**Fake News Challenge using Natural Language Analysis**

**Abstract**

The problem of fake news - stance classification in natural language processing entails determining viewpoint on a particular headline and its article body. This report suggests a paradigm with four classes: Agree, Disagree, Discuss, and Unrelated. The main objective is to examine how well different machine learning model in conjunction with TF-IDF and BERT model with deep learning architectures, perform. Based on the architecture and hyperparameters these models are constructed on, their performance in terms of Precision, Recall, and F1-Score will be evaluated.

**Proposed solution**

To assign a headline/body pair to a designated class, a two-level hierarchy of models is proposed. The first model determines if a headline and body pair are Related or Unrelated. The pair is then sent into our second model, which categorizes the related stances into Agree, Disagree, or Discuss, in response to a relevant prediction. The flow of the model and its implementation is as below:

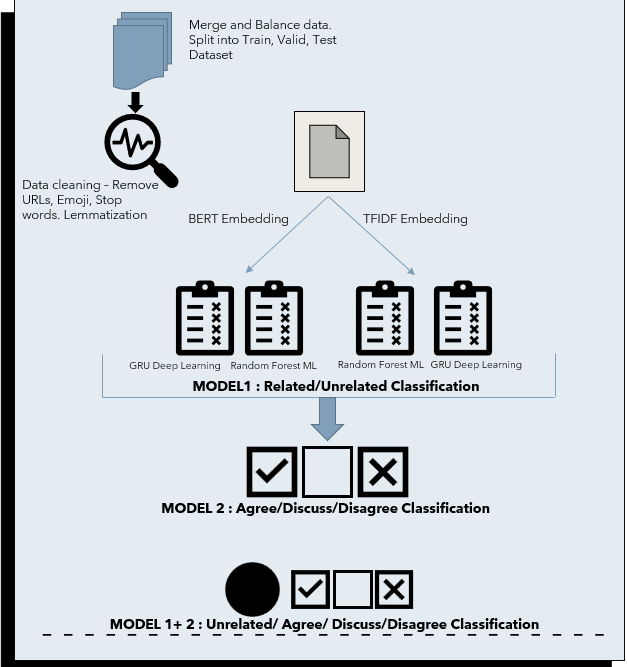


Figure 1: Fake News Classification Model

**TF- IDF**

Term Frequency-Inverse Document Frequency is used to assess a term's weightage in corpus of documents. A word that regularly appears in both the document and the corpus is less significant and revealing about the substance of the document. Tf-IDF is a lexical, not a semantic measure.

𝑡𝑓 (𝑡, 𝑑) = log (1 + 𝑓 (𝑡, 𝑑)) 𝑤ℎ𝑒𝑟𝑒 𝑓 (𝑡, 𝑑) = 𝑓𝑟𝑒𝑞𝑢𝑒𝑛𝑐𝑦 𝑜𝑓 𝑡𝑒𝑟𝑚 𝑡 𝑖𝑛 𝑑𝑜𝑐𝑢𝑚𝑒𝑛𝑡 𝑑

𝑖𝑑𝑓 (𝑡, 𝐷) = log (𝑁 𝑐𝑜𝑢𝑛𝑡 (𝑑 ∈ 𝐷: 𝑡 ∈ 𝑑)) 𝑤ℎ𝑒𝑟𝑒 𝑁 = 𝑛𝑢𝑚𝑏𝑒𝑟 𝑜𝑓 𝑑𝑜𝑐𝑢𝑚𝑒𝑛𝑡𝑠 𝑖𝑛 𝑡ℎ𝑒 𝑐𝑜𝑟𝑝𝑢𝑠 𝐷

𝑡𝑓𝑖𝑑𝑓(𝑡, 𝑑,𝐷) = 𝑡𝑓(𝑡, 𝑑) ∙ 𝑖𝑑𝑓(𝑡,𝐷)

Equation 1: TF-IDF

Advantages of TF-IDF:

* TF-IDF is computationally efficient and capable of handling big datasets.
* The TF-IDF method is adaptable and may be tailored to fit a variety of applications and domains.

