A comparative study of Fake (Sarcasm) News Text Detection Using BERT and Machine Learning

732A81 Text Mining Project

Author:
Chayan Shrang Raj
(chash345)

Abstract

Fake news contains incorrect information about a certain entity that could mislead people in taking erroneous decisions or which may cause unrest in some countries. The globalization and digitalization of the world has only worsened the situation by introducing innumerable news sources (Chaudhary, 2021). Since, the veracity of everyday news plays a central part in people's lives, we aim to research and experiment if recently developed attention mechanism can retain the contextual meaning of the data and might provide better text classification results than traditional TF-IDF method. Furthermore, since the inception of attention mechanism, it is interesting to see how traditional feature extraction methods compare with today's state-of-the-art tools. This is a binary text classification problem; hence we have multiple metrics such as accuracy, precision, recall, F1-score and ROCAUC score to evaluate our model.

1 Introduction

Manual fact checking can solve some of the problems relating to fake news based on technical background check and contacting authorities but is too slow to cover massive information spreaders such as social media platform (EDPS, 2021). It's the sheer range of fake news spread over different communication networks that has posed a major challenge for us. In this paper, we have aimed to leverage artificial intelligence tools based on natural language processing and text mining to classify news based on its satirical property. The purpose of this project is to create a solution that could be used to detect and filter news headlines containing fake news (Aldwairi et al, 2018). Our attempt at this research is comparing the performance difference between traditional ML algorithms and Transformer Based algorithms. Two ML algorithms chosen are Multinomial Naïve Bayes and Random Forests with TF-IDF feature extractor and two Transformer based algorithms are BERT and distilBert with its own embeddings. Furthermore, we want to understand how attention mechanism uses context based feature extraction from texts in contrast to non-contextualized TF-IDF.

2 Theory

2.1 Supervised Learning

The supervised learning method utilizes the training data to understand and learn the information about the intended job of the model in order to foresee the hidden activities and patterns in the data (Z Khanam et al, 2021). The amount of data depends on the method we use to train our model such as transfer learning or training the model from scratch.

2.2 Natural Language Processing

Natural Language Processing or NLP for short, is a broad field that includes methods from classifying text corpus to a predefined classes to understanding a dialogues between a human and a chatbot. The benefits of NLP for businesses outweighs its difficulty in implementing such solution as it is so critical to businesses to tap on to large volumes of not just numerical data but text data like social media, customer support tickets, online comments, online reviews and more (Wolff, 2020).

2.3 Machine Learning Text Classification

Classical ML algorithms were primarily used to perform NLP tasks before the advancement in deep learning algorithms such as Logistic Regression for binary classification or Random Forest Classifier. There are many textual features that are important for a robust fake news detection problem since lexical features may potentially in fake writing style from the true content that may include some semantic features also(Xinyi Zhou et al, 2020).

2.4 Deep Learning Text Classification

It is the procedure of designating pre-defined labels for text and has diverse applications such as sentiment analysis, topic labeling, dialog act classification, question answering and much more (Qian Li et al, 2021). These algorithms are proven to be more robust and produce better results when treated with huge amounts of data. They have the capability of understanding the features and patterns of the text data automatically in contrast with classical machine learning algorithms where we had to create meaningful features from the text data.

