Detecting Fake News Using Sentence Transformer Embeddings and Supervised Learning

Melih Cifci - mmc366, Section 6
Yunus Ulusoy - yeu1, Section 5
Rutgers University - 198:439:06
mmc366@scarletmail.rutgers.edu, yeu1@scarletmail.rutgers.edu

2. Research Question and Motivation

2.1 Research Question

How can we utilize and optimize Data Science and Natural Language Processing (NLP) tools and technology to efficiently clean, filter, and detect fake news using different libraries and frameworks?

2.2 Motivation

Developments in the generative AI models have made it cheaper to generate large amount of content very quickly and this is accelerating the spread of both mis- and disinformation across social media and digital platforms. Manual fact-checking is very time-consuming and insufficient for real-time usage. We need effective tools to allow us to identify and prevent the spread of fake news. This project aims to leverage transformer-based NLP models to classify fake news documents.

2.3 Background

This project draws on concepts from previous lectures in class on text vectorization, embeddings, preprocessing, transformers, classification, and machine learning models. The use of contextual embeddings through transformer models directly aligns with NLP techniques discussed in class. This a supervised learning project.

3. Data

3.1 Data Sources

For this project, we utilized the ISOT Fake News Dataset from the open-source website of the University of Victoria (Ahmed et al. 2018). There are 23,481 fake news and 21,417 real news documents in this dataset. Fake news are collected from websites flagged as unreliable by Politifact and Wikipedia. Below are two shortened examples of fake news from this dataset.

Fake news example 1:

“ Donald Trump Sends Out Embarrassing New Year’s Eve Message; This is Disturbing

Donald Trump just couldn t wish all Americans a Happy New Year and leave it at that. Instead, he had to give a shout out to his enemies, haters and the very dishonest fake news media. The former reality show star had just one job to do and he couldn t do it. As our Country rapidly grows stronger

and smarter, I want to wish all of my friends, supporters, enemies, haters, and even the very dishonest Fake News Media, a Happy and Healthy New Year, President Angry Pants tweeted. 2018 will be a great year for America! As our Country rapidly grows stronger and smarter, I want to wish all of my friends, supporters, enemies, haters, and even the very dishonest Fake News Media, a Happy and Healthy New Year. 2018 will be a great year for America!...”

Fake news example 2:

“ Drunk Bragging Trump Staffer Started Russian Collusion Investigation

House Intelligence Committee Chairman Devin Nunes is going to have a bad day. He s been under the assumption, like many of us, that the Christopher Steele-dossier was what prompted the Russia investigation so he s been lashing out at the Department of Justice and the FBI in order to protect Trump. As it happens, the dossier is not what started the investigation, according to documents obtained by the New York Times.Former Trump campaign adviser George Papadopoulos was drunk in a wine bar when he revealed knowledge of Russian opposition research on Hillary Clinton.On top of that, Papadopoulos wasn t just a covfefe boy for Trump, as his administration has alleged. He had a much larger role, but none so damning as being a drunken fool in a wine bar. Coffee boys don t help to arrange a New York meeting between Trump and President Abdel Fattah el-Sisi of Egypt two months before the election. It was known before that the former aide set up meetings with world leaders for Trump, but team Trump ran with him being merely a coffee boy...”

3.2 Data Processing

Preprocessing:

- **Binary Classification:** We labeled the fake news with [1] and the real news with [0]

- **Cleaning + Normalization**: Removal of URLs, HTML tags, and non-alphabetic characters using regex and lowercased.
    - We utilize the **def preprocess(text)** function for preprocessing.
        - text = text.lower()
        - text = re.sub(r'http\S+|www\.\S+|<.*?>', '', text)
        - text = re.sub(r'[^a-z\s]', '', text)

- **Tokenization and Stopword Removal:** Text was split into words; common English stopwords were removed using NLTK.
    - tokens = text.split()
    - stop_words = set(stopwords.words('english'))

- **Lemmatization**: WordNetLemmatizer.
    - Lemmatization is a technique that reduces a word's full length to the skeletal definition (of some sort) known as a lemma, working → work
        - Allows us to clean the tokens better for vectorization

- **Embedding:** Transform cleaned text into 384-dimensional dense vectors using sentence-transformers/all-MiniLM-L6-v2.
    - Embeddings are vector representations of specific objects that account for semantics within running the LLM model.

