

Duck News Reporters: Automated fake news detection through contextual similarity comparison

COMP9491: Applied Artificial Intelligence — Project Report








Dhruv Agrawal
z5361800@unsw.edu.au

Duke Nguyen
z5398432@unsw.edu.au

Jim Tang
z5208565@unsw.edu.au

August 7, 2023

Todo list

	[Introduction] Dhruv: Add intro citations	3
	[Methods/Feature — Non-latent features] Duke: We can move parts of this section here and below minus the ANOVA methodology to Non-Latent Results	8
	[Methods/Feature — Similarity model] Duke: Explain the point of similarity model: Hypothesis - we hypothesize that articles that are real vs false are more dissimilar than articles that are both real. We assume context articles are real	8
	[Methods/Feature — Similarity model] Jim: I think we can also reframe this section as: now that we have the extracted summary, we want to find related articles using those summaries. And then describe how Google PageRank fits our needs and so on, and therefore we use it then describe our methods. Because on a technical level, all we need is a dataset where we can fetch related articles from, and algorithm to look up article using query. GoogleNews satisfies both these requirements, but there could be other services that could do the same thing as well. Also we assume that context articles are REAL so maybe that's related to why we pick GoogleNews over say Brave News Brave Search or something because it might not necessarily be an algorithm that outputs popular things at the top.	9
	[Results and discussion] Jim: Add table of parameters of models	18
	[Conclusion] Summarise the study and discuss directions for future improvement	18
	[Individual contributions] Dhruv: ~1pg detailing individual contributions	24

Contents

1	Introduction	3
2	Related work	4
2.1	Similarity Comparison using Titles	4
2.2	Similarity Comparison using Fixed News Database	4
2.3	Linguistic and Additional Features	4
3	Methods	5
3.1	Preprocessing and tokenization	5
3.2	Feature — BERT embeddings	6
3.3	Feature — Non-latent features	6
3.4	Feature — Similarity model	8
3.4.1	Summary extraction	8
3.4.2	Article scraping	9
3.4.3	Article vectorisation	11
3.4.4	Similarity metric calculation	11
3.4.5	Similarity metric selection	12
3.4.6	Comparison Results	13
3.5	Feature Normalisation	13
3.6	Model — Machine learning	14
3.7	Model — Neural networks	14
4	Experimental setup	15
4.1	Dataset	15
4.2	Data Analysis Using PCA and KMeans	15
4.3	Evaluation metrics	17
5	Results and discussion	17
6	Conclusion	18
6.1	Attributions	19
6.2	Limitations and Future Work	19
Appendix A Individual contributions		24
A.1	Jim	24
A.2	Dhruv	24
A.3	Duke	24
Appendix B Article scraping		25
Appendix C Word Count Histogram		26
Appendix D Non-Latent Features		27
Appendix E Non-latent Feature Pearson Correlation Matrix		29
Appendix F ANOVA of Non-Latent Feature against Labels		30
Appendix G Machine learning		31

1 Introduction

[Introduction] Dhruv: Add intro citations

As the distribution of news shifts towards social media, there is a rise in the dissemination of fake news. We define fake news as the creation of information presented as legitimate journalism that describes a fictitious event or fabricates details of an event in an attempt to mislead readers. This phenomenon became a major focal point in journalism during the 2016 US elections with political parties labelling many publications and articles as fake news. There are two huge issues with how this occurred in 2016:

1. Many prominent political figures highlighted any news they disagreed with as fake news. This led to the political isolation of the party, whereby any news that looked at them unfavourably had the potential to be dismissed as fake news. This reduced the accountability of political figures in the US, a country where federal legislation has a sweeping impact across the country and the rest of the world.
2. There was a lack of fake news detection tools on social media and due to the polarisation of the media climate, it was extremely difficult for social media to regulate articles published on the platforms or remove factually incorrect articles posted or shared by politicians.

Since then, there have been many attempts to introduce ways to deal with these issues such as Politifact which manually reviews articles and social media posts for factual correctness. It posts its findings on its website and is easily accessible for people. Other similar websites exist but the reason manual fact-checking tools are not as prominent in spaces with high amounts of fake news is because it is impossible for manual review tools to scale to the number of news articles and journalistic social media posts published every day.

There are also many automated fake news detection algorithms which rely on linguistic features of the text, comparisons between the title of the article and its content, the medium of transmission, and any suspicious activity linked with its engagement online. These tools have become more effective since COVID and Twitter employs its own automated algorithms to automatically label articles about certain topics it chooses as fake news. However, these tools are known to be very unreliable as it is increasingly common for fake news to read the same way as real news articles and be shared by humans.

Therefore, in order for fake news detection to become more widespread and effective in combating fake news, there are a few different criteria it must fulfil:

1. The algorithms need to automatically classify news as real or fake so that they can scale with the growth of social media and the increase in fake news dissemination.
2. The algorithms need to incorporate current methods of fake news detection as these have been highly researched and are effective in many situations such as when fake news has been automatically generated or constructed in a highly polarised manner designed to provoke intense responses from readers.
3. An additional feature that looks at the content and meaning of the article beyond how has been written must be used to combat fake news that is well-written and designed to look like real news.
4. The dataset used to train and assess the algorithms must contain real and fake articles that are written in the same style so that it is not apparent simply from the way an article is written whether it is real or fake.

Our approach improves upon existing approaches and aims to combine the above criteria to analyse both the content and style of articles and make a significantly more informed decision on its classification. The model is restricted to a binary classification - it outputs either real or fake rather than giving a confidence metric of an article's legitimacy. This is done as the aim is for the tool to be easily adopted and focusing on the simplicity of the input and output is a priority.

We compiled a list of commonly used linguistic features for fake news detection. Multiple different pairings of features were formed and analysis was conducted to determine the most effective linguistic features for the task. This takes existing research into fake news detection and puts our model in line with current methods. A new feature - similarity - is used to achieve the third criterion above. At a high level, this feature compares the queried article to other articles on Google news which are at the top of Google's searches. As these have high PageRank scores, the model can be confident that they are real articles and it compares the similarity of the content between the queried article and each of these top searched articles. This is done as a way for the model to infer context before making a judgement on an article's legitimacy. This approach brings our model in line with the way humans manually fact-check articles which usually involves finding known trustworthy sources and comparing the content between the articles to determine whether the queried one is consistent with the trustworthy ones.

Through this research, we have analysed and determined the most effective linguistic features for fake news detection and shown that the use of similarity as a metric is effective in building upon these current metrics to increase accuracy. We have also compared the use of the similarity metric with different machine learning classifiers and discovered that it greatly increases the accuracy of less complex machine learning methods and brings their performance in line with complex models.

2 Related work

2.1 Similarity Comparison using Titles

Methods of similarity comparison have been attempted in a few fake news detection papers. Antoun et al. [1] presents an analysis of various fake news detection methods including an automated model winning an international competition on this topic. This model used the title of the queried article to search Google for similar articles and compared the word embeddings for the title against those of the top 5 searched articles. It used cosine similarity only and tested the similarity scores along with other features such as lexicon based features. Many classifiers were tested including SVM, XGBoost and random forest classifiers. This resulted in an effective model that was moderately effective for fake news classification.

2.2 Similarity Comparison using Fixed News Database

A different approach to similarity detection is used in Alsuliman et al. [2] where similarity scores are generated between the queried article and every article in a database of news articles. This method takes the highest similarity score for each queried article and then uses a greedy approach to set a similarity score threshold for real articles that maximises the overall accuracy. This paper uses three well-studied techniques for similarity score calculation - cosine similarity, word appearance and TD-IDF. It explains how these metrics are used within this problem and contrasts the results of each. The most significant limitation of this paper is that the similarity articles that were selected were rarely relevant to the topic of the queried article.

This paper forms the basis of our set of similarity metrics and we analyse each of the three metrics given. Our approach improves upon the methodology of this approach by capitalising on Google's PageRank algorithm to only select articles with a high chance of being relevant to the queried topic. Additionally, our dataset consists of around 10 times as many articles and our classification is done using machine learning and deep learning classifiers rather than a greedy approach to fit a more realistic situation where fake news detection is required.

2.3 Linguistic and Additional Features

Vijayaraghavan et al. [3] presents a series of different fake news models alongside useful linguistic features for this task. It addresses preprocessing techniques including the removal of punctuation and stop words which is an effective technique in this domain. It analyses the polarity of news articles and determines

that both real and fake news have similar polarity distributions making this an ineffective method of distinguishing between them. It also analyses the part of speech distribution between fake and real news and determines that features such as the number of adverbs and adjectives is higher in fake news but the number of nouns and pronouns is higher in real news. This is because fake news relies on descriptive language to establish facts whereas real news refers to research and experts to infer its legitimacy. Zhou & Zafarani produced a comprehensive survey on non-latent features which this research refers to extensively in Section 3.3, along with Garg & Sharma, and Horne & Adali [5, 6, 4].

3 Methods

Figure 1 shows our mostly linear classification pipeline. After preprocessing and tokenization, we extract contextual articles which are fed into a similarity model to form our first feature. Additionally, non-latent features from raw text and BERT embeddings form the rest of our features. The concatenation of all the features are fed into our classification models which infers a binary classification label.

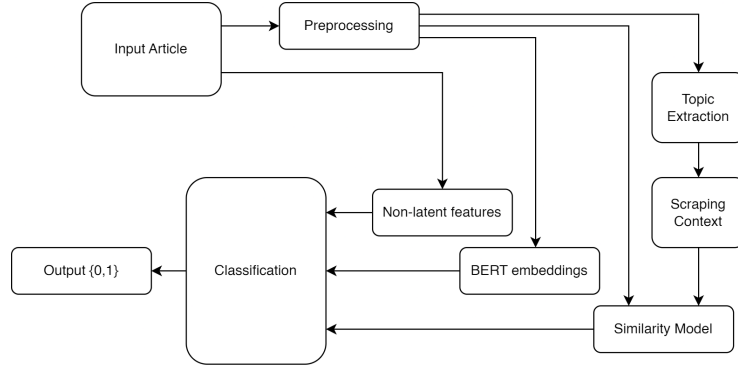


Figure 1: Our classification pipeline.

3.1 Preprocessing and tokenization

Before extracting any features, we will preprocess our input and convert the long form text into tokens. We perform the following preprocessing methods in order:

Remove non-ascii: Our input articles contained unnecessary unicode tokens such as unicode double quotation marks. These can be removed safely since they do not add any extra semantics to the input articles and may confuse feature extraction.