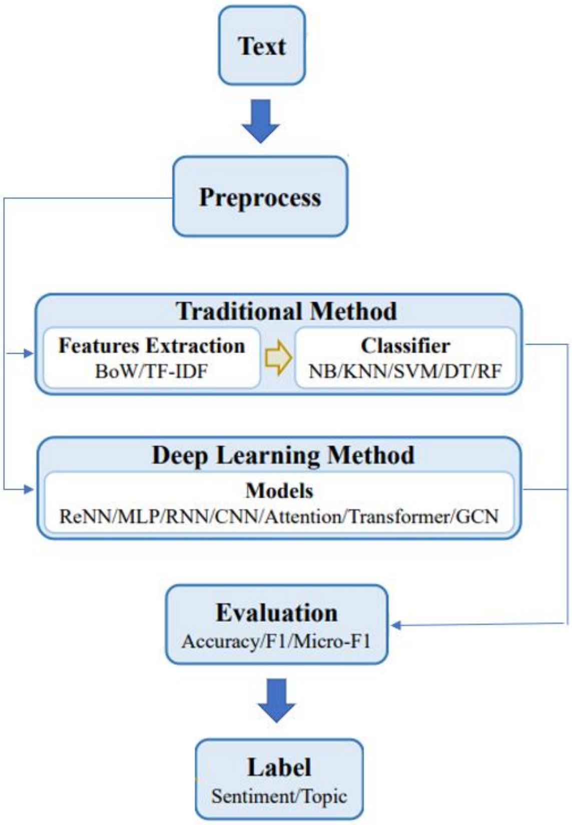


Figure 1: Flowchart describing lifecycle of text classification with classical methods in each module. It is important to extract essential features for standard machine learning methods, but they can be extracted automatically by deep learning algorithms. (Image credits: Author)

2.5 TF-IDF Feature Extractor

Term Frequency – Inverse Document Frequency is a statistical method that simply converts a corpus of texts into its corresponding numerical features to be able to model using a Machine Learning algorithm. Overcoming Bag-of-words shortcomings, this method also quantifies the importance of each word in the corpus and the document itself using the Inverse document frequency (Salton, 1983). This means, idf of a rare item should be high and of a common term should be low.

2.6 Transfer learning

Since the inception of well-defined machine learning models, it has opened an avenue for variety of industrial applications each with its own data requirements. Transfer learning is a method where a machine learning model is reused for a downstream task (similar task but with some constraints). Many smaller tasks can get very high accuracy by using a pre-trained model on similar data and modifying the architecture of the model for the small task (Browniee, 2017). Basically, pretrained models use features extracted from previous training on any kind of data they have been trained for but mostly that is industrial level data (size: petabytes of data) and use that knowledge to understand patterns of off different and unique datasets, which in our case, are the news headlines (text data). This saves vast compute and time resources required to develop neural network models on these problems. We have only used supervised pre-trained deep neural networks trained primarily on BooksCorpus (800 M words) and English Wikipedia (2500 M words) and then we fine-tuned the model for our dataset.

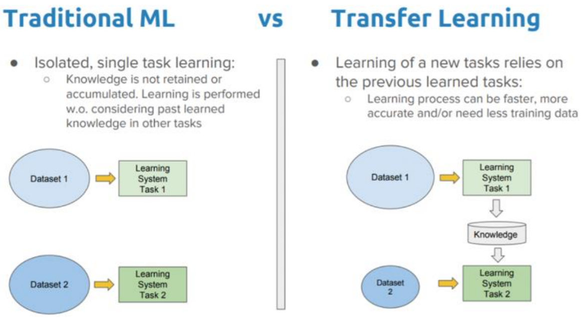

Traditional ML
vs
Transfer Learning
Isolated, single task learning:
Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks
Learning of a new tasks relies on the previous learned tasks:
Learning process can be faster, more accurate and/or need less training data
Dataset 1
Learning System Task 1
Dataset 2
Learning System Task 2
Dataset 1
Learning System Task 1
Knowledge
Dataset 2
Learning System Task 2

Figure 2: Traditional Learning vs Transfer Learning (Image credits: Dipanjan Sarkar)