Disadvantages of TF-IDF:

* Sparse data can be a problem with TF-IDF, especially if the number of unique terms in the corpus is very large.
* TF-IDF can be biased towards longer documents, as they tend to have more occurrences of each term.

**BERT**

Bidirectional Encoder Representations from Transformers and SBERT (Sentence-BERT) are two popular natural language processing models that use transformer-based architectures for text representation. They are trained to anticipate missing words in a sentence in the masked language modelling task to determine whether two provided phrases will follow one another. An adaptation of BERT made exclusively for embeddings is called SBERT. SBERT is trained on a task that measures sentence similarity rather than fine-tuning the previously learned BERT model on a downstream task.

Advantages of BERT:

* BERT employs a bidirectional transformer design, which enables it to capture the context of a word in a phrase by considering both the left and right context.
* BERT uses attention mechanism to recognize the intricate connections between words and their context depends on this attention process.

Disadvantages of BERT:

* BERT is a large and a complicated model that uses a lot of memory and processing power.
* BERT needs a lot of training data, which might not be accessible in all fields or languages.

**Data Preprocessing ad Balancing:**

Removing irrelevant information, standardizing text formats, and reducing vocabulary size can help the models to focus on important patterns and relationships in the data, leading to more accurate predictions and insights. To get the information out of raw data, data cleaning is done by removing special characters, URLs, Emojis. Finally, StopWords are removed from this dataset and Lemmatization is applied. Split of (Train data: Validate Data: Test Data) is = (8:1:1)

|  |  |  |
| --- | --- | --- |
|  | Raw Data | Balanced Data |
| Train Data | 39977 | 24162 |
| Validate Data | 4997 | 4997 |
| Test Data | 4998 | 4998 |

Figure (a): Stance Data and Data split

**Model 1: Related/ Unrelated Classification**

This report proposes 4 different methods with a combination between BERT/TFIDF and Machine learning Classifier / Deep Learning models to properly categorize related and unrelated articles.

**Implementation of Machine Learning models:**

1(a): TFIDF with Random Forest Machine Learning Classifier

This method combines TF-IDF for feature extraction, cosine similarity for gauging document similarity, and a machine learning classifier like Random Forest to train the model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| **Unrelated** | 0.95 | 0.94 | 0.95 | 3647 |
| **Related** | 0.85 | 0.88 | 0.86 | 1351 |
|  |  |  |  |  |
| **Accuracy** |  |  | 0.93 | 4998 |
| **Macro Average** | 0.90 | 0.91 | 0.91 | 4998 |
| **Weighted Average** | 0.93 | 0.93 | 0.93 | 4998 |

Chart, treemap chart

Description automatically generated

Fig 3: Results of Test Dataset – TFIDF + Random Forest

1(b): BERT with Random Forest Machine Learning Classifier

BERT creates contextualised word representations for the input sentences using a pre-trained transformer network. The variable-length input sequence is converted to a fixed-length vector by SBERT via a pooling operation over the transformer output.

This method combines BERT for feature extraction, cosine similarity for gauging document similarity, and a machine learning classifier like Random Forest to train the model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| **Unrelated** | 0.95 | 0.85 | 0.90 | 3647 |
| **Related** | 68 | 0.88 | 0.77 | 1351 |
|  |  |  |  |  |
| **Accuracy** |  |  | 0.86 | 4998 |
| **Macro Average** | 0.82 | 0.86 | 0.83 | 4998 |
| **Weighted Average** | 0.88 | 0.86 | 0.86 | 4998 |

Chart, treemap chart

Description automatically generated

Fig 5: Results of Test Dataset – TFIDF + Random Forest

**Implementation of Deep Learning Models:**

Justification for using GRU: Gated Recurrent Unit Network is a newer generation of Recurrent Neural Networks with a similar but simpler architecture than LSTM. GRU uses hidden state instead of cell state to transfer the Information hence GRU has less time and space complexity than LSTM. GRU addresses the vanishing gradient problem from which vanilla RNN suffers. With PyTorch Lightning, GRU has an automated learning rate identifier for our ADAM optimiser.