Convert to lowercase: In our research, we converted all text to lowercase. However upon further analysis, converting all text to lowercase hid acronyms such as “US” which could have affected the main themes of the text. Further, all proper nouns such as names and places were also hidden. We will discuss this limitation in Section 6.2.

Lemmatization: We used the `nltk` [7] library to reduce words down to their lemma in the hopes of reducing the complexity within our text which may benefit feature extraction. This looks up the work in the WordNet corpus to get the lemma. Later in the research, we realised that this hypothesis may have not been accurate.

Firstly the `nltk` library we were using does not automatically detect the part of speech and will by default, only lemmatize nouns. While it is arguably better for us to maintain the tense of nouns, we are technically not lemmatizing fully. Secondly, from more research, lemmatization may not be ideal for BERT embeddings since it removes some semantics that could be learnt by the BERT model. We will discuss these limitations further in Section 6.2.

Remove stopwords: Stopwords were removed from the text in order to reduce complexity.

Apart from the above methods, we also tested removing punctuation. However, this was not used in the end since we added non-latent features to measure punctuation counts and also to maintain semantics for BERT.

ID	Article extract	Tokens
118_Real	FBI Director James Comey said Sunday that the bureau won't change the conclusion it made in July after it examined newly revealed emails related to the Hillary Clinton probe. "Based on our review, we have not changed our conclusions that we expressed in July with respect to Secretary Clinton" Comey wrote in a letter to 16 members of Congress. [...]	['fbi', 'director', 'james', 'comey', 'said', 'sunday', 'bureau', 'change', 'conclusion', 'made', 'july', 'examined', 'newly', 'revealed', 'email', 'related', 'hillary', 'clinton', 'probe', '.', ' ', 'based', 'review', ' ', 'changed', 'conclusion', 'expressed', 'july', 'respect', 'secretary', 'clinton', '. . .']
15_Fake	After hearing about 200 Marines left stranded after returning home from Operation Desert Storm back in 1991, Donald J.Trump came to the aid of those Marines by sending one of his planes to Camp Lejuene, North Carolina to transport them back home to their families in Miami, Florida. Corporal Ryan Stickney was amongst the group that was stuck in North Carolina and could not make their way back to their homes. [...]	['hearing', '200', 'marines', 'left', 'stranded', 'returning', 'home', 'operation', 'desert', 'storm', 'back', '1991', ' ', 'donald', 'j', '.', 'trump', 'came', 'aid', 'marines', 'sending', 'one', 'plane', 'camp', 'lejuene', ' ', 'north', 'carolina', 'transport', 'back', 'home', 'family', 'miami', '. . .']

Table 1: Examples of preprocessing and tokenization extraction on items in dataset.

After preprocessing, tokens are then generated based on any whitespace and punctuation in the remaining text. Table 1 shows samples of tokenized input articles.

3.2 Feature — BERT embeddings

BERT (Bidirectional Encoder Representations from Transformers) is a language representation model proposed by Devlin et al [8]. It is pretrained on BookCorpus and the English Wikipedia using masked language modeling (MLM) and next sentence prediction (NSP). MLM masks some of the input tokens with a training objective to predict the masked token simply based on context, and the model also concatenates two sentences with 50% chance of being neighbouring sentences, and the model is pre-trained in the NSP layer to predict if the two are indeed neighbours [8]. BERT obtains SOTA results on several tasks and is suitable for representing document and textual data. Hence, we will be using BERT as the main feature to encode our articles. We will use HuggingFace's 'bert-base-uncased' pretrained model which trains on uncased data. To encode an article, we slice the first 512 tokens of the respective article and pass it through 'bert-base-uncased' and output the CLS token's vector as BERT features for our classification model. The plot in Appendix C shows the length distribution of the articles. We can see that input articles and context articles have means of 1090.58 and 1827.29 respectively, with 50% percentile being 583 and 1032 respectively. This means that the input token size of 512 for BERT cannot encode a significant amount of textual information in an article. However, we hypothesis that the 'fake news' quality of an article is at the minimum visible at a range of 512 tokens. In addition, any information not visible in this range will be captured by **non-latent features** or **similarity metrics**.

3.3 Feature — Non-latent features

From our literature review and survey, we are able to identify a significant amount of non-latent features [5, 6, 4]. After combining features that are similar, and removing features which we cannot calculate due to the need for proprietary softwares (i.e. LIWC), or due to the computational complexity of the algorithms, or related reasons, we are able to identify 81 numerical features suitable for our experiments. After synthesising their categorisations in the original articles, we come up with a grouping of 7 main

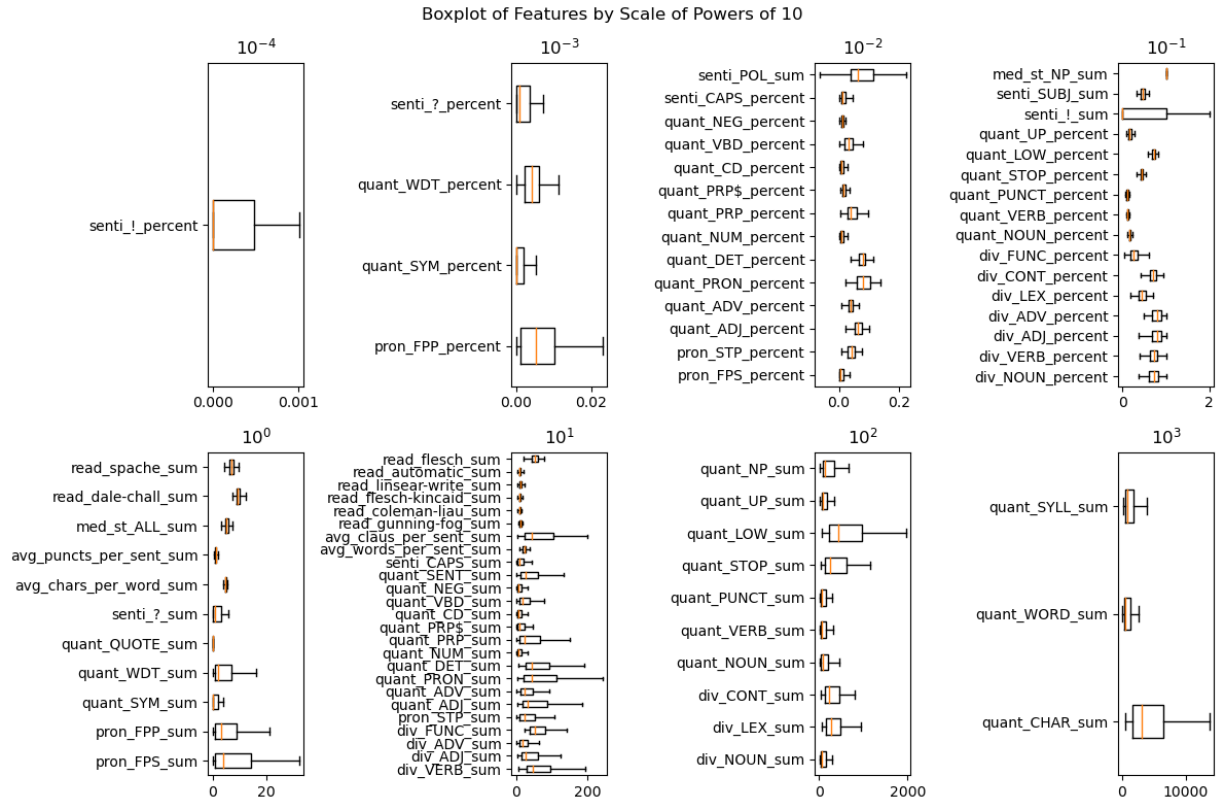


Figure 2: Box Plot of Non-Latent Box Plot Features, displaying the Median, Interquartile Range (IQR), Minimum, Maximum, does not display Outliers which would inflate plot size. Each subplot contains features with mean μ in the corresponding powers of 10.

categories (with their abbreviation in brackets), with each feature being able to be calculated with a few methods:

- Diversity (div): number of unique words, or percentage of all unique words of a particular part of speech (e.g. noun, verb)
- Quantity (quant): number of words, or percentage of all words of a particular part of speech or linguistic unit (e.g. noun, adjective, quote)
- Sentiment (senti): number of linguistic features denoting sentiment (e.g. exclamation mark, all-cap words), or sentiment measurement (polarity, subjectivity)
- Pronoun (pron): number of pronouns of a specific class (e.g. first person singular pronoun: I, me, my, mine)
- Average (avg): average number of linguistic unit **a** per linguistic unit **b** (e.g. characters per word)
- (Median) Syntax Tree Depth (med_st): the median syntax tree depth of a given unit (e.g. median noun phrase syntax tree depth)
- Readability (read): different readability indices (e.g. gunning-fog, coleman-liau)

The feature names are designated using the format "**category**_**featureType**_**calculationMethod**", where **category** is the abbreviated **category**, **featureType** is the feature type within the category. Table

6 contains a full description and explanation of all non-latent features, their category, feature type, and their calculation method. Figure 2 is a box plot of all the non-latent features by their scale of power of 10. The majority of features are between the range of $[10^{-2}, 10^1]$. Several quantity and some diversity features are in the higher range of $[10^2, 10^3]$.

We can then apply ANOVA on these non-latent features against the labels, and filter for those with a p-value less than the α significance level of 0.05. We identify that the removed features account to almost half of all features, and they include readability indices, the syntax tree depth features, and a few features from other categories, and diversity features are all below α . We then apply Pearson correlation amongst the selected features and identify all correlation clusters. The matrix is available in Appendix E. Afterwards, we can apply a selection method on the cluster where we remove all but one with the lowest p-value, specifically, we sort the selected features by p-value in ascending order with the lowest p at the front of the list, we then search for all features which have a correlation of at least 0.95 and remove them from the current list, and continue until the end. Figure 10 shows our results. After grouping them by their original category, we have the following counts, totalling to 29 features, which we will use for our classification models:

- Diversity: 9 features
- Quantity: 12 features
- Pronoun: 2 features
- Sentiment: 3 features
- Average: 1 feature

3.4 Feature — Similarity model

[Methods/Feature — Similarity model] **Duke:** Explain the point of similarity model: Hypothesis - we hypothesize that articles that are real vs false are more dissimilar than articles that are both real. We assume context articles are real

As a novel feature, we investigate the similarity of our input articles to contextual articles found online. In Sections 3.4.1 and 3.4.2 we will discuss our process of gathering three contextual articles from online sources which we treat as truth. These three articles extend our original dataset and are vectorized then fed into a similarity model which we describe in Sections 3.4.4 and 3.4.4. We use this similarity as a feature to ascertain whether our input article contains misinformation.