3 Data

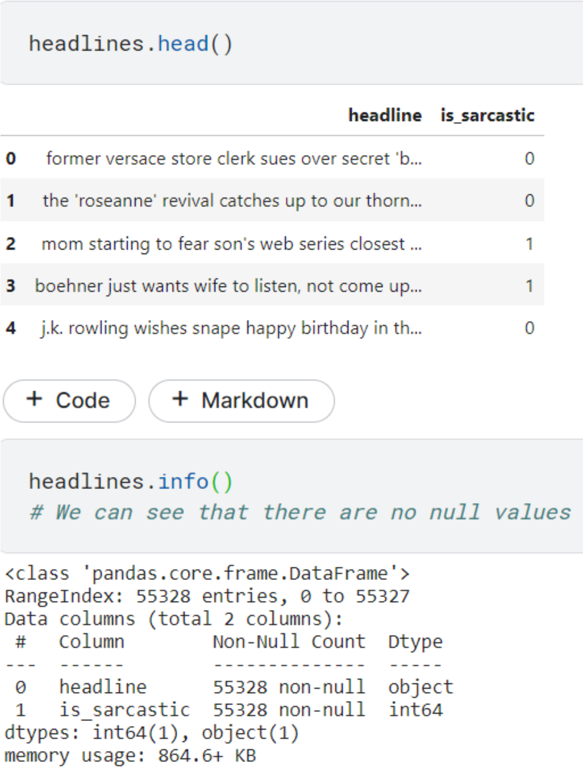
3.1 Data Description

The data is called ‘News Headlines Dataset for Sarcasm Detection’ which has been explored through Kaggle (Data science and machine learning platform) (Misra, 2023). It is primarily used for the task of fake news detection through sarcasm. There are many social media sources that generate fake news most commonly Twitter, Facebook, etc. But since most of them have unwanted noise in the form of hashtags, geographical nuances, labels, different languages, etc. the data source has been divided into two segments from verifiable sources namely “Huffpost” which provides real news and

"TheOnion" which provides us with sarcastic (fake) version of the news.

3.2 Data Format

The data is primarily in ".json" (Java script object notation) format that has python dictionary-based structure defining the elements in each sample. The data could be read using pandas json as pd.read_json() and pass in the location of the json file. The final dataset looks like this, it has 55328 rows with one column "headlines" that contains both real and fake news and second column "is_sarcastic" determining if the news is fake or not.



```
headlines.head()
```

	headline	is_sarcastic
0	former versace store clerk sues over secret 'b...	0
1	the 'roseanne' revival catches up to our thorn...	0
2	mom starting to fear son's web series closest ...	1
3	boehner just wants wife to listen, not come up...	1
4	j.k. rowling wishes snape happy birthday in th...	0

```
headlines.info()
# We can see that there are no null values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55328 entries, 0 to 55327
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   headline      55328 non-null  object
 1   is_sarcastic  55328 non-null  int64
dtypes: int64(1), object(1)
memory usage: 864.6+ KB
```

Figure 2: Above code snippet shows the structure of the data and info about the data types. (Image credits: Author)

3.3 Data Preprocessing

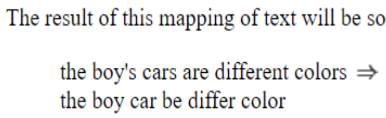
Text data is notorious for containing a lot of noise because of different linguistic styles, slangs, special characters, nuances and much more. One of the most important step in any NLP task is to preprocess the data and format it according to the model input requirements. In our case, we have also done several preprocessing steps before training our algorithm.

- **Feature Selection**: As we can see earlier, we have total of three features in our original dataframe. After analyzing the data set, we see that 'article_link' column is the link to the source of the headline, that does not give us any information about the content of the headline itself so we shall drop it. We are taking only two columns forward for data preprocessing, i.e, 'headline' and 'is_sarcastic'.

- **Stopwords Removal**: A stop word is a frequently occurring word in a text that has minor or no effect on the meaning of the sentence. It does not contribute to the context of the text and hence search engines have been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query (Upadhyay, 2023). NLTK is one of the libraries that has garnered much attention while preprocessing text data as it stores a list of stopwords stored in 16 different languages and we use the same in our use case. We have also used regex (regular expressions) library to remove any special characters and punctuation from the text.

- **Outlier Removal**: Outliers are defined as entities showing exceptional properties, different from the same group but similar to other groups. It is important to remove outliers before feeding it into our model as they could become a source of noise and make it difficult for the model to discern patterns in the data. We have removed the outliers from the headlines data.

- **Lemmatization**: For grammatical reasons, documents/texts may use different forms of a word, such as play, plays, played or playing. Additionally, there are families of derivationally related words with similar meanings, such as democracy, democratic, and democratization. In many scenarios, it appears as if it would be useful for a query for one of these words to return documents/texts that contain another word in the set (Stanford, 2009). Lemmatization usually refers to the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.