Architecture: A GRU layer with ReLU activation, a linear layer with sigmoid activation and sigmoid activation. The "headline" and "body" texts are the model's two inputs. The input texts are either converted into dense vectors using a TF-IDF vectorizer or encoded into embeddings by a pre-trained transformer (such as BERT), depending on the hyperparameter transformer.

Hyperparameters: If a transformer is used for encoding, the GRU layer's fixed input size is 768; if a TF-IDF vectorizer is employed, it is 20,000. The GRU layer has an output size of 512, two GRU layers, and a dropout rate of 0.2. A ReLU activation function, followed by a linear layer with a single output node, is applied to the output of the GRU layer. A sigmoid activation function is then used to the output of the linear layer to provide a probability score between 0 and 1.

The binary cross-entropy loss, which is frequently employed for binary classification issues, serves as the loss function. The model minimises this loss throughout training in order to update its parameters and enhance predictions. For the complete test set, the model computes the confusion matrix, accuracy, precision, recall, and F1-score. The logging system of PyTorch Lightning is used to record the metrics.

2(a): TF-IDF with GRU Deep Learning.

|  |  |  |
| --- | --- | --- |
|  | **Trail 1**  Hyperparameters used:  Batch Size = 32, Max Epochs = 5, dropout=0.4, Learning rate of the Adam optimizer: 1e-3 | **Trial 2**  Hyperparameters used:  Batch size = 128, Max Epochs = 5, dropout=0.2, Learning rate of the Adam optimizer: 1e-3 |
| **Test Accuracy** | 0.85877 | 0.87279 |
| **Test F1-Score** | 0.75514 | 0.78422 |
| **Test Precision** | 0.69480 | 0.72220 |
| **Test Recall** | 0.85288 | 0.86251 |

Table 2: Results of TFIDF + GRU

2(b): BERT with GRU Deep Learning

|  |  |  |
| --- | --- | --- |
| **Test Metric** | **Trail 1**  Hyperparameters used:  Batch Size = 32, Max Epochs = 10, dropout=0.4, Learning rate of the Adam optimizer: 1e-3 | **Trial 2**  Hyperparameters used:  Batch Size = 128, Max Epochs = 5, dropout=0.2, Learning rate of the Adam optimizer: 1e-3 |
| **Test Accuracy** | 0.92447 | 0.93649 |
| **Test F1-Score** | 0.86342 | 0.89127 |
| **Test Precision** | 0.80815 | 0.82508 |
| **Test Recall** | 0.94374 | 0.97109 |

Table 3: Results of BERT + GRU Model

**Comparison of all 4 Models**

* Accuracy = (True Positives + True Negatives) / (True Positives + False Positives + True Negatives + False Negatives)
* Precision = True Positives / (True Positives + False Positives)
* Recall = True Positives / (True Positives + False Negatives)
* F1 Score = 2 \* ((Precision \* Recall) / (Precision + Recall))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Test Accuracy** | **Test F1-Score** | **Test Precision** | **Test Recall** |
| **TFIDF + Random Forest** | 0.93 | 0.93 | 0.93 | 0.93 |
| **BERT + Random Forest** | 0.88 | 0.86 | 0.86 | 0.86 |
| **TFIDF + GRU** | 0.87279 | 0.78422 | 0.72220 | 0.86251 |
| **BERT + GRU** | 0.93649 | 0.89127 | 0.82508 | 0.97109 |

BERT + GRU Model have taken longer time to train itself, but it performed better than other methods in all 4 sectors: F1 Score, Accuracy, Precision, and recall. The longer train time could me due to system limitations- hence It is excluded.