3.4.1 Summary extraction

To get the context articles, we need to summarize the main topic of our input article down to at most 10 keywords. We use the Python `gensim` [9] library which provides various topic modelling interfaces for text inputs. We use the `ldamodel` which implements Latent Dirichlet Allocation (LDA) to extract a single topic. LDA is a probabilistic model which assumes you have a number of *documents* representing some latent topics characterized by a distribution over words. By feeding in the preprocessed sentences of our input article as each *document*, we are able to get the main themes. We sort the output keywords by the probability they represent the topic then cap the amount of words to 10 at most. We chose LDA because it is a simple algorithm that outputs reproducible and reliable results and can be customized in the number of topics and keywords it generates.

For the scope of our research, we are able to perform manual validation of the summaries extracted to check the summary represented the article content well. Table 2 shows some samples of items in our dataset after applying LDA. We see that while the summaries extracted are not perfect, they still represent the general meaning of the article and were sufficient for the purposes of our research. Two common issues we saw were:

[Methods/Feature — Non-latent features] **Duke:** We can move parts of this section here and below minus the ANOVA methodology to Non-Latent Results

ID	Article extract	Summary
118_Real	FBI Director James Comey said Sunday that the bureau won't change the conclusion it made in July after it examined newly revealed emails related to the Hillary Clinton probe. "Based on our review, we have not changed our conclusions that we expressed in July with respect to Secretary Clinton" Comey wrote in a letter to 16 members of Congress. [...]	email review fbi clinton said july comey news new wa
15_Fake	After hearing about 200 Marines left stranded after returning home from Operation Desert Storm back in 1991, Donald J.Trump came to the aid of those Marines by sending one of his planes to Camp Lejuene, North Carolina to transport them back home to their families in Miami, Florida. Corporal Ryan Stickney was amongst the group that was stuck in North Carolina and could not make their way back to their homes. [...]	home marines trump wa stickney way north plane family

Table 2: Examples of summary extraction on items in dataset.

- Unordered words in the summary — words representing the topics seemed to be unordered. To a human reading the summary by itself, they might be able to see that the words are all keywords of the article but put together in a sentence, will not completely make sense. We hypothesize that this could have caused sub-optimal results when we started scraping articles using the summaries.
- Appearance of stop words and other meaningless non-topic words in the summary — As a flow on issue from our preprocessing, our summary was left with words such as “wa” (from “was”) or “ha” (from “has”). This would have impacted the meaning of our summary and later article scraping.

We will discuss the possibility of extracting better summaries using a more robust model in Section 6.2.

3.4.2 Article scraping

[Methods/Feature — Similarity model] **Jim:** I think we can also reframe this section as: now that we have the extracted summary, we want to find related articles using those summaries. And then describe how Google PageRank fits our needs and so on, and therefore we use it then describe our methods. Because on a technical level, all we need is a dataset where we can fetch related articles from, and algorithm to look up article using query. GoogleNews satisfies both these requirements, but there could be other services that could do the same thing as well. Also we assume that context articles are REAL so maybe that's related to why we pick GoogleNews over say Brave News Brave Search or something because it might not necessarily be an algorithm that outputs popular things at the top.

We feed the summary of the input article into Google News and collect the top three articles. We chose Google News since it is a well know search engine that will return the most popular articles on the internet. This sort of PageRank algorithm is important since we assume that the contextual articles are real and describe what is the current *word-of-mouth-truth* from the internet. Therefore for purposes of comparison, an input article that is very different to our contextual article is likely to be Fake.

For our research, we will only manually feed in all summaries for our dataset. Our motivation for this research was to develop a tool that a user could potentially use to figure out if the current news they are reading contains misinformation. We acknowledge there exists APIs that provide either a wrapper around Google News or implement their own news search algorithm that we could have looked into. However, given the size of the dataset and our scope, this was not necessary to demonstrate our system.

SETUP: We use a virtual machine with a freshly installed latest version of Google Chrome. Searches are conducted in “Incognito Mode” tabs. We also use a VPN to the West coast of the US. These invariants serve the main purpose so that Google’s does not give any personalized

results based on a browser fingerprint or IP address. We chose the US as the VPN destination since our dataset articles were extracted from US news sources and we wanted to scrape for articles with a similar style of writing. If you were to use the tool in Australia, Google would usually return articles from local sources. We restrict our scope to specifically this dataset rather than train on a wide dataset from all sources.

Another invariant we implement is to add a `before:2020` to our summary. This forces Google News to only find articles before this year so that the news we get won't be from recent news. A common discussion topic from our dataset was Donald Trump's 2016 election campaign and we know that the news regarding Trump in 2023 is much different to that of 2016. This makes sense as we are not using a very recent dataset so clamping the date we find contextual articles assumes that if we were looking for fake articles at the time of reading the input article, we wouldn't have too much future articles available.

PROCESS: We attempt to get the top three articles and save the URL for each input article. Not all summaries returned three articles so we perform scraping in three passes:

1. We enter the whole summary without any changes. This is the most ideal approach and most machine-replicable. This covered 70% of our dataset.
2. Still performing only generic actions, we remove any *bad words* or non-important connectives then searched again. This should still be machine-replicable with further work. This covered the next 20% of our dataset.
3. For the last 10% of our dataset, we had to manually look at the input article content and summary generated to figure out why we still received no results. Our hypothesis was that this was a combination of our non-tuned summary extraction and the fact that some *outrageous* Fake articles simply didn't have any similar articles that could be found. We will discuss this limitation in Section 6.2.

From the above passes, we were not able to find context articles for four input articles described in a table in Appendix B. Furthermore, we were only able to find one or two articles for some inputs but we can still continue with our similarity model.

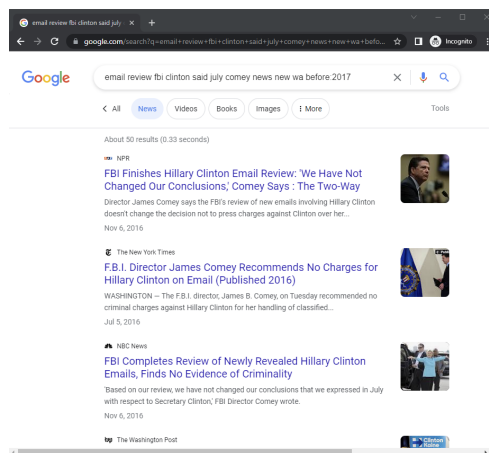


Figure 3: Sample of articles found in Google after searching an article summary.

After gathering three URL links for each context article, we use the Python `newspaper3k` [10] library to download the article and automatically extract its title and content.

3.4.3 Article vectorisation

Past research by Alsuliman, et. al in [2] proposes two different ways to vectorise the articles: TF-IDF, and Word2Vec. In addition to these two vectorisation methods, we propose a third – “*non-latent vectoriser*”.

TF-IDF [11]: The term frequency times inverse document frequency, which is a “*common term weighting scheme in information retrieval*”. The formula given as follows:

$$\frac{\text{article.count}(\text{term})}{\text{len}(\text{article})} * \log_2 \left(\frac{\text{len}(\text{articles})}{\text{df}(\text{articles}, \text{term})} + 1 \right)$$

The first term is the term frequency, and the second term is the inverse document frequency, where `article` is the article we are applying TF-IDF on, `article.count(term)` is the frequency of `term` in `article`, `len(article)` is the number of words in the article, `len(articles)` is the number of articles, `df(articles, term)` is the document frequency of `term` in our `articles` dataset, or the number of articles that contains this term [11]. The formula given has a '+1' term in idf so that the algorithm would not ignore terms that appear in all articles, this is the `sklearn` implementation and differs from the standard textbook formula which has the '+1' in the denominator in log2 of idf [11]. TF-IDF will be fitted on the original dataset of input articles (for `articles` in IDF term), then be used as vectorize to transform both the input and context articles. We will apply TF-IDF in two n-gram ranges of (1,1) and (1,2).

Word2Vec [12]: A word embedding architecture introduced by Google in 2013. It uses continuous bag-of-words (CBOW), which uses neighbouring words to predict target words, and continuous skip-gram which uses target words to predict the neighbouring words, using either hierarchical softmax or negative sampling [13] [14]. We will be using the gensim implementation of word2vec using the pretrained 'word2vec-google-news-300' model which is trained on the Google News dataset of about 100 billion words containing 3 million words and phrases in 300 dimensions. We chose this model due to the similarity between the domain of its dataset and out dataset being news. To calculate the article vector, we retrieve the vector of every single word in an article, existing those that do not exist in the embeddings, and then taking the average from the list [2].

Non-Latent Vectorizer: The non-latent vectorizer uses the non-latent feature selected in Section 3.3 and apply them on an article into a vector. This is the non-latent vector presentation of the respective article.

3.4.4 Similarity metric calculation

Past research by Alsuliman et al. in [2] proposes three different metrics to calculate the similarity between two documents: cosine distance, word appearance (word app), and matching score. In addition, we also propose a third metric being the harmonic mean of the three, to harmonise any statistical difference and incorporate all distributional differences between the measures.

Cosine distance: Calculated as one minus the cosine similarity of two vectors u and v . The cosine similarity is the cosine of the angle between the two vectors (calculated as the dot product of u and v) divided by the product of the Euclidean L2 norm of the two vectors to scale the range to [0, 1] [11]. Lower values denote higher similarity between the two vectors, and vice versa. The formula is given as follows [15]:

$$1 - \frac{u \cdot v}{\|u\| \|v\|}$$

Word app: Calculated as the number of unique common words between the prediction and the context articles divided by the number of unique words in the context article. Given that

$input_{unique}$ is the set of unique words in the input document, $context_{unique}$ is the set of unique words in the context document. The formula is as follows:

$$\frac{|input_{unique} \cap context_{unique}|}{|context_{unique}|}$$

Matching score: Calculated as the sum of the unique common words between the prediction and the context article vectorized, divided by the sum of the vectorized unique words in the context article. Given that $input_{unique}$ is the set of unique words in the input document, $context_{unique}$ is the set of unique words in the context document, $\text{vec}(\mathbf{x})$ is a function which vectorises the set of words x , and $\text{sum}(x) = \sum_{i=1}^n x_i$, the formula is as follows:

$$\frac{\text{sum}(\text{vec}(input_{unique} \cap context_{unique}))}{\text{sum}(\text{vec}(context_{unique}))}$$

Harmonic mean [16]: Calculated by the following formula :

$$H(x_1, x_2, \dots, x_n) = \frac{n}{\sum_{i=1}^n \frac{1}{x_i}}$$

When $n = 3$, $x_1 = c$ is the cosine distance, $x_2 = w$ is the word app, and $x_3 = m$ is the matching score, we have the following formula:

$$H = \frac{n}{\frac{1}{c} + \frac{1}{w} + \frac{1}{m}}$$

All these metrics are between the range of $[0, 1]$. Higher values for matching score and word app denote higher similarity, and vice versa. This is the opposite for cosine distance. Since we scrape up to three context articles per input article, we can apply *similarity metric smoothing* by calculating the similarity metric for the input article as the average of the similarity between the input article and each of the context article to reduce variance.