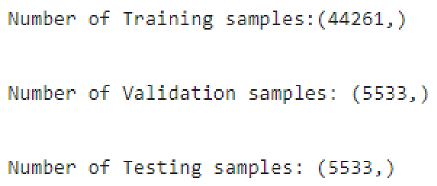
The result of this mapping of text will be something like:

the boy's cars are different colors ⇒
the boy car be differ color

Figure 3: Example of lemmatization on a sentence. (Image credits: Stanford, 2009)

In our case, we again use NLTK 'wordnet' lemmatizer to preprocess our data. It is a large, freely, and publicly available lexical database for English language aiming to establish structured semantic relationships between words.

- **Data Splitting:** We are performing supervised learning and for that we must split our data into train/valid/test. The ratio of the split has been set to 80:10:10 which means 80% of the whole dataset is train, 10% of the left is validation set and the remaining is testing set. We train our model on training set, tune model's hyperparameters which evaluating on validation set and finally test our model on testing set. We have total 55328 rows (samples) and 3 columns (features) in our original dataset. It looks something like below after the data split:



```
Number of Training samples:(44261,)


Number of Validation samples: (5533,)


Number of Testing samples: (5533,)
```

Figure 4: Data split between train/valid/test set. (Image credits: Author)

4 Method

We approached the problem using a standardized structure similar to any Natural language processing task where after data preprocessing, we build our models and set the hyperparameters. For this problem, we are going to take advantage of certain libraries, tools, machine learning libraries and transformer-based models.

4.1 TF-IDF Vectorizer

One issue with performing Natural language tasks with machine learning is that the data is in the form of text. We need to convert our text data into computer understandable numeric data because even normal computers can't work with just text data. For this, purpose, we have text representation algorithms that convert text data into numeric vectors and one such algorithm is TF-IDF (Term Frequency – Inverse Document Frequency) (Salton, 1983) which is very popular for traditional machine learning algorithms. Term frequency is the number of times a word appears in a document. Document frequency indicates how common the term is, by calculating the number of documents that contains that word. Inverse document frequency assigns a weight to each term depending on whether that term is scarce (more weight) or common (less weight).

$$idf = log(n/df)$$

$$w\ (weight) = tf\ x\ idf$$

4.2 Multinomial Naïve Bayes

Naïve bayes algorithm is derived from Bayes theorem using conditional and prior probability. Relevance of Naïve in this algorithm indicates that the features are mutually independent. The Bayes theorem by Thomas Bayes, estimates the likelihood of occurrence of an event based on prior knowledge of the event's conditions (UpGrad, 2022). For example, if we have an event B, we can calculate the likelihood of event A. It's based on the formula: P(A|B) = P(A) * P(B|A)/P(B).

P(B) = prior probability of event B
P(A) = prior probability of event A
P(B|A) = occurrence of event B given event A probability

For our case, when we have all the features, it can calculate the likelihood of corresponding classes with greatest probability.

4.3 Random Forests

Random forests is a collection of a number of decision trees acting together as an ensemble method (Breiman, 2001). It can handle both regression and classification tasks since the elemental decision tree is a rule-based method. Individual decision trees could be prone to bias and overfitting but when multiple decision trees form an ensemble, they overcome this problem by averaging out each classification from the trees which makes this algorithm highly robust to bias and outlier effects (IBM, 2023). A major difference between decision trees and random forests is that decision trees will treat all the features while data split, but a random forest classifier will only select a subset of those features that induces certain level of randomness and decreases bias.

4.4 Transformers

We will be using huggingface (huggingface, 2020) implementation of pre-trained transformers model in our project. The transformer in NLP is an ingenuous approach that aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease. One of the major contributions in these models is the concept called **self-attention** mechanism which helps the model create similar connections but within the same sentence, it allows us to focus on parts of out input sequence while we predict our output sequence.

