# Introduction:

Often, news articles are given misleading headlines in order to attract attention and readers. This report explores the use of Natural Language Processing techniques, such as Term-frequency Inverse-document-frequency and the BERT transformer, in order to extract sentence sentiments from news articles, and aims to classify the body of an article in relation to it's headline using different learning models.

### Problem Definition:

This report aims to apply Tf-Idf and BERT, to produce features representative of the documents in the Fake News Competition corpus. Using the features, a learning model is used to classify documents, formatted into (Headline, articleBody) pairs, as related or unrelated. Following, the documents classed as related are fed into a separate learning model to classify whether the sentiments of the articleBody agrees, disagrees, or discusses the sentiment of the Headline.

# Proposed Solutions:

### 1. Feature Extraction

For feature extraction, two methods are used:

a. Tf-Idf:
   Tf-Idf is a metric to measure the relevancy of a word in the context of a document, in a corpus. It is calculated as follows:

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \qquad \mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \qquad \mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D)$$

   where *t, d, f, N* and *D* are term, document, frequency, number of documents in the corpus, and the corpus respectively. It lends itself well as a sentence vector. However, it is flawed in that it is unable to differentiate homonyms.

b. BERT:
   BERT is a transformer that can be used to embed tokens/words into vectors, taking into account the context of a given 'sentence' input. A word is converted into a token, and passed through 12 transformer blocks, each generating a 768-sized embedding. BERT also uses a set of special tokens, such as '[CLS]', a special token that can be used in classification tasks, and '[SEP]', a token signifying the separation of two sentences in the input. BERT is typically configured for a maximum of two sentences, with a combined length of 512 tokens.

To compare, Tf-Idf associates the same value to a given word, regardless of the context it is used in within the document. However, BERT embeds every word differently with regards to context. Tf-Idf somewhat alleviates this by using n-grams, however this greatly increases the vocabulary/feature size.

On the other hand, BERT is limited to a sequence length of 512, hence it is difficult to encode an entire document. This is not an issue for Tf-Idf.

With regards to speed, it is faster to generate Tf-Idf features, in comparison to BERT. However, more pre-processing is required for Tf-Idf, such as stopword removal and stemming.

**2. Two-Step Classification**

**a. Related/Unrelated Classification**

With TfIdf, for a given headline-body pair, both are vectorized separately, and cosine-similarity is used to find the distance between the two feature vectors. This achieves ~90% accuracy across different machine learning models, such as the Random Forest, SVM, and Logistic Regression classifiers.

With a pretrained BERT, for a given headline-body pair, multiple methods were tested:

i. Passing a headline/body pair as input to the tokenizer to generate token sequences

ii. Passing the headline and body separately, and:
    1. Comparing the features with cosine similarity
    2. Concatenating the features

The embeddings tested are:

1. Using the '[CLS]' embedding as a sentence embedding
2. Using the pooled output as a sentence embedding
3. Concatenating the last 4 layers of the transformer, and using the '[CLS]' embedding

To compare machine learning models to deep learning models, a simple 2-layer classifier is applied to concatenated Tf-Idf features, as shown in fig.1. In contrast, the BERT features are tested with:

1. A Bidirectional-LSTM (fig.2)
2. A Softmax-activated classifier layer on the '[CLS]' features (fig.3)

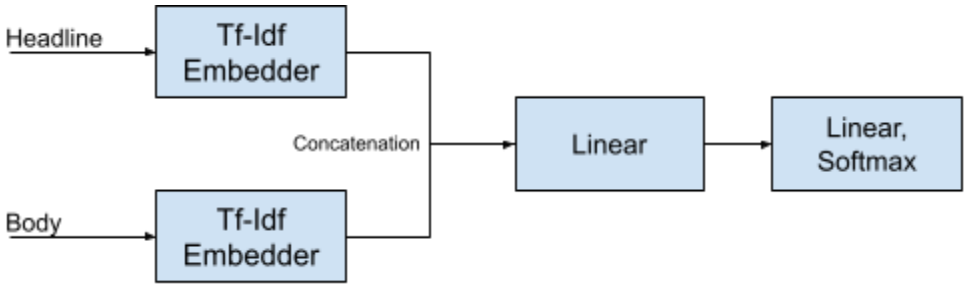


Fig. 1: Tf-Idf-based Classifier Architecture

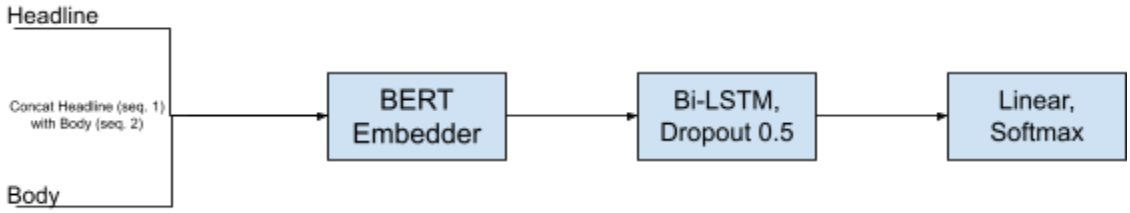


Fig. 2: BERT-based Bi-LSTM Architecture

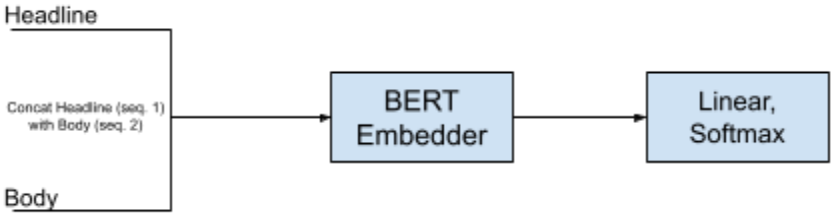


Fig. 3: Basic BERT Classifier Architecture

The softmax activations reduce the hidden features to two probabilities corresponding to the 'unrelated' and 'related' classes. Dropout layers are used throughout all networks to combat overfitting.

To reduce training time, the Tf-Idf features are limited to unigrams, and the BERT embedder is frozen. In all deep-learning configurations, a validation dataset is used to track progress.

**b. Agree/Disagree/Discuss Classification**

The model in Fig.2 is re-used, with the softmax activation layer outputting 3 values corresponding to the probability the body and headline agree/disagree/discuss.

Based on the output of (a), the input sequences may be passed to (b) if they are related, hence a stance is predicted.

## Analysis of Results:

Looking at Table.1, when using Tf-Idf features, finding the cosine similarities and euclidean distances between the headline/body features resulted in the greatest accuracy, using an SVM/logistic regression provided marginally better accuracy. SVMs also performed better with the other input features. Overall, using the cosine similarities and euclidean distances as features trained the fastest, taking ~0.5 hours, whereas concatenating the Tf-Idf vectors resulted in a 4-5 hour training time. Combining the texts took around 2 hours.

Table.1 also shows the accuracies of the BERT classifiers. It is clear to see that pairing the headlines and bodies prior to tokenizing the text resulted in the best results. There is little variance in performance between different machine learning models, but the random forest classifier showed better performance overall in the paired configuration. In terms of the type of

feature used from the BERT output, CLS had the best results throughout all machine learning models.

| | Tf-Idf | | | BERT | | |
|---|---|---|---|---|---|---|
| | Cos Sim + Euclidean Distance | Concat Vectors | Combine Texts | Pair + CLS | Pair + Pooled | Pair + Concat, CLS |
| Random Forest | 0.95 | 0.72 | 0.70 | 0.98 | 0.98 | 0.98 |
| SVM | 0.97 | 0.73 | 0.72 | 0.98 | 0.97 | 0.98 |
| Logistic Regression | 0.97 | 0.39 | 0.32 | 0.98 | 0.97 | 0.97 |

| | BERT | | | | | |
|---|---|---|---|---|---|---|
| | Split + CLS + Cos Sim | Split + Pooled + Cos Sim | Split + Concat, Mean + Cos sim | Split + CLS | Split + Pooled | Split + Concat, Mean |
| Random Forest | 0.63 | 0.61 | 0.63 | 0.73 | 0.70 | 0.74 |
| SVM | 0.73 | 0.72 | 0.74 | 0.75 | 0.72 | 0.73 |
| Logistic Regression | 0.61 | 0.46 | 0.48 | 0.54 | 0.57 | 0.53 |

Table 1: Table of results, showing prediction accuracy for different configurations and machine models

| Classification | Tf-Idf + Similarity + SVM | | | Paired BERT + CLS tag + Random Forest | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | F1-Score | Precision | Recall | F1-Score |
| Unrelated | 0.97 | 0.98 | 0.98 | 0.98 | 0.99 | 0.98 |
| Related | 0.95 | 0.92 | 0.94 | 0.96 | 0.96 | 0.96 |

Table 2: Table of results, showing precision, recall and f1-score for the best Tf-Idf and BERT configurations

Table.2 shows that overall, BERT outperforms Tf-Idf marginally in all metrics, for both unrelated and related classification.