3.4.5 Similarity metric selection

Since we have different similarity metrics, we ought to compare them and select for the one that helps differentiate the **REAL** and the **FAKE** articles the best. We will use three methods to aid our selection: $\delta\mu$, Jensen-Shannon Divergence, and ANOVA.

$\delta\mu$, or the difference between the mean of **REAL** and **FAKE** articles is a very naive and simplistic measure to compare how differentiated the two distributions are. This measure does not remedy the behaviour of outliers which might significantly shift the mean of the distributions. It also ignores the variance of the distributions. However, it is still a numerically simple metric which would aid us when the distributions are well-formed.

Jensen-Shannon Divergence (JSD) [15] is based on the Kullback-Leibler Divergence (KL Divergence) to measure the similarity of two probability distributions. It is symmetric, positive, and in the range of $[0, 1]$ with values closer to 0 denoting more similar distributions and closer to 1 more dissimilar distributions. This is a value we hope to maximise. It is given by the formula:

$$\sqrt{\frac{D(p \parallel m) + D(q \parallel m)}{2}}$$

D is the KL Divergence and m is the “point-wise mean of p and q ” [15].

ANOVA (Analysis of Variance) is a statistical method to determine if there are significant differences between the means of groups [17]. We will be performing one-way repeated measure ANOVA which test for any significant difference between the means of the dependent variables (our non-latent features) among the groups defined by the independent variable (our label of **REAL** and **FAKE**) to reject the null hypothesis that the group means(**FAKE** and **REAL**) are equal. We reject this hypothesis when the p-value associated with the f-statistic of the respective feature is below the significant level $\alpha = 0.05$.

3.4.6 Comparison Results

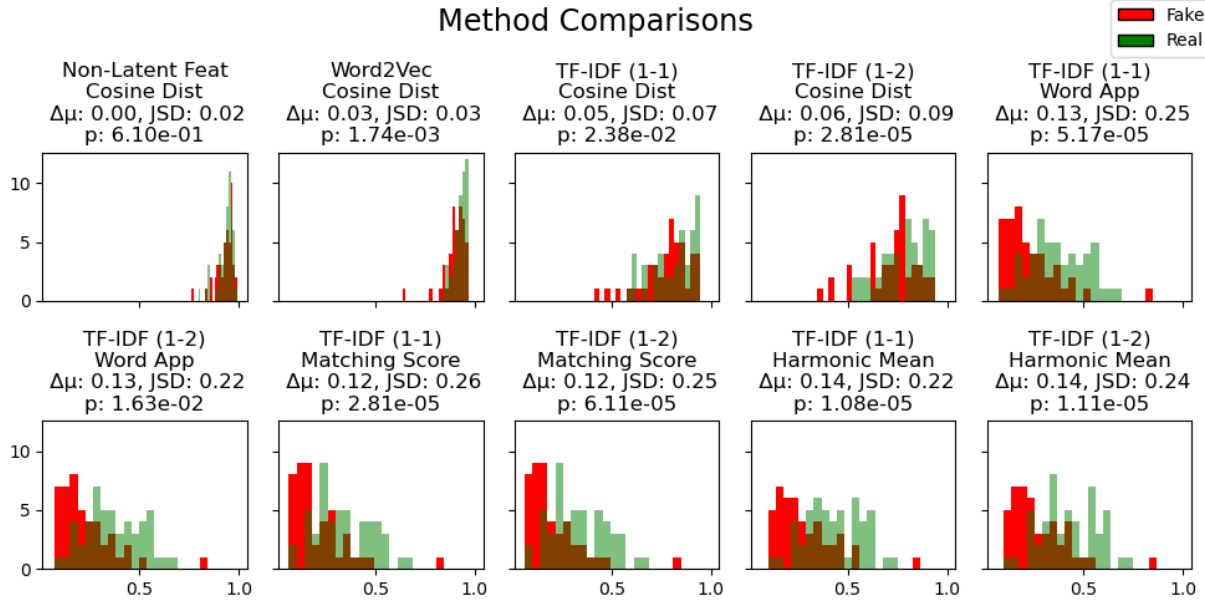


Figure 4: Similarity metrics comparison results. Each plot contains the name of the vectorizer, the similarity metric, and $\delta\mu$, JSD (Jensen-Shannon Distance), and p-value when ANOVA is performed with the metrics as dependent variables against the labels which are independent variables. The red distribution is the dependent variable when the label is label **FAKE**, and the green distribution is the dependent variable when the label is **REAL**.

Figure 4 shows our results with similarity metrics comparison. A good metric should have the fake (in red) and the real (in green) distributions be somewhat well separated. The brown area indicates the overlapping region. Non-latent Cosine Distance is the worst metric with $0\delta\mu$, $0.02JSD$, and $6.10e-01p$, where the two **FAKE** and **REAL** distributions are overlapping each other almost at the same point. Word2vec Cosine Distance and TF-IDF(1-1) also perform quite poorly with $\delta\mu$ of 0.03 and 0.05, JSD of 0.02 and 0.06, and p-value of $1.74e-03$ and $2.38e-02$ respectively. Both these plots have the **REAL** distribution shifted further to the right but the distributions still overlap significantly, unlike the Non-Latent Cosine Distance, however, these metrics fall below the typical significance level α of 0.05. Although TF-IDF (1-2) Cosine Distance has a similar $\delta\mu$ and JSD of 0.06 and 0.08 respectively, its p-value is markedly low at $2.81e-05$. Word App follows a similar pattern where TF-IDF (1-1) and (1-2) produce distinctly different results of $5.17e-05$ and $1.63e-02$ in p-value, but similar $\delta\mu$ of 0.13 and JSD of 0.22 and 0.23 respectively. We can conclude that the ngram-range of TF-IDF definitely contribute to marked differences in the metrics output. Matching score metrics have similar $\delta\mu$ of 0.12 and JSD range of $[0.23, 0.28]$ with low p-values of $2.81e-05$ and $6.11e-05$ respectively. The most significant features however are the harmonic mean, combining the previous metrics, with TF-IDF (1-1) at $\delta\mu = 0.14$, $JSD = 0.24$, $p = 1.08e-05$ and TF-IDF (1-2) at $\delta\mu = 0.14$, $JSD = 0.17$, $p = 1.11e-05$. Since TF-IDF (1-1) yields more difference between the distributions, we will use this as our definitive similarity metric.

3.5 Feature Normalisation

We experimented with different feature normalisation techniques across the entire dataset and for specific features. Normalisation was conducted columnwise for each feature so that each feature was normalised individually.

MinMaxScaler: This was used for features where there was a clear range of values in order to convert these values into the range of 0-1. This was not implemented frequently across our features as they did not have a fixed range of values making this technique ineffective.

StandardScaler: This was primarily used for linguistic features that did not have a clear range. This technique works by finding the Z-score for each value within the feature where $Z\text{-score} = \frac{X-\mu}{\sigma}$. This is done by first calculating the mean and standard deviation of the feature and then transforming each value into its Z-score.

RobustScaler: We experimented with this in place of StandardScaler for values prone to outliers. This was done as RobustScaler uses $IQR = Q3 - Q1$ instead of the standard deviation in the transformation calculation from StandardScaler. This is used because standard deviation is highly affected by outlier values and this was noticeable in features such as word count context articles.

Ultimately, after each scaling technique was applied and the classification was conducted, the results with each type of normalisation and various combinations of them underperformed compared to the original dataset without normalisation. There are a few different reasons why this was the case but after testing various classification models, the most probable reasons are the small size of the dataset used and the lack of appropriate scaling methods available for the linguistic features that rely on counting. Due to this, we decided not to use normalisation for our results.

3.6 Model — Machine learning

We used four state of the art machine learning models (commonly used in fake news detection) to perform our classification. We chose Logistic Regression (LR), Support Vector Machines (SVM), Decision Trees (DT), and XGBoost (XGB). Due to the small size of our dataset, we needed to tune the regularization hyperparameters to ensure our models didn't overfit. In particular, tree-based models such as DT and XGB should be able to fully segment our classes so we need to control the depth of the tree and splitting criteria. Models such as LR and SVM will need to control L2 regularization. In SVM, we will test the type of kernel used.

To find the best hyperparameters, we perform 5-Fold cross validation across 80% of our dataset and average the validation score. We pick the parameters with the best validation score and test our model with the remaining 20% of the dataset. The table in the Appendix G shows the hyperparameters tested for each model and a reason for why the ranges were selected.

3.7 Model — Neural networks

The second approach taken for classification was using deep learning as we hypothesised that it would be able to find more complex relationships between the large variety of BERT, linguistic, and similarity features. Initially both convolutional neural networks and simple feedforward layers were considered. After analysing the features and noting that they did not have any spacial relation, only the second approach was considered.

We used a very simple structure with 4 hidden layers using the ReLU activation function and an output layer with one neuron and the sigmoid activation function as the task required binary classification. We kept the structure simple due to the small size of the dataset and to prevent overfitting. Additionally, a dropout of 0.2 was added to further prevent overfitting. The dataset was split in a 60:20:20 train, validation, test split to allow for hyperparameter training and binary-crossentropy loss was used. Multiple optimisers were used during training, however, based on the validation accuracy we decided to use Adam as the optimiser.

118_Real	1_Fake
<i>FBI Completes Review of Newly Revealed Hillary Clinton Emails Finds No Evidence of Criminality</i>	<i>5 Million Uncounted Sanders Ballots Found On Clinton's Email Server</i>
FBI Director James Comey said Sunday that the bureau won't change the conclusion it made in July after it examined newly revealed emails related to the Hillary Clinton probe. "Based on our review, we have not changed our conclusions that we expressed in July with respect to Secretary Clinton" Comey wrote in a letter to 16 members of Congress. [...]	Hillary in hot water over her email server, again. Sacramento, CA — Democratic nominee Hillary Clinton is in hot water again after nearly 5 million uncounted California electronic ballots were found on her email server by the F.B.I. The majority of those ballots cast were by Bernie Sanders supporters. [...]