    - Similar sentences or words meaning the same end up close in this embedding space, allowing classifiers to draw decision trees more effectively.
    - They also group similar objects to be easily interpreted for analysis, like Cosine Similarity, one of the widely used models.
    - We chose this embedding transformer due to its size and performance on classification tasks as reported on the mteb leaderboard on Hugging Face.

- **Train/Test Split:** Stratified 80/20 split to preserve class imbalance in both sets.
    - In a Train/Test split we utilize 80% of the data used to train the data and apply it on the 20% testing case portion.
    - We need to stratify the split because it allows us to have a better representation of the fake/real ratios.
    - This generally improves the learning of the model by allowing it to learn with the training and data, but also ensures balanced data.

- **Imbalance Handling:** Creating a balanced weight of the data
    - To evaluate our models on this highly imbalanced problem, we rely on metrics that treat each class equally. Specifically, we report the macro-averaged F1-score and inspect precision–recall (PR) curves, which are more informative than ROC curves when one class is rare.
    - Due to fake news being around ~5% of the data set corpus, it is more rare to be encountered; therefore, this classification emphasizes the false negatives where a specific fake news is classified as real news, and therefore needs to be adjusted according to
    - Apply weighting for XGBoost
        - (class_weight='balanced' for sklearn; scale_pos_weight for XGBoost) in modeling

3.3 Challenges

Our dataset is balanced with approximately 20,000 real and 20,000 fake news documents. We don't know the real distribution of fake news among all news. News can be subjective, but outright fake news shouldn't be very prevalent for someone who has high digital literacy and can discern the quality of news sources. To resemble this scenario, we only kept 5% of the fake news, which reduces our sample of fake news to 1174 fake news documents. Ideally, we would want to represent the real-world proportions.

Another challenge we had was that the model is not equipped for advanced semantics, meaning they are not explicitly trained to detect satire, parody, or partial truths. Therefore, it will reduce the precision of the model's performance.

We utilize a public dataset that is labeled as real or fake news based on Politifact and Wikipedia. Classifying fake news is inherently a difficult task. It may be challenging to separate subjective reporting and facts sometimes. The quality of the data we train our model on is important. So our trained model's performance relies on the data quality of the data it is trained on.

4. Methodology and Analysis Plan

4.1 Techniques

- **Preprocessing + Embedding:**
  - Preprocessing pipeline including cleaning, lemmatization, and embedding transformation using **MiniLM-L6-v2.**
- **Model Training:**
  - **Cross-validation:** We split the data with a stratified 80/20 train/test split to preserve the fake/real ratio, then apply Stratified 5‑fold cross‑validation on the training set. This ensures each fold reflects the overall class imbalance and yields stable estimates of macro‑F1. After CV, models are retrained on the full training set before final evaluation on the hold‑out test set.
    - In cross validation, the training data is split into five equal "folds," each preserving the fake/real ratio; the model trains on four folds and validates on the fifth, repeating five times so every fold serves as the validation set once, this ultimately ensures the data is well trained and allows better performance estimates (precision) meanwhile reducing overfitting.

Machine Learning Model/Algorithms:

Logistic Regression
Logistic Regression is a linear model that uses a logistic function to turn the weighted sum of input features into a probability between 0 and 1. It learns how much each feature contributes to the chance that an article is fake. In our case, we used class weighting to make the model pay more attention to fake news, which helps boost recall without lowering precision too much. This is especially helpful since fake news is usually the minority class.

Support Vector Machine (SVM)
SVM is a model that finds the best boundary (or hyperplane) that separates two classes while leaving the biggest possible margin between them. It uses hinge loss, which puts more focus on the examples near the boundary; these are usually the toughest ones to classify. This margin-based approach works well with high-dimensional data, like the embeddings we used, and it helps the model generalize better. It's also less sensitive to outliers, which makes it a strong option for tasks like fake news detection.

Random Forest
Random Forest is an ensemble model made up of multiple decision trees. Each tree is trained on a different random sample of the data, and the final prediction is made based on the majority vote. The trees split the data based on feature values and can capture complex patterns and interactions. However, Random Forest can sometimes overfit, especially with noisy or high-dimensional data. Using class weights helps it focus more on fake news, which is useful when working with imbalanced datasets like this one.