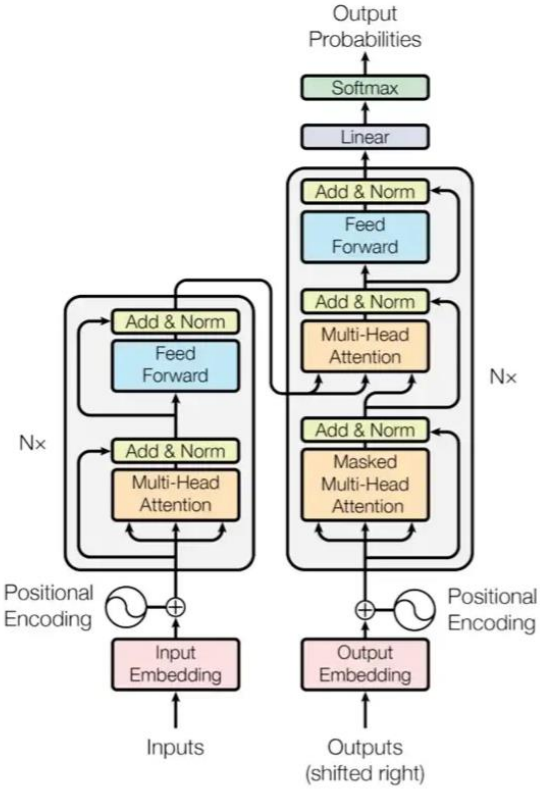
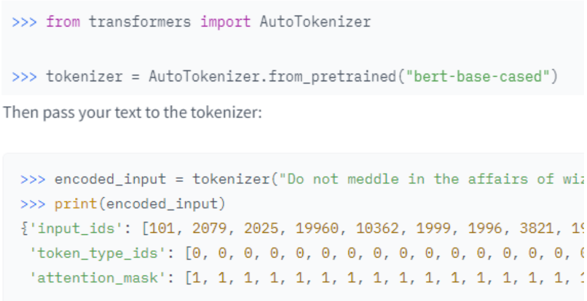


Figure 5: Transformer architecture (Image credits: huggingface, 2020)

It contains one encoder block which maps an input sequence of symbol representations to a sequence of representations. And decoder blocks which given z, generates an output sequence of symbols one element at a time.

4.5 BERT Tokenizer

BERT stands for Bidirectional Encoder Representations from Transformers and is a language representation model by Google (Kulshesthra, 2020). It is one of the most popular and go to transformer-based architecture for Natural language processing tasks. Simply put, it is a stack of bunch of Transformer's Encoder. We shall be using different implementations of BERT for experimentation purposes and produce results containing different metrics as discussed earlier. It oversees preparation of text inputs for a model into appropriate numerical data which the model can understand and train on it. Basically, it converts text into a sequence of tokens, create a numerical representation of the tokens, and assemble them into tensors.



```
>>> from transformers import AutoTokenizer


>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
```

