Jorge Meneu | Mukhtadir Syed | Kaumu Joshi | Zhizheng Wang

Natural Language Processing
Business Summary

In this report we provide an overview on the implementation of Factify®, a Fake News Detector. The report showcases a high-level perspective on the products evolution, from the gathering of data to the final implementation as a Flask API.

The Task

We chose the following dataset: https://www.kaggle.com/competitions/fake-news/. It consisted of a train.csv and a test.csv. Both were provided with the following features:

- id: unique id for a news article
- title: the title of a news article
- author: author of the news article
- text: the text of the article; could be incomplete
- label: a label that marks the article as potentially unreliable (only in train)
    - 1: unreliable
    - 0: reliable

Why did we choose this task?

Lies spread faster than the truth, and the same can be said about information. In the middle of this, we reject to accept this fate, and through Natural Language Processing, we proposed ourselves the development of a solution which can easily check whether a news article is reliable or not. We are aware fact-checking companies are exponentially growing nowadays, but most of them manually label the news articles. For that reason, we may find a competitive advantage with the fact of being one of the first ones to implement it.

Our project is divided into 2 parts:

- **Part A**: Traditional NLP & Machine Learning Classifiers
- **Part B**: Neural Networks with LSTM & Transfer learning using Huggingface® Transformers.

Part A

After an extensive Exploratory Data Analysis of our dataset, we leveraged on the NLTK library to perform certain preprocessing steps:

- Removing stop words using nLTK library

- Applying stemming
- Removed punctuations
- Converting all text to lowercase

The above steps lead to preparation of our corpus and then we applied TF-IDF on this corpus and our data was ready for applying traditional machine learning classifiers. The following classifiers were then used:

- MultinomialNB Algorithm
- Passive Aggressive Classifier
- MaxEnt Classifier (Logistic Regression)

The same transformations were also done on the test set. All the models gave us a high accuracy being **Passive Aggressive classifier** the top performer of the set.

Part B

In this part, we first explored **LSTM architecture**. The same corpus as prepared in Part A was used over here. However, for LSTM, there are two additional pre- processing steps

1. Applying one hot representation
2. Word Embeddings

After applying the above, our data was ready to be fed into the LSTM architecture. We experimented extensively with the neural network and came up with the following configuration:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 12059, 128)        1280000

 dropout (Dropout)           (None, 12059, 128)        0

 lstm (LSTM)                 (None, 100)               91600

 dropout_1 (Dropout)         (None, 100)               0

 flatten (Flatten)           (None, 100)               0

 dense (Dense)               (None, 128)               12928

 dropout_2 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 1,384,786
Trainable params: 1,384,786
Non-trainable params: 0
_________________________________________________________________
```

After training the neural network on our train set, we performed the transformations on the test dataset for our final submission and score on Kaggle.

Finally, we then moved onto the latest and greatest technique in NLP i.e., **transformers and transfer learning.**

The preprocessing steps for this architecture were as follows:

1. Created the train and validation set
2. Tokenize the dataset as per the model used
3. Converting the dataset to torch dataset, we created a helper function to accomplish this.

We then defined the training parameters to fine tune the model on our dataset. **We managed to implement several models, all variations from BERT and RoBERTa. Due to computational and timely constraints, we implemented distilled versions of both. After some fine tuning, we chose our best performing one: distilbert.**

Afterwards, we tested our model and finally, run the predictions to submit onto Kaggle.

From all the techniques used, our best score on Kaggle was achieved through transformers and transfer learning, truly showcasing the power of these state-of-the-art techniques.

Part A Traditional NLP			**Part B Neural Networks**	
MultinomialNB	**Passive Aggressive Classifier**	**MaxEnt Classifier**	**LSTM**	**Transformers**
0.911	0.955	0.911	0.926	0.974

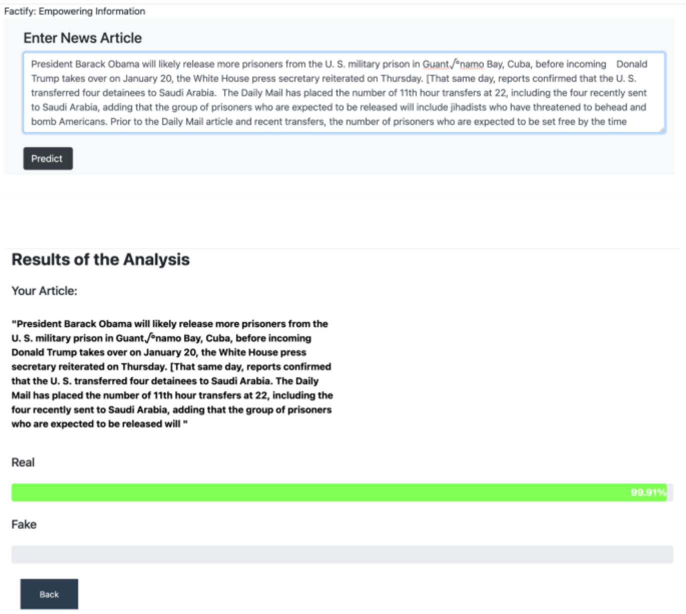
Flask App

To bring a more user-friendly interface for readers, we deployed our model as a Flask API. The developed API provides an interface for users to submit news articles and receive a prediction of whether the article is real or fake. The app consists of two main parts: the front-end and the back-end.

- Front-end: simple HTML interface for the user to interact with, to insert news articles.
- Back-end: in charge of predicting the reliability of the news article.

When the user submits a news article, the back-end receives the text, passes it through the model, and returns the predicted label. We have used Python and Flask to create the back-end of the app.

Finally, to further enhance the app, we can consider adding some additional features, such as a history of past predictions, a leaderboard of top contributors, or a feedback system for users to report incorrect predictions or provide feedback on the app's performance.



An overview on the interface of Factify

Conclusion and Next Steps

Our project demonstrates the potential of NLP in detecting fake news, which is an increasingly pressing problem in today's world. By using state-of-the-art models like transformers, we were able to achieve a high level of accuracy in detecting fake news, which can help in building more reliable systems for fact-checking.

In terms of next steps, we would like to deploy our web app as a browser extension. A browser extension can make it easier for people to verify news articles before sharing them on social media or other platforms. It can also make the process more accessible to a broader audience.

In building the browser extension, we will need to consider the user interface, the accuracy of the model, and the scalability of the system. We may also want to explore other NLP techniques or models that can improve the accuracy of our detection system further. There

are some of these Chrome® extensions already in the market such as TrustServista®, FakerFact® and Stopaganda® plus to name a few. Comparing our model with existing apps and browser extensions that are already live is also a crucial next step to take.
Overall, our project has demonstrated the potential of NLP in detecting fake news, and our proposed next steps are a promising direction for future development.