XGBoost
XGBoost is a powerful model that builds a series of decision trees one after another, with each new tree trying to fix the mistakes made by the previous ones. It does this by focusing on the residual errors and minimizing a specific loss function. For our fake news task, XGBoost was helpful because it naturally gives more attention to the harder examples. We also used a parameter called scale_pos_weight to make the fake news class more influential during training, which improved its ability to catch minority class examples.

4.2 Tools and Libraries

- **Core Libraries:** Pandas, NumPy, Scikit-learn, NLTK, re, Matplotlib, torch, xgboost
- **Modeling & Embeddings:**
    - transformers (HuggingFace) and sentence transformers for embeddings.
    - Logistic Regression, SVM, XGBoost, Random Forest
- **Development:** Google Colab utilizing Google Drive integration to import the datasets.

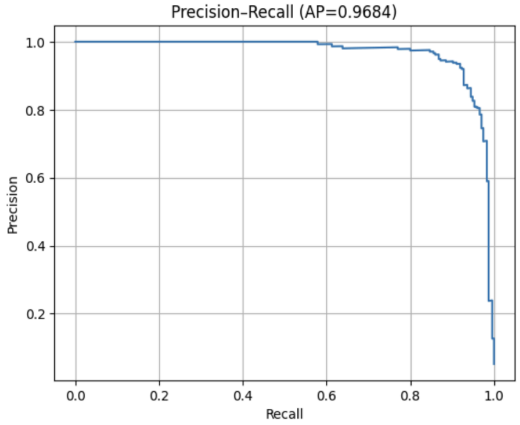
5. Expected Outcomes and Evaluation

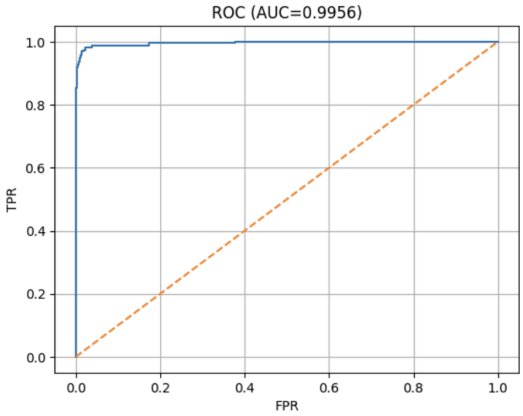
5.1 Anticipated Findings

We predict that transformer-based embeddings will present strong performance for all of the classifiers, with linear models, which are logistic regression and SVM, delivering particularly high precision and recall, as shown in the plots. xgboost, with default parameters, will keep up with the linear methods; however is not optimized completely. It would be better if we target hyperparameter tuning. Random forest shows a preference for precision over recall unless specifically changed. We believe that compared to a simple TF-IDF model, all of these models will have a higher macro (F1 macro), average precision (AP), and ROC→AUC, allowing the model to detect the fake news at a more optimized version. These are initial research however, we are not yet too familiar with how the models work and impact data due to minimal prior knowledge.

5.2 Evaluation Metrics

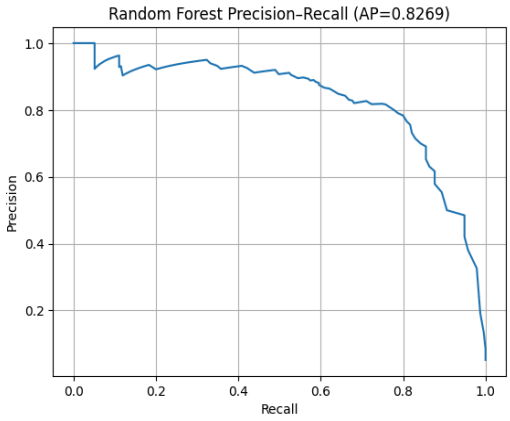
Logistic Regression (F1 Macro + Precision–Recall + ROC Plot + F1-Score + Confusion Matrix)

