDF stands for Term Frequency-Inverse Document Frequency, a vectorisation technique that is widely used in Natural Language Processing. The technique assigns weights to words based on their frequency within a document and their importance across an entire corpus. In the context of my systematic comparison, TF-IDF will evaluate the frequency of the word, such as "Snowden" within a specific document (a headline in this case) and determine its importance across all 15,492 rows in the dataset.

The TF-IDF process involves the following steps:

1. **Calculate Term Frequency (TF):** This step measures how frequently a word appears in a document. The more frequent a word appears in a documents, the higher the TF value it has.

2. **Calculate Inverse Document Frequency (IDF):** This step evaluates the importance of a word across the entire corpus. Words that are common across many documents receive lower IDF scores. Words that are rarer receive higher scores.
3. **Compute TF-IDF:** This final step multiplies TF by IDF to assign a weight to each words. This reflects both its frequency in the specific document (headline) and its importance within the corpus (training DataFrame).

Technical Considerations

The TF-IDF calculation is mathematically represented as:

$$TF{-}IDF(t,d) = TF(t,d) \times IDF(t)$$ [26]

Where:

TF(t,d) is the term frequency of term t in document d.

IDF(t): Inverse document frequency of term t.

$$log(N / (1 + Number\ of\ documents\ containing\ term\ t))$$

Rationale for Selection

I have chosen to use TF-IDF due to its simplicity and ease of implementation. The underlying calculation is easy to understand and can be executed using Python programming libraries. TF-IDF also serves as a valuable baseline for comparison with more advanced embedding techniques, such as BERT and Word2Vec, because it does not account for the semantic or contextual relationship between words. The simplicity enables a direct assessment of how more complex methods perform in comparison, both regarding accuracy and time and storage usage.

Given my focus on computational efficiency and ease of implementation, especially for user-oriented tools, the results of models that use using TF-IDF will be highly relevant. If these models achieve comparable accuracy, TF-IDF may be preferred over more sophisticated due its lower computational cost and simplicity.

TF-IDF can be used in conjunction with each of my chosen models, as it appropriately weights words in headline pairs. This weighting usefully captures the significance of certain words that hint at specific events and their temporal context.

Since TF-IDF considers the importance of all words in a corpus, it is necessary to remove stopwords – common words that could skew data representations. I will ensure that stopword removal is conducted to prevent these biases. I will also carry out hyperparameter tuning on TF-IDF to optimise its performance within the models.

Hyperparameter Tuning

I had originally intended to use Grid Search to identify the best parameter settings for each embedding or vectorisation option. Unfortunately, due to frequent server crashes, I had to abandon this approach and limit Grid Search to the machine learning models (SVM, Random Forest, XGBoost).

Instead, I manually selected the following parameters for the TF-IDF vectorization:

max_features=2076

- This parameter limits the vocabulary size to a maximum number of terms.
- I selected 2076 because my dataset contains 2076 unique words across its headline columns. By setting this parameter, I ensure that all unique terms in my dataset, after preprocessing, are included in the vocabulary.

min_df=1

- This parameter filters out terms that appear in fewer than *x* documents.
- I chose min_df=1, which means no terms are filtered out based on document frequency, thus retaining all terms that appear in at least one document.

max_df=0.6

- This parameter filters out terms that appear in more than *x*% of documents.
- I set max_df=0.6, meaning that terms appearing in more than 60% of the headlines are excluded. Frequently occurring terms may not add significant semantic value and could dilute the meaningfulness of the text embeddings. Removing them could enhance the semantic relationships between the remaining terms.

ngram_range=(1, 3)

- N-grams refer to adjacent sequences of *n* items (words) from a given sample of text.
- This parameter is set to consider unigrams (single words), bigrams (groups of two consecutive words), and trigrams (groups of three consecutive words). This exhaustive approach captures a broader range of word combinations, which potentially improves the model's understanding of context within the headlines.

4.5.5 BERT Embedding

BERT (Bidirectional Encoder Representations from Transformers) is a sophisticated embedding technique that considers the meaning of words and the relationships between them by transformer architecture. Developed by Google researchers, BERT is an advancement over traditional embedding techniques like TF-IDF, because it considers the context of words within a sentence, which subsequently creates more accurate and meaningful word representations.