| | Tf-Idf Deep Classifier | | | | BERT-LSTM Classifier | | | | BERT Deep Classifier | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Prec. | Recall | F1 | Acc. | Prec. | Recall | F1 | Acc. | Prec. | Recall | F1 | Acc. |
| Unrelated | 0.97 | 0.37 | 0.54 | 0.54 | 0.99 | 0.99 | 0.99 | 0.98 | 0.99 | 0.98 | 0.98 | 0.98 |
| Related | 0.37 | 0.97 | 0.54 | | 0.97 | 0.98 | 0.97 | | 0.95 | 0.97 | 0.96 | |

Table 3: Table of results, showing precision, recall, f1-score and accuracy for 3 deep learning classification models

Table.3 shows even better performance with BERT, especially when combined with an LSTM. In contrast, the Tf-Idf based deep classifier struggles significantly, performing worse than all other models besides the concatenated and text-combined linear regression Tf-Idf models.

| | BERT-LSTM Stance Classifier | | | | Combined Relation/Stance Model | | | |
|---|---|---|---|---|---|---|---|---|
| | Prec. | Recall | F1 | Acc. | Prec. | Recall | F1 | Acc. |
| Agree | 0.54 | 0.63 | 0.58 | 0.7 | 0.58 | 0.57 | 0.58 | 0.91 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Disagree | 0.47 | 0.08 | 0.14 | | 0.41 | 0.10 | 0.16 | |
| Discuss | 0.79 | 0.83 | 0.81 | | 0.74 | 0.85 | 0.79 | |
| Unrelated | - | - | - | - | 0.99 | 0.98 | 0.99 | |

Table 4: Table of results, showing precision, recall, f1-score and accuracy for the stance classification models

Adapting the BERT-LSTM model for 3-class classification, performance falls. It is also made clear that the model struggles to classify the disagree stance-class.

## Discussion:

The results show that BERT-based solutions produced the best results, especially when tokenizing the headline/body sequences together instead of separately, as shown in table.1, although Tf-Idf can be configured to achieve competitive results. With both features, concatenating separate body/article features did not achieve particularly good results.

In general, the BERT-based models took less time to train (2 hours), with exception to the similarity-based Tf-Idf models (<10 minutes).

In designing the deep learning models, it was found that the Tf-Idf features are more suited as sentence vectors, hence rendering most NLP-centric architectures incompatible. Concatenating the Tf-Idf features for input into a basic linear classifier ultimately yielded poor results. In contrast, both BERT-based deep classifiers performed equally or better than their machine learning counterparts. The BERT-based LSTM model improved the prediction results very marginally, as seen in table.3. Overall, the linear classifiers took up to 3 hours to train, whereas the LSTM classifier took up to 6.

The BERT-based LSTM for stance classification performs reasonably. It is most accurate in detecting ‘discuss’ inputs, followed by ‘agree’ and ‘disagree’. This is a difficult task in general, as articles that ‘agree/disagree’ with the headline can very easily be mistaken as ‘discussion’.

The final combined model in table.4 shows a classification accuracy of 0.91. However, this is heavily skewed due to the data imbalance, where the dataset contains a disproportionate amount of ‘unrelated’ pairs.

## Ethical Implications:

The aim with the fake-news dataset is to the stance of an article given a headline. By writing a headline related to an important topic, and pairing it with a large collection of articles, the process of finding related articles becomes easier. Further narrowing down the articles by stance allows human fact-checkers to identify articles that may be spreading misinformation. For example, given a topic such as ‘vaccination’, articles that ‘disagree’ with ‘vaccination’ can be excessively harmful in spreading misinformation. Ultimately, this project serves as a way to easily identify ‘fake news’ in hopes of better informing society. This could easily be misused, however, to manipulate the news being served to the public, in favor of a human fact-checker’s own bias.

Due to the very broad classes, it can be argued that bias is reduced; by classifying an article relative to a headline, the sentiments of the two are compared. Typical concerns for other models include the bias for doctors to be male for example. However, BERT can mitigate this to some extent, due to its ability to capture context.

## Conclusion:

This report proposes two feature extraction methods from text, finding that BERT features provided the best performance, at the cost of training time. It further proposes an architecture that successfully predicts headline-body relationship to a 98% accuracy. Finally, a full stance-classification network is produced by combining the previous relationship classifier with a separate stance classifier, for an accuracy of 91%. Some ethical implications are discussed, and it is concluded that BERT serves to mitigate some biases, and the risk largely depends on the user's intentions.

### References:

[1] A.K. Chaudhry, D. Baker, P. Thun-Hohenstein. Stance Detection for the Fake News Challenge: Identifying Textual Relationships with Deep Neural Nets.