Table 3: A sample of one fake and real article in our dataset. The article ID, title and content are shown in the rows. Both articles are regarding a scandal with Hillary Clinton using a private server to store emails. The fake article reports on an event that never happens whereas the real article reports the true event – that Clinton was exonerated from criminality.

4 Experimental setup

4.1 Dataset

For our research, we use the **FakeNewsData** dataset collated by Horne and Adali in [4] on research regarding fake news in the 2016 presidential elections. This dataset contains two subsets, “*Buzzfeed Political News*”, and “*Random Political News*”. We make use of the *Buzzfeed* subset since this contains long form text articles that are binary categorized in with Fake and Real labels. The *Random* subset contains an extra label, Satire, which is out of scope for our research.

The original dataset was collated by Craig Silverman (BuzzFeed News Editor) in an article [18] analyzing fake news. The analysis concentrates on the Facebook engagement on real and fake news articles shared to the social media website. Various keywords related to events during the election were searched and articles with highest engagement were collected. A ground truth was assigned by manual analysis using a list of known fake and hyperpartisan news sites. A details description of their process can be found in their article.

Following BuzzFeed’s analysis, Horne extracted the content and title from the articles and formed the dataset. In total, there were 53 real and 48 fake articles. Notable events during the election such as Donald Trump’s campaign and various Hillary Clinton scandals and rumors were features in the articles.

After extending this dataset with our novel context article scraping and similarity methods, we used a 60/20/20 train/validation/test set. This was stratified and randomized to ensure the best results. An example of items in our dataset can be found in Table 3.

4.2 Data Analysis Using PCA and KMeans

In order to observe relationships between data within the entire dataset, the BERT, linguistic and similarity features for each data point were collected and dimensionality reduction was conducted using Principle Component Analysis to reduce the dimensions of the data to 2. Plot of these values were constructed with two different combinations of features: one using only BERT features for PCA and the other using linguistic and similarity features in addition to BERT features.

Additionally, to observe whether the data was forming clusters based on its reduced dimensions, KMeans was used with 5 clusters. The reason for using 5 clusters instead of 2 was because both fake and real articles about different topics or written differently can form clusters with other articles in a

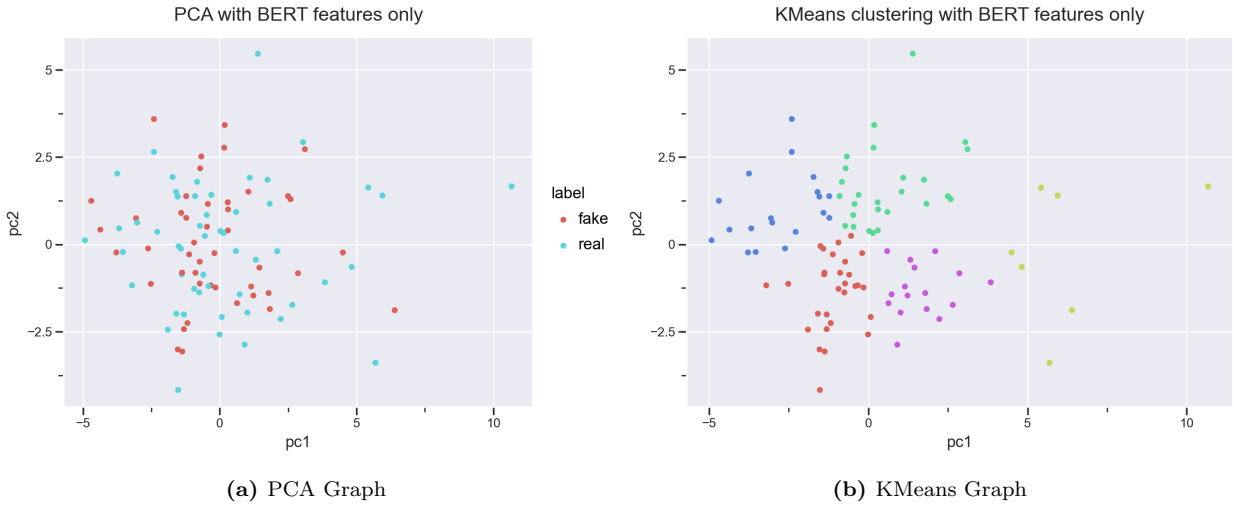


Figure 5: Analysis on PCA based on BERT features

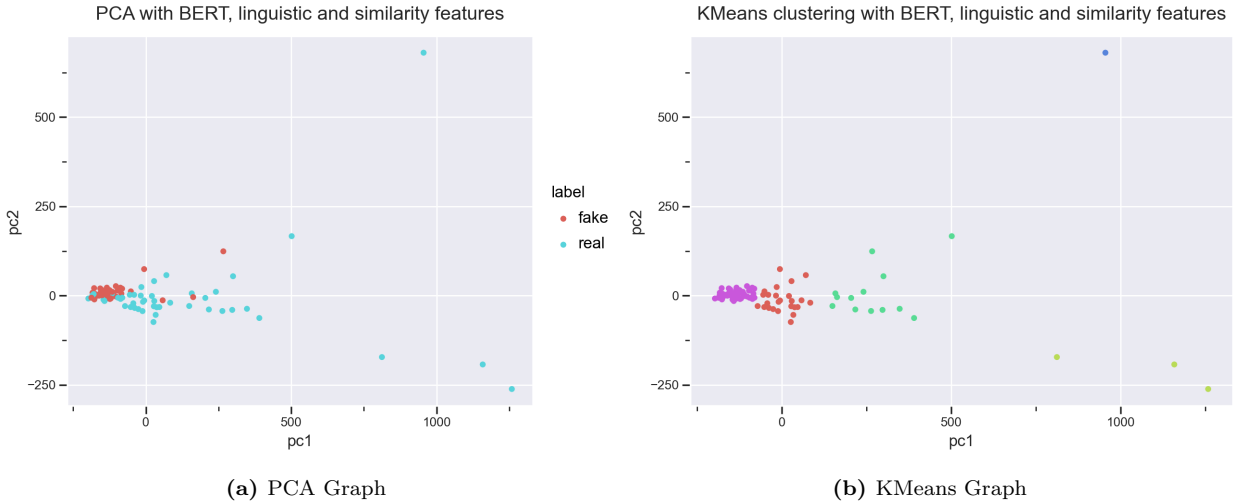


Figure 6: Analysis on PCA based on BERT, linguistic and similarity features

similar style and we were observing for distinct sets of real and fake clusters rather than simply one region of fake articles and another with real ones.

As can be seen in Figure 5a, there is very little relationship between the principle components of fake and real articles. When clustering is performed, as shown in Figure 5b, each cluster has an even distribution of real and fake articles implying there is no evident pattern in the data.

Whereas in Figure 6a, there is a clear grouping of values for fake data. Interestingly, real data does not have a clear cluster of values and instead occupies a large region of area to the right of the fake data. One potential reason for this is that there are a lot of similar features between fake articles such as the number of adverbs and the types of polarising language. Additionally, the similarity metric could be playing a strong role in influencing the first principle component, pc1, which would potentially result in fake news having low scores and real news having a broader range of similarity scores. Regardless, there is a clear distinction and this is also shown in Figure 6b where the majority of fake values are clustered in the purple cluster and the majority of real values are present in two separate clusters.

This indicates that the features selected are effective in separating the dataset into distinct fake and real clusters.

4.3 Evaluation metrics

To evaluate our classification models, we will use accuracy and F1 score. These metrics are commonly used for binary classification problems as well as in the misinformation detection domain. When referring to these metrics, we will label real articles as positive and fake articles as negative.

Accuracy is measured as the proportion of the total number of correctly classified samples over the total count of samples. Mathematically this is represented as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

Accuracy is an effective metric for this analysis as it is a holistic measure of the model's ability to identify a real article as real and a fake article as fake, which is the core component of the problem of fake news detection. Our dataset is quite balanced so this will be a good general first step measure.

Recall refers to how likely the model is able to identify an article is real when its ground truth is real. Precision is a measure of the proportion of all articles the model predicts to be real that have a ground truth of real. As such, recall is a measure of the quantity of the predictions whereas precision is a measure of the quality of the predictions. We are focusing on F1-score as it is the harmonic mean of recall and precision where

$$F_1 = 2 \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.$$

This allows for a fair assessment of both the quantity and quality of the predictions in a format that makes it easier to compare both models.

5 Results and discussion

For our analysis, we will be comparing the top 3 performing machine learning methods alongside the best performing neural network for each task. There are three models that are compared

1. **Model 1** is trained only on BERT features
2. **Model 2** is trained on both BERT and linguistic features
3. **Model 3** is trained on BERT, linguistic and similarity features

Figure 7 presents the classification results. As hypothesised from dataset analysis with PCA, Model 1 does not perform well. With a testing accuracy around 50% for each classifier, it is simply random guessing on the data. This is used as the baseline for our analysis as it does not incorporate any of our novel contribution to this knowledge domain.

Model 2 incorporates the most effective linguistic features in addition to the BERT features from Model 1. There is a significant increase in testing accuracy and F1-score across every model. Evident in the more complex models of neural networks and XGBoost, it is clear that there is a complex relationship being captured. There is an increase in test accuracy to around 85% and increase in F1-score to 0.87 for the top performing model. However, the decision tree classifier which is a far simpler model than XGBoost underperforms with this set of features.