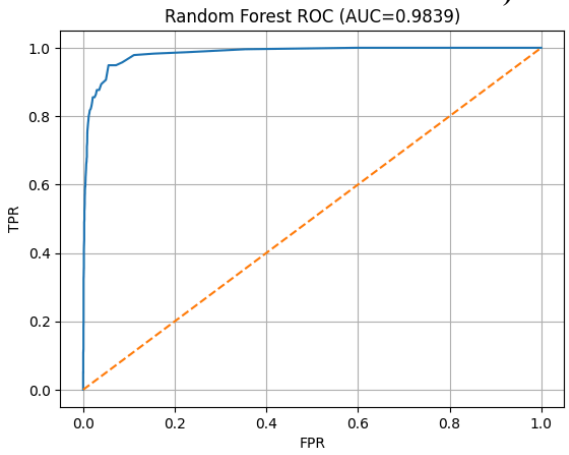




```
LogisticRegression CV F₁ macro: 0.9199 ± 0.0123

LogisticRegression on test set:
[[4210   74]
 [   7  228]]
              precision    recall  f1-score   support

           0     0.9983    0.9827    0.9905      4284
           1     0.7550    0.9702    0.8492       235

    accuracy                         0.9821      4519
   macro avg     0.8767    0.9765    0.9198      4519
weighted avg     0.9857    0.9821    0.9831      4519
```

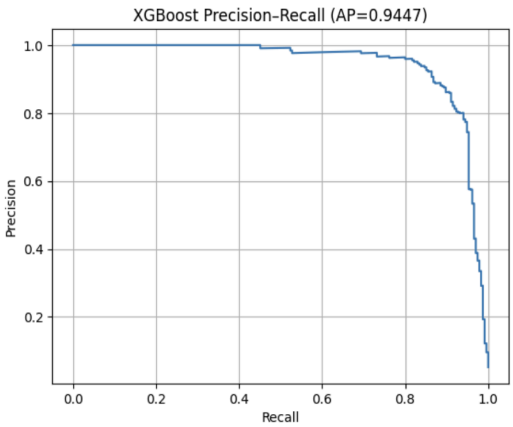
Random Forest (F1 Macro + Precision–Recall + ROC Plot + F1-Score + Confusion Matrix)

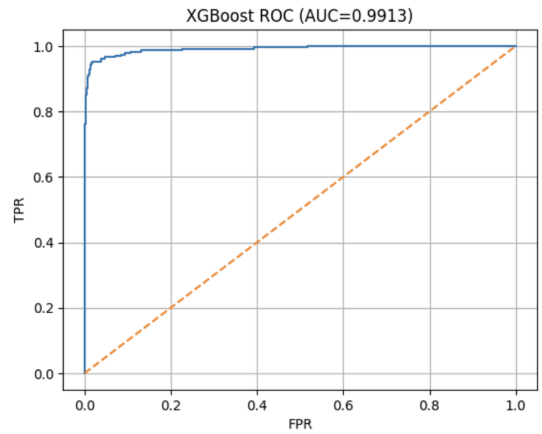





```
Random Forest 5-Fold CV F₁ macro: 0.7631 ± 0.0081

Random Forest on test set:
[[4274   10]
 [ 127  108]]
              precision    recall  f1-score   support

           0     0.9711    0.9977    0.9842      4284
           1     0.9153    0.4596    0.6119       235

    accuracy                         0.9697      4519
   macro avg     0.9432    0.7286    0.7981      4519
weighted avg     0.9682    0.9697    0.9649      4519
```

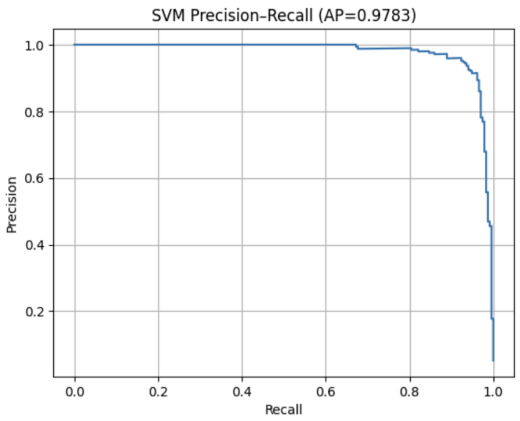
XGBoost (F1 Macro + Precision–Recall + ROC Plot + F1-Score + Confusion Matrix)