**Model 2: Agree / Disagree / Discuss Classification**

The GRU (gated recurrent unit) neural network that underpins the sentiment classifier used here accepts two sets of input data: the headline and body. To create embeddings, each set of input data is first run through a pre-trained transformer network - DistilRoberta-base, a popular pre-trained transformer-based language model, is an effective and condensed version of the original RoBERTa model. Because DistilRoberta-base can capture the intricate semantic and contextual links between Headlines and Bodies, it has been proven to be successful at stance identification.

Following the concatenation and transmission of these embeddings through the GRU network, a ReLU activation function and a linear layer that generates a 3-dimensional output are applied.

The model calculates the difference between the predicted and real sentiment labels using the cross-entropy loss function and Overfitting is avoided by using a dropout rate.

The result is then transformed into 3 classifications using a SoftMax activation function.

|  |  |  |  |
| --- | --- | --- | --- |
| **GRU hyperparameters** | **Linear layer** | **Loss function &**  **Optimizer** | **DataLoader hyperparameters:** |
| input\_size: 1536  hidden\_size: 512  num\_layers: 2  dropout: 0.2 | input\_size: 512  output\_size: 3 | CrossEntropyLoss  Adam with learning rate of 1e-2 | batch\_size: 128  num\_workers: 2  pin\_memory: True  shuffle: False |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| **Agree** | 0.00 | 0.00 | 0.00 | 349 |
| **Disagree** | 0.28 | 0.49 | 0.35 | 83 |
| **Discuss** | 0.73 | 0.95 | 0.83 | 912 |
|  |  |  |  |  |
| **Accuracy** |  |  | 0.68 | 1344 |
| **Macro Average** | 0.33 | 0.48 | 0.39 | 1344 |
| **Weighted Average** | 0.51 | 0.68 | 0.58 | 1344 |

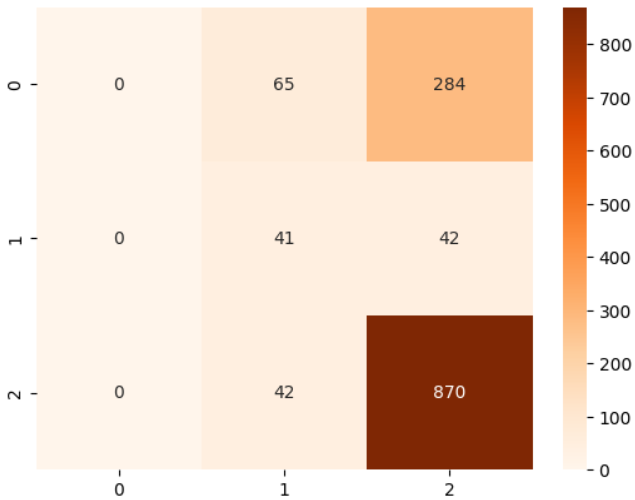


Figure 6: Results of GRU Model for Agree/Disagree/ Discuss class.

**Chart, line chart

Description automatically generated** **Chart, line chart

Description automatically generated**

Text, calendar

Description automatically generated with medium confidence

Figure 7 : Performance of GRU through 10 Epochs

The output of a GRU model that was trained over 10 epochs is seen above. Over time, both the training and test losses are reduced, and the test accuracy is consistently between 67 and 68%. The improvement in training accuracy is from 54.71% to 62.88%.

In Conclusion, Model 1 can Classify the Data into related and Unrelated Stance. Next, Model 2 takes the related Stance as Input and classifies them into Agree/Disagree/Discuss . Finally a 4 Class model can be built by mounting Model 1 on Model 2. This will require fast processing systems, high ram units and more GPUs.

**Ethical Implications:**

Due to their biases and risk of overreliance, fake news detection models that uses TF-IDF and BERT with Random Forest, and Deep Learning have ethical repercussions. These models can aid in spotting misleading information, but they can also result in the suppression of ideas, which might be detrimental to free speech and democracy. When implementing these models, it is essential to consider aspects such as validation, authentication of source, fairness, and accountability. The use of open data, open training methods, and open decision-making procedures may all contribute to transparency. Fairness must assure the objective handling of various information sources, and ongoing audits and reviews are required to guarantee their efficacy and morality.