The process of BERT embedding involves the following stages [27]:

1. Tokenisation – Words are broken down into smaller units called tokens. Tokens include not only complete words but sub-words units, for instance “”. Tokens make BERT capable of handling complex words and unknown vocabulary.

2. Positional Encoding – Positional information is attached to these tokens to preserve the order and structure of the original text. Positional encoding ensures that the model understands the sequence of words.
3. Transformer Layers: The positionally-encoded tokens are passed through multiple transformer layers. These layers analyse the contextual relationships between tokens, allowing the model to understand how words interact with each other in a headline.
4. Vector Representation: BERT produces a vector representation for each individual token. These vectors capture both the meaning of the word and its context within the headline.

Technical Considerations

BERT operates on a deep learning framework using transformers. It considers the context from the words that come before and after each words when creating embeddings. The bidirectional context is what makes BERT distinctive and facilitates nuanced understanding of language to do a task like headline similarity detection.

Rationale for Selection

I have selected BERT embedding as part of my systematic comparison because it is regarded as a state-of-the-art model that outperforms traditional word embedding techniques. In the context of my project, where the aim is to compare the linguistic content of headline pairs, using a model that actively considers meanings and contextual relationships between words allows for a more accurate and nuanced analysis than simpler methods like TF-IDF.

BERT embedding represents a huge step towards automating what has traditionally been done manually – understanding the context and meaning behind words to determine similarity. This makes BERT an ideal choice for developing a user-oriented tool for social media platforms, where media verification is desperately needed.

One concern with BERT is its computational cost. Despite the availability of pre-trained models, BERT remains resource-intensive, requiring substantial storage and processing power. This could limit the extent of my investigation, as well as it practical application in scenarios where computational resources are constrained. Therefore, my evaluation will include assessing the trade-offs between BERT's accuracy and its computational demands to determine its viability for commercial use.

Hyperparameter Tuning

For BERT embedding, there are limited tuning options available since the model is pre-trained. However, I chose to switch from the conventional 'bert-base-uncased' model to 'distilbert-base-uncased' due to the reduced storage requirements of the latter. This change may result in a trade-off, sacrificing some model accuracy or depth of language understanding for improved efficiency and lower computational costs.

4.5.6 Word2Vec Embedding

Word2Vec is an embedding technique that makes use of neural networks to learn high-quality word embeddings by capturing the context in which words appear. Unlike similar models, Word2Vec focuses on the meaning of words and their relationships with a large corpus. These considerations make it effective at attaining semantic understanding for my headline similarity detection task [28].

The process of Word2Vec embedding involves the following stages:

1. Create Neural Network: Word2Vec typically uses a shallow neural network architecture, consisting of an input layer, a hidden layer and an output layer. The architecture is designed to capture semantic relationships between words.
2. Train the Model: The model is trained on a large corpus of text, where it predicts target words based on surrounding context words. This can be achieved using either Continuous Bag of Words or Skip-Gram.

3. Extract Embeddings: The learned weights of the neural network are extracted as word embeddings. These embeddings represent words in a continuous vector space; semantically similar words are located closer to each other.

Technical Considerations

Word2Vec trains the neural network on large datasets and uses techniques such as negative sampling and hierarchical softmax to optimise the learning process and produce linguistically meaningful word representations.

Rationale for Selection

I have selected Word2Vec for my systematic comparison because it captures the semantic relationship between words. Its inclusion in my analysis allows for a direct comparison between embedding techniques that focus on semantic relationships (i.e. Word2Vec) and those that focus on contextual relationship (i.e. BERT).

While Word2Vec is considered a state-of-the-art model, it does come with a relatively high computational cost, particularly during the training phase. In addition to evaluating the accuracy of the models using Word2Vec, I will also assess the time required for training and testing. This will factor into my evaluation of its overall value to user-oriented classification tasks, where efficiency and speed are important considerations.

I will carry out hyperparameter tuning to optimise the performance of Word2Vec, focusing on parameters such as the size of the context window. These adjustments will enhance the model's accuracy and make it more suited to the requirements of my project.