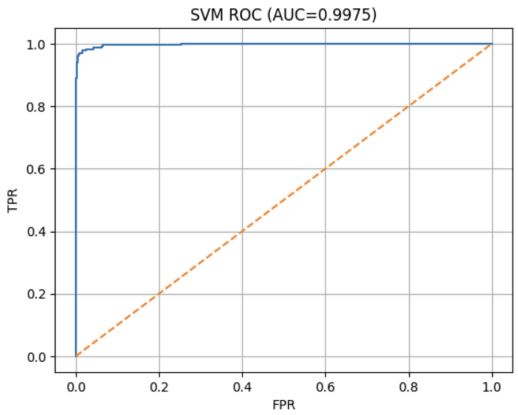




```
XGBoost 5-Fold CV F₁ macro: 0.9279 ± 0.0041

XGBoost on test set:
[[4273   11]
 [  39  196]]
              precision    recall  f1-score   support

           0     0.9910    0.9974    0.9942      4284
           1     0.9469    0.8340    0.8869       235

    accuracy                         0.9889      4519
   macro avg     0.9689    0.9157    0.9405      4519
weighted avg     0.9887    0.9889    0.9886      4519
```

SVM (F1 Macro + Precision–Recall + ROC Plot + F1-Score + Confusion Matrix)






```
SVM 5-Fold CV F₁ macro: 0.9503 ± 0.0079

SVM on test set:
[[4251   33]
 [   8  227]]
              precision    recall  f1-score   support

           0     0.9981    0.9923    0.9952      4284
           1     0.8731    0.9660    0.9172       235

    accuracy                         0.9909      4519
   macro avg     0.9356    0.9791    0.9562      4519
weighted avg     0.9916    0.9909    0.9911      4519
```

Model performance summary:

Model	Precision (class 1)	Recall (class 1)	F1 (class 1)	F1 (macro)	Average Precision	ROC AUC
Random Forest	0.915	0.46	0.612	0.798	0.827	0.984
XGBoost	0.943	0.843	0.89	0.942	0.946	0.992
SVM	0.873	0.966	0.917	0.956	0.978	0.998
Logistic Regression	0.755	0.97	0.849	0.92	0.968	0.996

Evaluation:

Confusion Matrix

Breaks down predictions into:

- TP: Fake news correctly identified
- FP: Real news wrongly flagged as fake
- TN: Real news correctly identified
- FN: Fake news missed

Helps show what kinds of errors the model makes.

- Precision: Of the news labeled fake, how many were actually fake?
- Recall: Of all actual fake news, how many did the model catch?
- High precision = fewer false alarms.
- High recall = fewer missed fakes.

F1 Score (Class 1 + Macro)

- F1 Class 1: Balance of precision and recall for fake news.
- F1 Macro: Average F1 for both real and fake.

Shows overall accuracy while handling class imbalance.

AUC ROC

- Measures how well the model separates fake from real news.
- Closer to 1 = better performance.

Interpreting the Results
Based on the evaluation results, the SVM model clearly performed the best overall. It had the highest precision, recall, F1 scores, and AUC, which means it was both accurate and consistent at detecting fake news. This is likely because SVM is good at finding the best boundary between classes, especially when using high-dimensional embeddings. XGBoost also did really well, showing a strong balance between catching fake articles and avoiding mistakes. Its boosting strategy helped it learn from previous errors and improve over time.

On the other hand, Logistic Regression had good recall but lower precision, meaning it caught most fake news but also flagged more real news by mistake. Random Forest had high precision but very low recall, so while it didn't falsely label many real articles, it missed a lot of fake ones. This shows that it struggled with the class imbalance. Overall, SVM was the most reliable, and XGBoost is a solid second choice for this kind of classification task.

5.3 Potential Extensions

- Multilingual Capability: Expand to multilingual detection using newly developed language transformer models to allow us to utilize the same technology for misinformation in different languages
- Hyper-tuning: Implement hyper-tuning to the model, such as Setfit. We tried implementing it, but it made it very confusing to utilize the Machine Learning algorithms. We also had a hard time figuring out how it works.
- Other embedding models: Experiment with other embedding models that perform well on classification tasks.

References

Ahmed, Hadeer, Issa Traore, and Sherif Saad. "Detecting Opinion Spams and Fake News Using Text Classification." *SECURITY AND PRIVACY* 1, no. 1 (2018): e9. https://doi.org/10.1002/spy2.9.