In Model 3, similarity is added as a feature. The neural network accuracy increases as a result, going from 85% in Model 2 to 90% in Model 3. Additionally, F1-score increases from 0.86 to 0.91. Although SVC and XGBoost have similar accuracy to Model 2, their F1-score increases slightly. This is because there is a slight increase in precision and decrease in recall. This means that fewer articles that have a ground

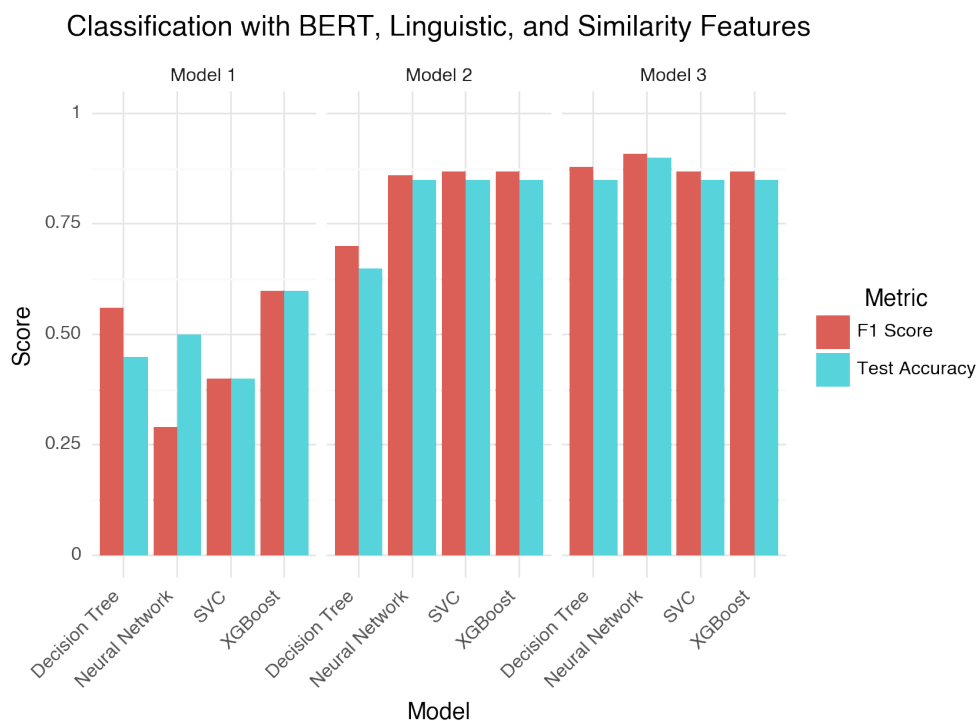


Figure 7: Accuracy and F1-score metrics for top performing models across different features

truth of real are classified as real but the quality of the real predictions increases. Another interesting note is that the simpler classifier - decision tree - performs much better when the similarity score is added. This indicates that the use of similarity score results in a split within the decision tree with much higher information gain than was possible with only BERT and linguistic features.

The reason these combinations of features were chosen for analysis was that BERT and linguistic features have a significant amount of prior research. These were included to represent the current state-of-the-art research within this section of long-form journalistic fake news detection. In terms of this, we conducted analysis and refined these features to develop a set of the most effective features. Model 3 adds similarity to this as this is a novel metric that we developed and analysed. As shown in the results, the linguistic features selected perform extremely well in this task and the additional similarity metric reduces the complexity of the classification whilst increasing the accuracy and F1-score for the neural network and decision tree classifiers.

Table 4 shows the results for the four machine learning classifiers used and adds more details than visible in Figure 7. It displays the training and testing metrics for each model to show the effect of changing the combination of features.

[Results and discussion] **Jim:** Add table of parameters of models

6 Conclusion

[Conclusion] Summarise the study and discuss directions for future improvement

Features	Model	Train Acc.	Train F1	Test Acc.	Test F1
BERT	LR	0.99	0.99	0.5	0.5
	SVC	1.0	1.0	0.4	0.4
	DT	0.99	0.99	0.45	0.56
	XGB	1.0	1.0	0.6	0.6
BERT + Linguistic	LR	0.94	0.94	0.70	0.73
	SVC	0.79	0.79	0.85	0.87
	DT	1.0	1.0	0.65	0.70
	XGB	1.0	1.0	0.85	0.87
BERT + Linguistic + Similarity	LR	0.94	0.94	0.7	0.73
	SVC	0.79	0.79	0.85	0.87
	DT	1.0	1.0	0.85	0.88
	XGB	1.0	1.0	0.85	0.87

Table 4: Table showing all training and test accuracies and F1 scores on our machine learning models with different feature inputs. The best test accuracy and F1 score have been emboldened.

6.1 Attributions

First and foremost, we thank Professor Yang Song and Maurice Pagnucco, and Wenbin Wang, for providing various insights and suggestions to our research.

We thank Horne and Adali et al. for providing the base dataset in their research [4] to which we provided extensions to. We also acknowledge all related works which we have used to experiment with our features.

Lastly the team has used open source software as part of developing our software. The licenses and links to the open source projects have been included in and **ATTRIBUTIONS** file which has been distributed alongside this report.

6.2 Limitations and Future Work

Preprocessing and tokenization

While our preprocessing was quite generic for NLP tasks, we believe there were a few oversights that caused unreliable results. Two issues were:

- Converting everything to lowercase destroyed acronyms such as “US”. This changed the meaning of some sentences.
- The `nltk` lemmatizer required manually specifying the part of speech to work. This caused some verbs to be incorrectly lemmatized. We could have used a different library such as `spacy` to extract the POS automatically. Alternatively we could have investigated not lemmatizing at all to maintain proper structure for non-latent features and BERT embeddings.

This research would have really benefited from less or no preprocessing at all. For all our features, there could be an argument where no preprocessing would have been better. For example summary extraction or BERT features could have learnt meaning from the unfiltered text. This could be investigated for future research to strengthen our features.

BERT Features

In this research, we have used the last layer hidden state of the CLS token (processed by Linear layer with Tanh activation) to encode the features as BERT features due to its common usage as a common feature for classification [19]. However, the HuggingFace transformer documentation has also mentioned that this output is not necessarily a 'good summary of the semantic content of the input', and suggests applying pooling or averaging the sequence of hidden-states for the entire input [19]. In addition, we can also use the fully connected layers on top of the frozen BERT, which is the typical approach for textual classification. These are things that we could experiment with to improve our BERT features input.

Moreover, our research assumes the hypothesis of the 'fakenews'-ness visibility within 512 tokens of an article. However, we could conduct our experiments with LongFormer to test for our hypothesis and for any further improvement.

Non-Latent Features

Our non-latent feature extraction pipeline currently applies on the content of the article. However, we could easily use the same procedure to detect for any features which would significantly differentiate between the **REAL** and the **FAKE** distributions in the title of the articles. Past research has also applied feature extraction method on the title of articles, and our model would benefit from any important features.

Summary extraction

Our summary extraction extracted most of the correct meaning from the text. However two main issues remained causing it to produce imperfect results:

- Words would be unordered and if read together in a sentence, wouldn't make sense to a human.
- *Junk* such as connectives from poor preprocessing or words not too related to the article content would be left in the summary. This could be a sign of a ineffective topic extraction method.

We believe future research could have looked into better tuned models that synthesized higher quality topic sentences. From a cursory search for "*keyword extractor*" on HuggingFace, we found various pre-trained models using modern transformer methods such as BERT to output keywords. We believe that our method still worked well enough to demonstrate that this could be done but better summaries would have yielded higher quality contextual articles. In addition, we could have also included the title since semantically, the title is supposed to tell the reader what the rest of the article is about.

Article scraping

We had intended our pipeline to be fully automatic, using an API to scrape for articles based on the keywords. For the scope of this project, we settled on manual scraping to show that using contextual articles would improve results. Unfortunately, after being passed to Google, some input summaries would return no or limited results and required human intervention to produce any articles. This was a particular problem.

For Real labelled articles, we believe an improvement on the summary extractor reducing the amount of *junk* returned would have improved results. However, we hypothesize that if an outrageous Fake article as introduced, we may very well find no results about the event online. One example is the following article about Trump "*snorting cocaine*":

10Fake: The Internet is buzzing today after white supremacist presidential candidate Donald Trump was caught by hotel staff snorting cocaine.

Maria Gonzalez an employee at the Folks INN & Suites Hotel in Phoenix brought room service to his room witnessed it all.

“When I walked in I saw 3 naked prostitutes and maybe 100,000 in hundred dollars bills and a mountain of white powder on the table, I thought it was a dog on the floor sleep but it was his hair piece, he was bald and sweating like crazy.” [...]

This event never happened and consequently, we could not find any contextual articles about it. For our research, we skipped articles with no context. We believe one way to resolve this is to include the article with no similarity score or a low one. More research needs to be done into whether mixed articles with and without a similarity can perform well together especially considering in the real world, this is definatly an issue.

Article vectorisation

Similar to 6.2, for future work, we can construct a non-latent vectoriser from the title of the articles, and perform ANOVA on this vectoriser applied on the input article against the context articles. It is likely however that the non-latent 'title' vectoriser might yield a p-value higher than the significance threshold, since this is the case with the non-latent 'body' vectoriser.

Similarity metric calculation

For our similarity metric measure, we could also build a simple classifier which takes an input article and context article, and output a similarity measure, and maximising the output if the input article is **REAL** and minimising the output if it is **FAKE**.

In addition, we also have a hypothesis that **REAL** articles have a higher level of semantic similarity between the content and the title. We can vectorise the pair, and then apply similarity metric measures on them and repeat the same methodology as outlined in Section 3.4.3, 3.4.4, 3.4.5.

Dataset

For the scope of our research, we used a fairly small dataset to present our contributions to contextual article scraping. However, this dataset only concentrated on the political events surrounding the 2016 United States election. This causes two main problems for our model:

- We are prone to overfit our models since such a small dataset will be easily segmented by most state of the art methods.
- Our model will learn specifics in the US election which we do not want to learn. This could cause the models to be confused if we try to classify more recent news.

In the future research could be done to provide an automated API for scraping which would have allowed for a much larger dataset to be used that covers multiple world events across different years. Along with this, masking out event specific words could have been done to reduce learning of event specifics.