Then pass your text to the tokenizer:

```
>>> encoded_input = tokenizer("Do not meddle in the affairs of wi
>>> print(encoded_input)
{'input_ids': [101, 2079, 2025, 19960, 10362, 1999, 1996, 3821, 1
 'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

Figure 6: The tokenizer using huggignface tansformer library and calling Autotokenizer.frompretrained method for 'bert-base-cased' model. (Image credits: huggingface, 2020)

The tokenizer returns a dictionary with three important items:

- input_ids: are the indices corresponding to each token in the sentence.
- attention_mask: indicates whether a token should be attended to or not.
- token_type_ids: identifies which sequence a token belongs to when there is more than one sequence.

These elements are finally converted to actual tensors that get fed to the model.

4.6 Fine-tuning BERT on our custom dataset

Fine-tuning is defined as utilizing the model architecture, pre-trained on a bigger dataset, to do various machine learning tasks such as classification, segmentation, generation by swapping out the appropriate inputs or outputs. To train-task specific models, we add an extra output layer to existing BERT and fine-tune the resultant model. Essentially, we do not change the central part of the BERT model but tweak input/output layers which allows change in only minimal number of parameters which need to learned from scratch making the procedure fast, cost and resource efficient.

We shall be using 2 different BERT implementations for our downstream task:

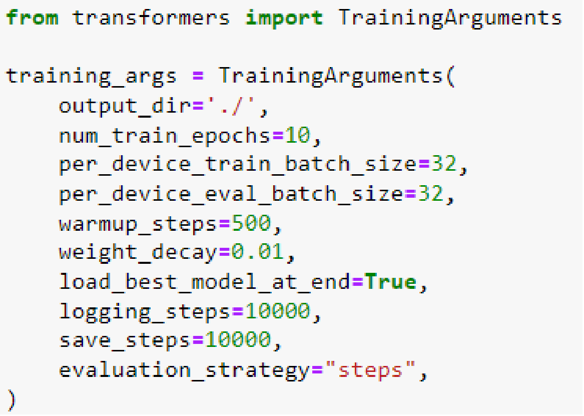
- BERT base (uncased) – Pretrained model on English language using a masked language modeling (MLM) objective (Devlin et al, 2019). This model is uncased: it does not make a difference between English and English.

- DistilBERT base uncased – This model is a distilled version of the BERT base model (Sanh, 2019). It is a transformer model, smaller and faster than BERT, which was pretrained on the same corpus in a self-supervised fashion, using the BERT base model as a teacher.

4.7 Model Training (Huggingface Trainer)

The Trainer class on huggingface is a simple method for training any kind of model, it provides an API for feature-complete training in PyTorch for most standard use cases. It requires another API called 'TrainingArguments' which specifies all the points of customization during training. We can also create our custom methods to produce custom behavior. The Trainer class is optimized for huggingface Transformer models.

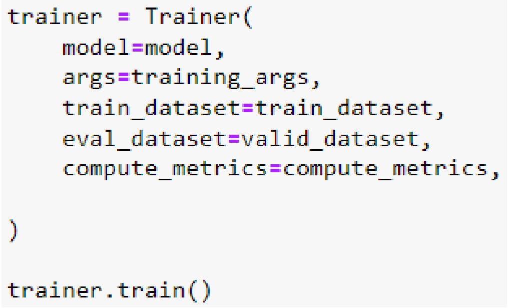
Below is the TrainingArguments that we have used to provide customization during model training:



```
from transformers import TrainingArguments

training_args = TrainingArguments(
    output_dir='./',
    num_train_epochs=10,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=32,
    warmup_steps=500,
    weight_decay=0.01,
    load_best_model_at_end=True,
    logging_steps=10000,
    save_steps=10000,
    evaluation_strategy="steps",
)
```

Figure 8: Training Arguments as a parameter to huggingface Trainer. (Image credits: Author)

And the training step is where we define the train and eval data sets along with optimizers and model parameters.



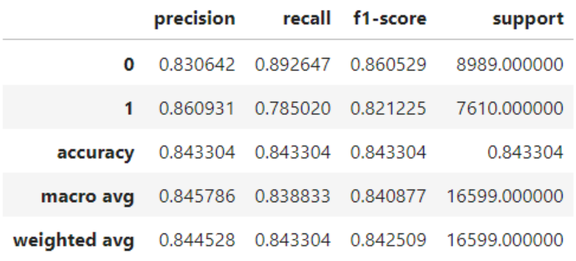
```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=valid_dataset,
    compute_metrics=compute_metrics,

)