Hyperparameter Tuning

For Word2Vec embedding, I selected the following hyperparameter inputs:

vector_size=100: This parameter sets the dimensionality of the the word vectors to 100, a common choice within the domain of NLP. It offers a balance between capturing meaningful semantic relationships and managing computational complexity.

window=5: This parameter sets the maximum distance between the current and predicted word within a sentence to 5. It allows the model to consider a broader context around each word, which can improve the quality of the word embeddings by capturing more contextual information.

min_count=1: This parameter instructs that all words, even those appearing only once, are included in the training process. This allows the model to learn from the entire vocabulary within the dataset.

workers=4: I selected this parameter to match the number of CPU cores on my computer. It optimises the training speed by using all available cores.

5 EVALUATION

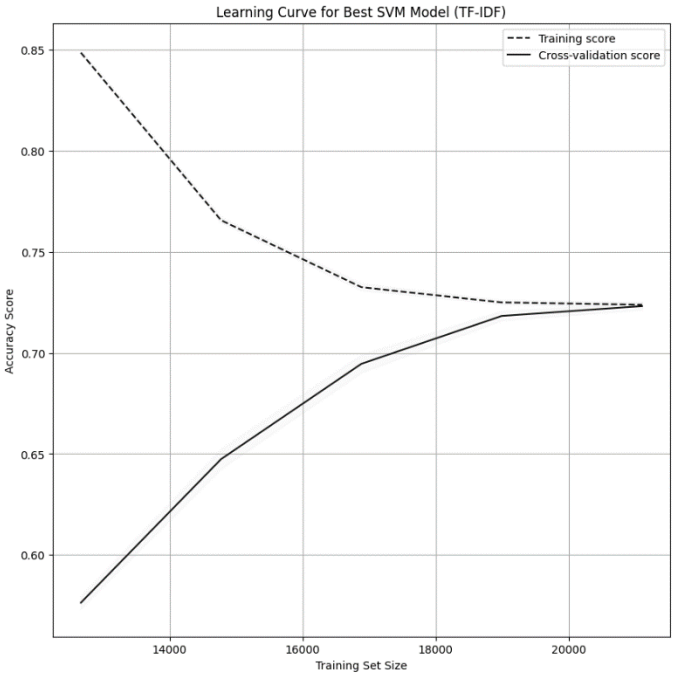
5.1 TF-IDF + MODEL RESULTS

5.1.1 TF-IDF + SVM

For Support Vector Machines:

- The initial accuracy of the model is: 0.700098608362811
- The accuracy of the most refined model is: 0.723220247277333

The refined parameter choices have made a sizeable difference to the proportion of correctly classified headlines for SVM.



I have plotted the learning curve for each model's performance. This represents the relationship between the size of the training dataset and the model's accuracy. The x-axis maps the increase in the training dataset size, and the y-axis maps the change in accuracy score. The unbroken line represents the training accuracy, and the dotted line represents the cross-validation accuracy, which reflects how well the separately trained subsets performs.

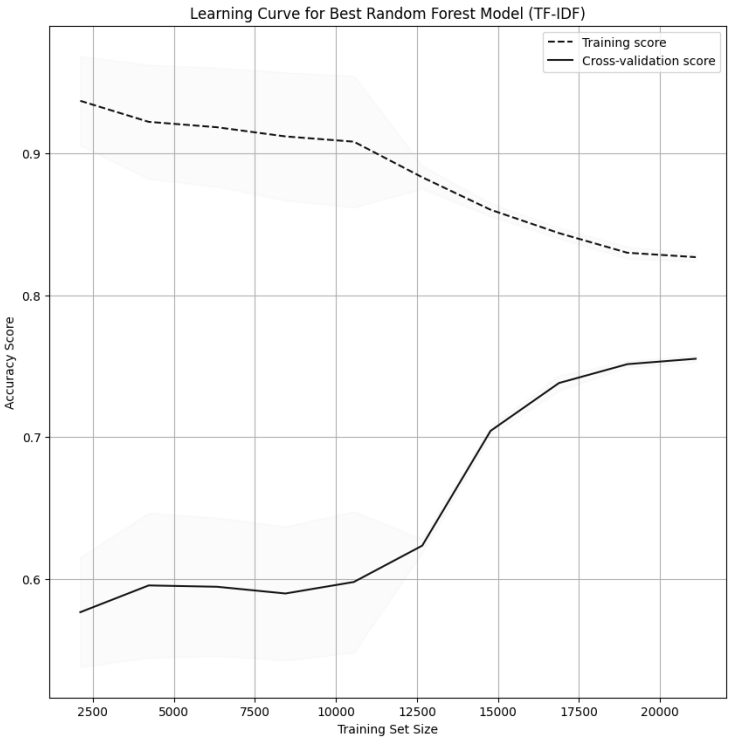
In this instance, the SVM model's cross-validation accuracy increases as the training set size grows, plateauing at around 0.72. Conversely, the training accuracy decreases as the training set size increases. This is an indication the model is avoiding generalising better; it avoids overfitting as it is exposed to more data.