References

- [1] Wissam Antoun et al. “State of the art models for fake news detection tasks”. In: *2020 IEEE international conference on informatics, IoT, and enabling technologies (ICIOT)*. IEEE. 2020, pp. 519–524.
- [2] Fahad Alsuliman et al. “Social Media vs. News Platforms: A Cross-analysis for Fake News Detection Using Web Scraping and NLP”. In: *Proceedings of the 15th International Conference on Pervasive Technologies Related to Assistive Environments*. 2022, pp. 190–196.
- [3] Sairamvinay Vijayaraghavan et al. “Fake news detection with different models”. In: *arXiv preprint arXiv:2003.04978* (2020).
- [4] Benjamin Horne and Sibel Adali. “This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news”. In: *Proceedings of the international AAAI conference on web and social media*. Vol. 11. 1. 2017, pp. 759–766.
- [5] Xinyi Zhou and Reza Zafarani. “A survey of fake news: Fundamental theories, detection methods, and opportunities”. In: *ACM Computing Surveys (CSUR)* 53.5 (2020), pp. 1–40.
- [6] Sonal Garg and Dilip Kumar Sharma. “Linguistic features based framework for automatic fake news detection”. In: *Computers & Industrial Engineering* 172 (2022), p. 108432.
- [7] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. ”O’Reilly Media, Inc.”, 2009.
- [8] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *CoRR* abs/1810.04805 (2018). arXiv: [1810.04805](https://arxiv.org/abs/1810.04805). URL: <http://arxiv.org/abs/1810.04805>.
- [9] Radim Řehůřek and Petr Sojka. “Software Framework for Topic Modelling with Large Corpora”. English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50.
- [10] Lucas Ou-Yang. *newspaper3k*. 2013. URL: <https://newspaper.readthedocs.io/en/latest/>.
- [11] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [12] Google Code Archive. *word2vec*. 2013. URL: <https://code.google.com/archive/p/word2vec/>.
- [13] Radim Řehůřek and Petr Sojka. “Software Framework for Topic Modelling with Large Corpora”. English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. <http://is.muni.cz/publication/884893/en>. Valletta, Malta: ELRA, May 2010, pp. 45–50.
- [14] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: [1301.3781](https://arxiv.org/abs/1301.3781) [cs.CL].
- [15] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: [10.1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2).
- [16] *Harmonic Mean*. May 2023. URL: https://en.wikipedia.org/wiki/Harmonic_mean.
- [17] Gurchtan Singh. *ANOVA: Complete guide to Statistical Analysis & Applications*. July 2023. URL: <https://www.analyticsvidhya.com/blog/2018/01/anova-analysis-of-variance/>.
- [18] Craig Silverman. *This Analysis Shows How Viral Fake Election News Stories Outperformed Real News On Facebook*. Nov. 2016. URL: <https://www.buzzfeednews.com/article/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook>.
- [19] Thomas Wolf et al. “Transformers: State-of-the-Art Natural Language Processing”. In: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: <https://www.aclweb.org/anthology/2020.emnlp-demos.6>.

- [20] *Jensen–Shannon divergence*. July 2023. URL: https://en.wikipedia.org/wiki/Jensen%E2%80%93Shannon_divergence.

A Individual contributions

Member	Major contributions to project and report
Jim	Preprocessing & tokenization
	Similarity model article scraping
	Machine learning models
Dhruv	
Duke	article vectorisation
	similarity metric calculation & selection
	similarity comparison result
	non-latent features
	EDA

A.1 Jim

Jim contributed to the project particularly in the sections of preprocessing & tokenization, similarity model (Article Scraping), and the machine learning models. At the start of the project Jim contributed to the manual scraping of articles. Towards the end of the project, Jim was also responsible for writing code to combine all the features together and integrating all work completed by the other members so that models can be run using common bootstrap code.

In the project proposal and literature review, Jim contributed equally with the team and worked on each other's sections. In the project demo, Jim contributed less than average due to the fact that a live online demo was cut from the final version due to time and validity reasons. It was decided that since the rest of the team already started their assigned sections, Jim would start on the report and attempt to contribute more than average.

In the report Jim contributed to the Sections 3.1, 3.4.1, 3.4.2, 3.6, 4.1, 4.3, the machine learning part in results and related limitations.

A.2 Dhruv

[Individual contributions] **Dhruv:** ~1pg detailing individual contributions

A.3 Duke

Duke contributed to the project particularly in the sections of non-latent features, and similarity model (article vectorisation, similarity metric calculation, similarity metric selection, comparison results) and the EDA (exploratory data analysis) notebook, and the associated code and plots. At the start of the project, Duke contributed to the similarity model (article vectorisation, similarity metric calculation). Towards the end of the project, Duke contributed to the similarity model (similarity metric selection, comparison results), and the methodology and results for non-latent feature extraction and analysis, as well as the EDA.

In the project proposal, Duke contributed equally with the team. In the literature review, Duke reviewed the literature on datasets, features, evaluation, and model performance in fake news detection. In the project demo, Duke contributed mainly to the features section (BERT, Non-latent, Similarity metrics). In the report, Duke contributed to Sections 3.2, 3.3, 3.4.3, 3.4.4, 3.4.5, 3.4.6, and BERT, non-latent features and similarity metrics section in results and limitations, and Appendices C, D, E, F.

B Article scraping

ID	Article extract	Summary
128_Real	<p>[...]I have a prediction. I know exactly what November 9 will bring. Another day of God’s perfect sovereignty. He will still be in charge. His throne will still be occupied. He will still manage the affairs of the world. Never before has His providence depended on a king, president, or ruler. And it won’t on November 9, 2016. “The LORD can control a king’s mind as he controls a river; he can direct it as he pleases” (Proverbs 21:1 NCV). On one occasion the Lord turned the heart of the King of Assyria so that he aided them in the construction of the Temple. On another occasion, he stirred the heart of Cyrus to release the Jews to return to Jerusalem. [...]</p>	god wa one never every king novem-ber still heart
2_Fake	<p>Washington, D.C. – South African Billionaire, Femi Adenugame, has released a statement offering to help African-Americans leave the United States if Donald Trump is elected president. According to reports, he is offering \$1 Million, a home and car to every Black family who wants to come to South Africa. Concerns about Donald Trump becoming president has prompted a South African billionaire to invest his fortune in helping African-Americans leave the United States to avoid further discrimination and inequality. [...]</p>	ha adenugame africanamericans south femi united states africa presi-dent donald
10_Fake	<p>The Internet is buzzing today after white supremacist presidential candidate Donald Trump was caught by hotel staff snorting cocaine. Maria Gonzalez an employee at the Folks INN & Suites Hotel in Phoenix brought room service to his room witnessed it all. “When I walked in I saw 3 naked prostitutes and maybe 100,000 in hundred dollars bills and a mountain of white powder on the table, I thought it was a dog on the floor sleep but it was his hair piece, he was bald and sweating like crazy.” [...]</p>	wa room hotel maria told em-ployee gonzalez hit video get
34_Fake	<p>It has been more than fifteen years since Rage Against The Machine have released new music. The members of the band have involved themselves in various other projects during their lengthy hiatus, but one pressing issue has forced the band to team up once again. In a statement posted online, Rage Against The Machine announced they would be releasing a brand new album aimed at spreading awareness about “how awful Donald Trump is”. [...]</p>	trump rage album machine band ha donald music out-side year

Table 5: Articles we were not able to find context articles for. Articles like 10_Fake describe a situation that plainly does not happen whereas articles like 128_Real and 34_Fake produces summaries that confuse Google News and don’t produce grate articles.

C Word Count Histogram

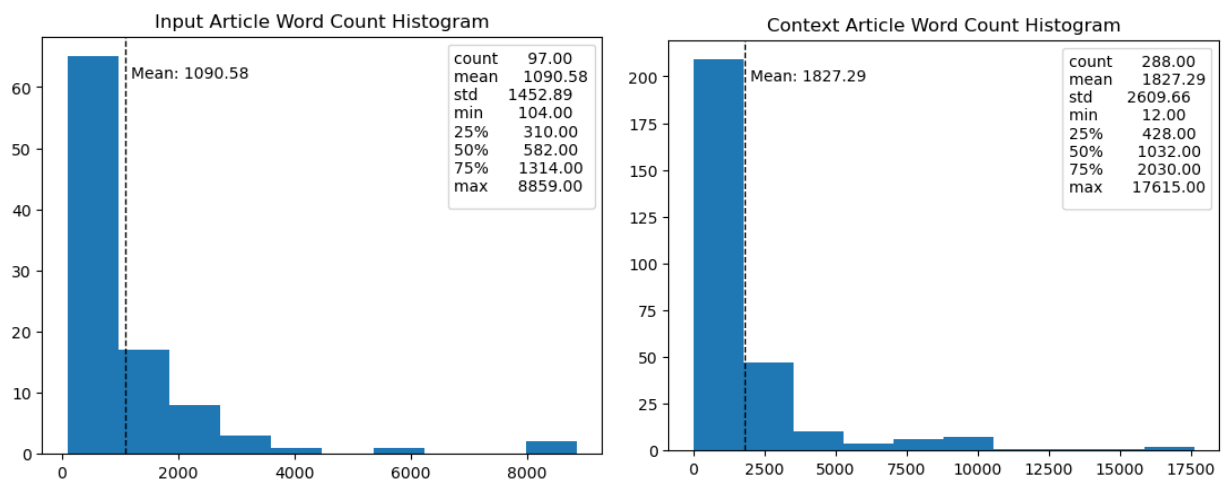


Figure 8: Word count histogram of input (left) and context articles (right)

D Non-Latent Features

Table 6: Non-latent feature names (far-right column) and their corresponding category, feature type, and calculation method

Category	Feature Type	Calculation	Feature Name
Diversity	Noun (Unique)	Count	div_NOUN_sum
	Verb (Unique)		div_VERB_sum
	Adjective (Unique)		div_ADJ_sum
	Adverb (Unique)		div_ADV_sum
	Lexical Word (Unique)		div_LEX_sum
	Content Word (Unique)		div_CONT_sum
	Function Word (Unique)		div_FUNC_sum
	Noun (Unique)	Percent	div_NOUN_percent
	Verb (Unique)		div_VERB_percent
	Adjective (Unique)		div_ADJ_percent
	Adverb (Unique)		div_ADV_percent
	Lexical Word (Unique)		div_LEX_percent
	Content Word (Unique)		div_CONT_percent
	Function Word (Unique)		div_FUNC_percent
Quantity	Noun	Count	div_NOUN_sum
	Verb		div_VERB_sum
	Adjective		div_ADJ_sum
	Adverb		div_ADV_sum
	Pronoun		div_PRON_sum
	Personal Pronoun		div_DET_sum
	Possessive Pronoun		div_NUM_sum
	Determinant		div_PUNCT_sum
	Number		div_SYM_sum
	Punctuation		div_PRP_sum
	Symbol		div_PRP\$_sum
	Wh-Determinant		div_WDT_sum
	Cardinal Number		div_CD_sum
	Verb (Past Tense)		div_VBD_sum
	Stop Word		div_STOP_sum
	Lowercase Word		div_LOW_sum
	Uppercase Word		div_UP_sum
	Negation		div_NEG_sum
	Noun	Percent	div_NOUN_percent
	Verb		div_VERB_percent
	Adjective		div_ADJ_percent
	Adverb		div_ADV_percent
	Pronoun		div_PRON_percent
	Personal Pronoun		div_DET_percent
	Possessive Pronoun		div_NUM_percent
	Determinant		div_PUNCT_percent
	Number		div_SYM_percent
	Punctuation		div_PRP_percent
	Symbol		div_PRP\$_percent
	Wh-Determinant		div_WDT_percent
	Cardinal Number		div_CD_percent