trainer.train()
```

Figure 9: huggingface Trainer API to train the model. (Image credits: Author)

5 Results

Our binary text classification problem can be evaluated and compared with several metrics such as accuracy, precision, recall, F1-score or AUROC. Below we have summarized the result from each model describing each metric:

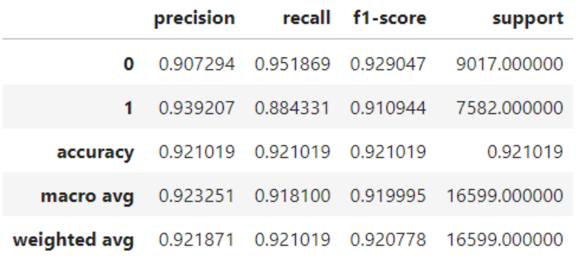
- **Multinomial Naïve Bayes**



	precision	recall	f1-score	support
0	0.830642	0.892647	0.860529	8989.000000
1	0.860931	0.785020	0.821225	7610.000000
accuracy	0.843304	0.843304	0.843304	0.843304
macro avg	0.845786	0.838833	0.840877	16599.000000
weighted avg	0.844528	0.843304	0.842509	16599.000000

Figure 10: Image by Author

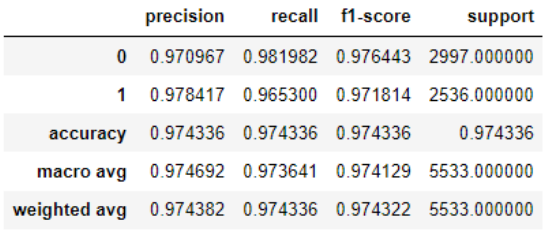
- **Random Forests**



	precision	recall	f1-score	support
0	0.907294	0.951869	0.929047	9017.000000
1	0.939207	0.884331	0.910944	7582.000000
accuracy	0.921019	0.921019	0.921019	0.921019
macro avg	0.923251	0.918100	0.919995	16599.000000
weighted avg	0.921871	0.921019	0.920778	16599.000000

Figure 11: Image by Author

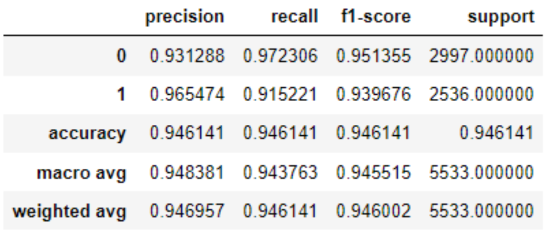
- **BERT**



	precision	recall	f1-score	support
0	0.970967	0.981982	0.976443	2997.000000
1	0.978417	0.965300	0.971814	2536.000000
accuracy	0.974336	0.974336	0.974336	0.974336
macro avg	0.974692	0.973641	0.974129	5533.000000
weighted avg	0.974382	0.974336	0.974322	5533.000000

Figure 12: Classification report of BERT base uncased model.

- **DistilBERT**



	precision	recall	f1-score	support
0	0.931288	0.972306	0.951355	2997.000000
1	0.965474	0.915221	0.939676	2536.000000
accuracy	0.946141	0.946141	0.946141	0.946141
macro avg	0.948381	0.943763	0.945515	5533.000000
weighted avg	0.946957	0.946141	0.946002	5533.000000

Figure 13: Classification report of DistilBERT base uncased model.

Since we have a balanced data set, we can check the Accuracy can also give us a good idea of our models as shown below:

Model	**Accuracy**
Multinomial Naïve Bayes	84.43%
Random Forest	92.10%
BERT	97.43%
DistilBERT	94.63%

Table 1: Model Accuracy

All the models ran for 10 epochs using transfer learning method and we see that BERT-base-uncased produced the highest accuracy among the four models. It is a vanilla BERT model, and it was interesting to see that DistilBERT and ALBERT came close to it while having less parameters to learn and faster training time. ELECTRA is a novel idea of using discriminator and generator and gives very good results after 10 epochs.

6 Discussion

In our research we have tried to experiment and understand how new innovations in Natural Language processing have shown promising results in text classification tasks. The choice of methods was motivated by an attempt to understand how word embeddings can play a key role in text classification. For traditional machine learning algorithms, TF-IDF vectorizer was used that fails to capture deeper contextual understanding in texts since it takes each word independently and shows no correlation or importance to other words in a sentence. We can see the comparison between traditional machine learning algorithms and transformer-based algorithms where Naïve Bayes algorithm and Random Forests have an accuracy of 84% and 92% respectively which is much below than BERT models having an accuracy of around 98% for the same dataset. This can show us that using attention layers, which contains several transformer encoders giving it the power to apply bidirectionality in extracting features from the texts and exhibiting on par performance with the best machine learning performing models (Alghamdi, J., 2022). It should also be noted that transformers models are not specifically pre-trained for fake news detection problems rather on a large general corpus of text data that doesn't give it a way ahead for this specific task of fake news detection but because of its innovative bidirectional attention ability that really steps up for NLP tasks. One of the drawbacks of BERT models could be the training time since it works with densely computed word vectors and it could depend on the embedding sizes (Devlin, 2018). It was also interesting to see that minimal text preprocessing was done for the BERT model. The power of transfer learning where we could leverage a pre-trained model for our specific task with decent computational resources. Even though, we are using transfer learning, these models need to learn large amounts of parameters and consequently it is always better to use GPU instead of CPU. We trained our models on Nvidia GPU P100 which reduced the training time from almost 120 minutes to 30 minutes for each model.