Future Misuse: Potential biases in such models might lead to incorrect identification of bogus information or the suppression of reliable content. Future abuse of these models may also amplify some voices, restrict access to alternative viewpoints, and have an adverse effect on democracy and free expression.

Action: To minimize possible harm and assure the ethical and efficient use of these models, cooperation with experts in relevant domains is also important, continued research and development, consideration of alternative machine learning methods, and collaboration with subject matter experts are all necessary to increase the efficacy of these models.

# References

1. AlindGupta. (2014). *geeksforgeeks*. Retrieved from https://www.geeksforgeeks.org/gated-recurrent-unit-networks/
2. Ashish Vaswani, N. S. (2017). *Attention Is All You Need*. Retrieved from https://research.google/pubs/pub46201/: https://arxiv.org/pdf/1706.03762.pdf
3. *Github*. (n.d.). Retrieved from Github - FNC: https://github.com/FakeNewsChallenge/fakenewschallenge.github.io
4. *Github - FNC*. (n.d.). Retrieved from Github : https://github.com/FakeNewsChallenge/fnc-1
5. JHA, A. K. (n.d.). *One Hot Enc+LSTM 91.67% Acc*. Retrieved from Kaggle: https://www.kaggle.com/code/abhinavkrjha/one-hot-enc-lstm-91-67-acc
6. Loye, G. (2019). *Gated Recurrent Unit (GRU) With PyTorch*. Retrieved from FloydHub Blog: https://blog.floydhub.com/gru-with-pytorch/
7. Michael Waskom. (n.d.). *seaborn: statistical data visualization*. Retrieved from seaborn: statistical data visualization — seaborn 0.12.2 documentation (pydata.org)
8. Naik, K. (n.d.). *Github*. Retrieved from GitHub - Fake News: https://github.com/krishnaik06/Fake-News-Classifier
9. Nora Alturayeif, H. L. (2023, Jan). *SpringerLink*. Retrieved from A systematic review of machine learning techniques for stance detection and its applications: https://link.springer.com/article/10.1007/s00521-023-08285-7
10. Python · Toxic Comment Classification Challenge. (.). *CountVectorizer, TfidfVectorizer, Predict Comments.* .: Kaggle. Retrieved from https://www.kaggle.com/adamschroeder/countvectorizer-tfidfvectorizer-predict-comments
11. Reimers, N. (2022). *SentenceTransformers Documentation*. Retrieved from https://www.sbert.net/
12. TARIK, S. H. (2022). *Python · Stance detection dataset*. Retrieved from Kaggle: https://www.kaggle.com/code/sadoukhamzatarik/stance-based-model
13. Rao, D. and McMahan, B. (2019) Natural language processing with PyTorch build intelligent language applications using deep learning. 1st ed. Beijing: O’Reilly.
14. Stevens, E. et al. (2020) Deep learning with PyTorch. Shelter Island, NY: Manning Publications Co.
15. Bird, S., Klein, E. and Loper, E. (2009) , 1st ed. Cambridge [Mass: O’Reilly.
16. , Natural language processing with Python.
17. (Naik, n.d.) (Michael Waskom., n.d.) (Reimers, 2022) (Loye, 2019) (AlindGupta, 2014) (Nora Alturayeif, 2023)
18. Goldberg, Y. (2017), San Rafael, California: Morgan & Claypool. Neural network methods for natural language processing.
19. Milovanović, I. (2013) Python data visualization cookbook. Birmingham, UK: Packt Publishing.
20. Uwiragiye, E. and Rhinehardt, K.L. (2022) ‘TFIDF-Random Forest: Prediction of Aptamer-Protein Interacting Pairs’, IEEE/ACM transactions on computational biology and bioinformatics, 19(5), pp. 1–1. doi:10.1109/TCBB.2021.3098709.
21. PyTorch Lightning https://lightning.ai/docs/pytorch/latest/common/lightning\_module.html#minimal-example