	Verb (Past Tense)	Count	div_VBD_percent
	Stop Word		div_STOP_percent
	Lowercase Word		div_LOW_percent
	Uppercase Word		div_UP_percent
	Negation		div_NEG_percent
	Quote		div_QUOTE_sum
	Noun Phrase		div_NP_sum
	Character		div_CHAR_sum
	Word		div_WORD_sum
	Sentence		div_SENT_sum
	Syllable		div_SYLL_sum
Sentiment	Exclamation Mark	Count	div_!_sum
	Question Mark		div._?_sum
	All-Cap Word		div_CAPS_sum
	Polarity	Index	div_POL_sum
	Subjectivity		div_SUBJ_sum
Pronoun	First Person Singular	Count	div_FPS_sum
	First Person Plural		div_FPP_sum
	Second / Third Person		div_STP_sum
	First Person Singular	Percent	div_FPS_percent
	First Person Plural		div_FPP_percent
	Second / Third Person		div_STP_percent
Average	Character Per Word	Average	div_chars_per_word_sum
	Word Per Sentence		div_words_per_sent_sum
	Clause Per Sentence		div_claus_per_sent_sum
	Punctuation Per Sentence		div_puncts_per_sent_sum
Syntax Tree Depth	Median Syntax Tree Depth	Median	div_ALL_sum
	Median Noun Phrase Syntax Tree Depth		div_NP_sum
Readability	Gunning-Fog Index	Index	div_gunning-fog_sum
	Coleman-Liau Index		div_coleman-liau_sum
	Flesch Kincaid Grade Level		div_dale-chall_sum
	Linsear Write		div_flesch-kincaid_sum
	SPACHE		div_linsear-write_sum
	Dale Chall Readability		div_spache_sum
	Automated Readability Index (ARI)		div_automatic_sum
	Flesch Reading Ease		div_flesch_sum

E Non-latent Feature Pearson Correlation Matrix

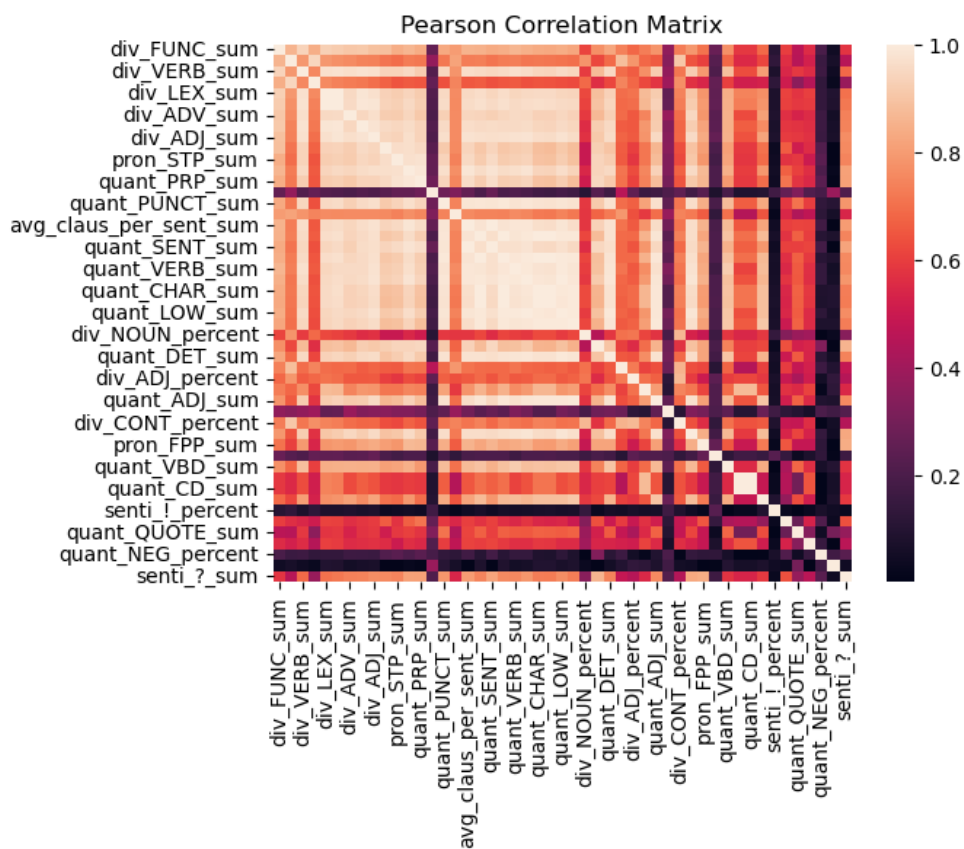


Figure 9: Pearson correlation matrix of selected non-latent features (which are red and white in Figure 10). Brighter areas show higher rate of correlation, and vice-versa. An explanation of features is available in Appendix D

F ANOVA of Non-Latent Feature against Labels

< 0.05
 >= 0.05
 Correlated

Features sorted by p-values

Feature	p-value	Feature	p-value
div_FUNC_sum	4.55e-09	div_FUNC_sum	4.55e-09
div_LEX_percent	1.08e-07	div_LEX_percent	1.08e-07
div_VERB_sum	4.64e-07	div_VERB_sum	4.64e-07
div_FUNC_percent	4.7e-07	div_FUNC_percent	4.7e-07
div_LEX_sum	5.01e-07	div_LEX_sum	5.01e-07
div_CONT_sum	9.85e-07	div_CONT_sum	9.85e-07
div_ADV_sum	1.28e-06	div_ADV_sum	1.28e-06
div_NOUN_sum	1.77e-06	div_NOUN_sum	1.77e-06
div_ADJ_sum	2.75e-06	div_ADJ_sum	2.75e-06
quant_PRPs_sum	5.28e-06	quant_PRPs_sum	5.28e-06
pron_STP_sum	1.34e-05	pron_STP_sum	1.34e-05
quant_PRON_sum	1.49e-05	quant_PRON_sum	1.49e-05
quant_PRP_sum	1.96e-05	quant_PRP_sum	1.96e-05
quant_PUNCT_percent	2.30e-05	quant_PUNCT_percent	2.30e-05
quant_PUNCT_sum	2.43e-05	quant_PUNCT_sum	2.43e-05
div_ADV_percent	2.71e-05	div_ADV_percent	2.71e-05
avg_claus_per_sent_sum	2.75e-05	avg_claus_per_sent_sum	2.75e-05
quant_NP_sum	2.96e-05	quant_NP_sum	2.96e-05
quant_SENT_sum	3.52e-05	quant_SENT_sum	3.52e-05
quant_WORD_sum	3.56e-05	quant_WORD_sum	3.56e-05
quant_VERB_sum	3.64e-05	quant_VERB_sum	3.64e-05
quant_STOP_sum	4.32e-05	quant_STOP_sum	4.32e-05
quant_CHAR_sum	4.77e-05	quant_CHAR_sum	4.77e-05
quant_SYLL_sum	5.46e-05	quant_SYLL_sum	5.46e-05
quant_LOW_sum	5.73e-05	quant_LOW_sum	5.73e-05
quant_NOUN_sum	7.09e-05	quant_NOUN_sum	7.09e-05
div_NOUN_percent	7.16e-05	div_NOUN_percent	7.16e-05
quant_NEG_sum	9.91e-05	quant_NEG_sum	9.91e-05
quant_DET_sum	1.02e-04	quant_DET_sum	1.02e-04
div_VERB_percent	1.09e-04	div_VERB_percent	1.09e-04
div_ADJ_percent	1.1e-04	div_ADJ_percent	1.1e-04
quant_UP_sum	2.31e-04	quant_UP_sum	2.31e-04
quant_ADJ_sum	2.31e-04	quant_ADJ_sum	2.31e-04
quant_UP_percent	2.38e-04	quant_UP_percent	2.38e-04
div_CONT_percent	2.66e-04	div_CONT_percent	2.66e-04
quant_ADV_sum	3.83e-04	quant_ADV_sum	3.83e-04
pron_FPP_sum	7.2e-04	pron_FPP_sum	7.2e-04
quant_VBD_percent	1.46e-03	quant_VBD_percent	1.46e-03
quant_VBD_sum	1.79e-03	quant_VBD_sum	1.79e-03
quant_NUM_sum	2.27e-03	quant_NUM_sum	2.27e-03
quant_CD_sum	2.27e-03	quant_CD_sum	2.27e-03
quant_WDT_sum	2.34e-03	quant_WDT_sum	2.34e-03

Figure 10: Table of features followed by their p-value when ANOVA is performed with features as dependent variables against labels as the independent variable. Features are sorted by their p-value. The table on the right continues from the last row of the left table. The significance level is $\alpha = 0.05$, features with p-value above the threshold is in blue. Features below the level are subjected to correlation clustering algorithm which keeps features with lowest p-value and remove all their correlated features as outlined in Section 3.3, selected features are in white, whilst removed features are in red.

G Machine learning

Model	Parameter	Selection	Reasoning
Logistic Regression	Inverse L2 coefficient	0.2:1.2:0.2	This is the main regularization parameter. We chose a range around the default 1.0 but shifted our range to be more biased towards higher regularization.
	Solver	lbfgs, liblinear	The liblinear solver was suggested by the documentation as an alternative for small datasets.
SVM	Inverse L2 coefficient	0.2:1.2:0.2	Same reasoning as LR regularization.
	Kernel	rbf, poly, sigmoid	Selecting the right kernel for a dataset will make our methods perform better.
	Kernel coefficient	$\frac{1}{n_features \times var(X)}$, 0.01, 0.05	Same reason as above.
Decision Tree	Criterion	gini, entropy	To test different methods of measuring split quality on the node.
	Max depth	no limit, 3:9:2	Controls how complex the tree is. A less deeper tree is more regularized.
	Max features	$0.3 \times n_features$, $\sqrt{n_features}$, all features	Standard defaults suggested by documentation. Controls regularization so not all features are considered at each split.
	Min samples for splitting node	2:4:1	Reduce the number of leafs with only one sample of representation to increase regularization.
XGBoost	Learning rate	0.1:0.5:0.1	Smaller learning rates reduce overfitting.
	Max depth	1:6:1	Same as max depth for DTs.
	L2 coefficient	0.8:1.6:0.2	Testing higher regularization. Default is 1.
	L1 coefficient	0:0.4:0.2	Testing higher regularization. Default is 0.0.

Table 7: Table of all the models chosen and the hyperparameters selected for each model. We describe a range of values in the format start:end:step, where start and end are inclusive. Our main focus was to investigate regularization parameters that would better fit our smaller dataset.