7 Conclusion

We introduced Transformer-based Natural language processing specific models to classify and filter our fake news from real news based on headlines from two different sources (real news and sarcastic news). Comparing BERT based results with traditional machine learning algorithms using TF-IDF vectorization, we can see the performance of BERT models is much better and it is interesting to compare different word embedding methods. Traditional feature extraction techniques may come across as a lost art now, but it is difficult to experiment this as the performance also depends on the type of dataset used, feature engineering, style content and much more. We can move towards more innovative solutions like attention mechanism used in transformers-based methods for capturing useful clues from a short piece of text based on pre-training methods also. We have also used models like DistilBERT which is a pre-trained version of BERT, 40% smaller, 60% faster and that retains 97% of the language understating capabilities, to demonstrate that such model could be a compelling option for on-the-fly edge applications such as smart watch, smart car, etc.

References

https://medium.com/mlearning-ai/fine-tuning-bert-for-tweets-classification-ft-hugging-face-8afebadd5dbf

https://www.thepythoncode.com/article/finetuning-bert-using-huggingface-transformers-python

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

Xinyi Zhou, Atishay Jain, Vir V. Phoha, Reza Zafarani, “Fake News Early Detection: An Interdisciplinary Study” https://doi.org/10.48550/arXiv.1904.11679 (2020).

Salton, Gerald. "McGill, Michael. Introduction to modern information." (1983)

www.upgrad.com/blog/multinomial-naive-bayes-explained

Breiman, L. Random Forests. *Machine Learning* **45**, 5–32(2001), https://doi.org/10.1023/A:1010933404324

https://www.ibm.com/se-en/topics/random-forest

https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

https://edps.europa.eu/press-publications/publications/techsonar/fake-news-detection_en

Monther Aldwairi, Ali Alwahedi, *Detecting Fake News in Social Media Networks, Procedia Computer Science,* Volume 141, 2018, Pages 215-222, https://doi.org/10.1016/j.procs.2018.10.171.

https://iopscience.iop.org/article/10.1088/1757-899X/1099/1/012040/pdf

https://machinelearningmastery.com/natural-language-processing/

https://monkeylearn.com/blog/what-is-natural-language-processing/

Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. 2021.

A Survey on Text Classification: From Traditional to Deep Learning. ACM Trans. Intell. Syst. Technol. 37, 4, Article 111 (April 2021), 39 pages. https://doi.org/10.1145/1122445.1122456

https://machinelearningmastery.com/transfer-learning-for-deep-learning/

Misra, Rishabh and Prahal Arora. "Sarcasm Detection using News Headlines Dataset." AI Open (2023)

https://www.geeksforgeeks.org/removing-stop-words-nltk-python/

https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

https://huggingface.co/docs/transformers/index

https://towardsdatascience.com/keeping-up-with-the-berts-5b7beb92766

Victor Sanh and Lysandre Debut and Julien Chaumond and Thomas Wolf, *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.* NEURIPS, 2019.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut, *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*, International Conference On Learning Representations, 2019.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, Christopher D. Manning, *ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS*, ICLR, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, arXiv:1810.04805, 2018

Alghamdi, J.; Lin, Y.; Luo, S. A Comparative Study of Machine Learning and Deep Learning Techniques for Fake News Detection. *Information* **2022**, *13*, 576. https://doi.org/10.3390/info13120576