5.1.2 TF-IDF + RF

For Random Forest:

- The initial accuracy of the model is: 0.751155931091525.
- The accuracy of the most refined model is: 0.755287669622978

Hyperparameter tuning has made a marginal improvement in the proportion of correctly classified headlines for Random Forest.

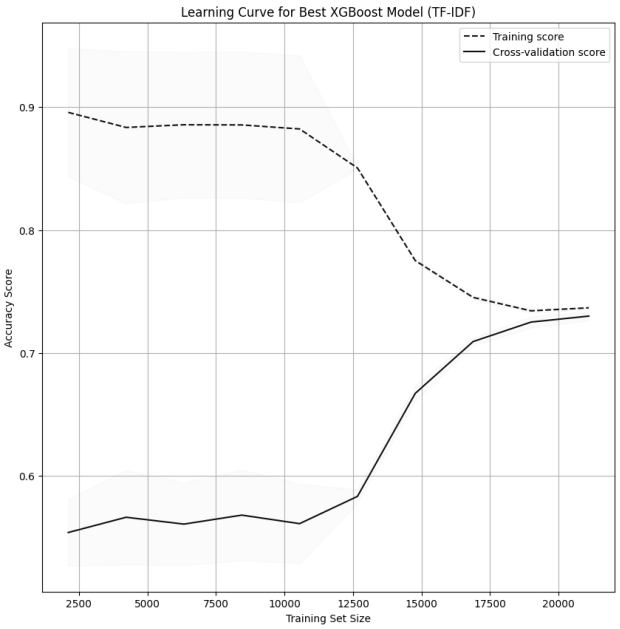


In this instance, the Random Forest model's cross-validation accuracy increases as the training set size exceeds 12500, after which it plateaus at around 0.75. Conversely, the training accuracy decreases as the training set size increases, but it does so at a slower and more linear rate. This is an indication the model is gradually generalising better while maintaining a relatively stable performance.

TF-IDF + XGB
For Gradient Boosting (XGBoost):

- The initial accuracy of the model is: 0.7297020787565247
- The accuracy of the refined model is: 0.7301569041729576

The refined parameters have resulted in a slight improvement in the proportion of correctly classified headlines for Gradient Boosting.



In this instance, the XGBoost model's cross-validation accuracy increases sharply as the training set size exceeds 12500, after which it plateaus at around 0.73. Conversely, the training accuracy decreases sharply beyond 12500 as the training set size increases. This is an indication the model is effectively reducing overfitting and achieving better generalisation when faced with unseen data.

5.2 CONCLUSION

The results of my TF-IDF vectorised training shows that the Random Forest model produces the highest level of accuracy.

6 REFERENCES

1. *Computational Social Science - the Decision Lab*. (n.d.). The Decision Lab. https://thedecisionlab.com/reference-guide/computer-science/computational-social-science

2. Edelmann, A., Wolff, T., Montagne, D., & Bail, C. A. (2020). Computational Social Science and Sociology. *Annual Review of Sociology*, *46*(1), 61–81. https://doi.org/10.1146/annurev-soc-121919-054621

3. Rony, M. M. U., Hassan, N., & Yousuf, M. (2017). Diving Deep into Clickbaits. *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. https://doi.org/10.1145/3110025.3110054

4. *Sensationalism in media*. (n.d.). https://reporter.rit.edu/news/sensationalism-media

5. *Research Guides: "Fake News," Lies and Propaganda: How to Sort Fact from Fiction: What is "Fake News"?* (n.d.). https://guides.lib.umich.edu/fakenews

6. *LibGuides: Fact Checking & Verification for Reporting: Social Media Verification*. (n.d.). https://researchguides.journalism.cuny.edu/factchecking-verification/UGC-verification

7. alastair@firstdraftnews.com. (2020, September 29). *Verifying online information: The absolute essentials*. First Draft. https://firstdraftnews.org/articles/verifying-online-information-the-absolute-essentials/

8. *Classification: Accuracy, recall, precision, and related metrics*. (n.d.). Google for Developers. https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

9. Laban, P., Bandarkar, L., & Hearst, M. A. (2021). News Headline Grouping as a Challenging NLU Task. -. https://doi.org/10.18653/v1/2021.naacl-main.255

10. https://www.vocabulary.com/dictionary/state-of-the-art

11. Media Bias Fact Check. (2024, February 19). *Media Bias/Fact Check news*. Media Bias/Fact Check. https://mediabiasfactcheck.com/

12. https://help.x.com/en/using-x/community-notes

13. B, I. (2020, July 30). *7 verification tools for better fact-checking*. Reuters News Agency. https://www.reutersagency.com/en/reuters-community/7-verification-tools-for-better-fact-checking/

14. *InVID Verification Plugin - InVID project*. (2024, January 3). InVID Project. https://www.invid-project.eu/tools-and-services/invid-verification-plugin/

15. "Most people, found the survey, get their news from the website of professional publishers (47%), followed by TV (35%), social media (33%), radio (24%), and printed publications (15%)."

Majid, A. (2023, April 24). Where do Britons get their news? Publisher websites and TV most popular sources. *Press Gazette*. https://pressgazette.co.uk/media-audience-and-business-data/media_metrics/traditional-outlets-still-top-uk-news-sources-survey/#:~:text=Most%20people%2C%20found%20the%20survey,and%20printed%20publications%20(15%25).

16. Peter Bourgonje, Julian Moreno Schneider and Georg Rehm (2017), From Clickbait to Fake News Detection: An Approach based on Detecting the Stance of Headlines to Articles, Second workshop on Natural Language Processing meets Journalism

17. Sophie Chesney, Maria Liakata, Massimo Poesio and Matthew Purver (2017), Incongruent Headlines: Yet Another Way to Mislead Your Readers, Second workshop on Natural Language Processing meets Journalism

18. *FAQ: What is Clickbait? What is ClickMask? How does ClickMask work?* (n.d.). https://clickmask.com/clickbait_faq.html

19. Luis Romero Gomez, Tess Watt, Kehinde O. Babaagba, Christos Chrysoulas, Aydin Homay, Raghuraman Rangarajan, and Xiaodong Liu. 2023. Emotion Recognition on Social Media Using Natural Language Processing (NLP) Techniques. In Proceedings of the 2023 6th International Conference on Information Science and Systems (ICISS '23). Association for Computing Machinery, New York, NY, USA, 113–118. https://doi.org/10.1145/3625156.3625173

20. *Datasets: Imbalanced datasets*. (n.d.). Google for Developers. https://developers.google.com/machine-learning/crash-course/overfitting/imbalanced-datasets

21. https://www.nltk.org/index.html https://www.askpython.com/python-modules/nltk-punkt

22. *NLTK's list of english stopwords*. (n.d.). Gist. https://gist.github.com/sebleier/554280

23. Anderson, C., Bjorkman, B., Denis, D., Doner, J., Grant, M., Sanders, N., & Taniguchi, A. (2022, February 28). *5.1 What is morphology?* Pressbooks. https://ecampusontario.pressbooks.pub/essentialsoflinguistics2/chapter/5-1-what-is-morphology/

24. Kowalczyk, A. (2023, April 30). *SVM - Understanding the math : the optimal hyperplane*. SVM Tutorial. https://www.svm-tutorial.com/2015/06/svm-understanding-math-part-3/

25. Glen, S. (2019, July 28). *Decision Tree vs Random Forest vs Gradient Boosting Machines: Explained Simply*. Data Science Central. https://www.datasciencecentral.com/decision-tree-vs-random-forest-vs-boosted-trees-explained/

26. KNIMETV. (2022, September 30). *Data Science pronto! - TF-IDF* [Video]. YouTube. https://www.youtube.com/watch?v=C3V2Lf1Y9Qk

27. *BERT 101 - state of the art NLP model explained*. (n.d.). https://huggingface.co/blog/bert-101

Karani, D. (2022, December 28). Introduction to Word Embedding and Word2VEC - towards data science. *Medium*. https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa