Fake News Detection

CS 579 Online Social Network Analysis

4/27/2022

Bharath Bandaru

Himamshu Lakkaraju

1. Abstract:

Fake news Detection has been an active research area where social media companies and news organizations are trying to mitigate the spread of fake news. There are several bot accounts and organizations that are trying to influence social behavior and people's perception of topics of their interest either for profits or ideologies or political affiliations etc. It is a very difficult task to detect these fake news articles as natural language understanding is a difficult task and there are several other factors to consider like the account or user posting the articles, previous history of posting fake news etc. In this project we try to create models that are trained on titles obtained from User A and User B and label it with one the classes agree/disagree/unrelated.

2. Introduction:

With the advent of personal computing and the internet many social media companies that allow users to share content/articles and views came into existence. These platforms grew tremendously in the last decade. Companies like facebook, twitter, reddit etc have all become a common place for users to engage and share their views. These platforms also started acting as a major news source for users. There are entities that try to take

advantage of these platforms to spread their personal agendas or goals. There have been a tremendous amount of bots crawling these platforms and engaging with genuine users trying to change the perception of a story or plant ideas etc. Identifying fake news has been a major challenge for all these platforms. Natural language understanding is a difficult task and manually trying to go through the content to remove fake news is not possible with the amount of data generated todas. ML algorithms are being used to identify and flag these articles/posts and these ML algorithms are trained based on various parameters like the user history, past violations etc. The content is matched against a known or verified source. Number of users reporting the content etc. In this project we have a dataset from twitter with titles from user A and user B and their true label. This is used to train a model that can detect if the tweets are agreeing, disagreeing or if they are unrelated. There are several ways to approach a problem like this. There are statistical methods that can be used to find parameters that fit on the training data and the assumption is the test data is similar to the train data. There are several issues with a statistical model like generalization of the model. A statistical model works well only if the out of the sample data is similar to the data that the model is trained on, which might not always be the case in the real world. Neural Networks are built to solve such problems but they have their own challenges like the model size, hardware and data requirements to train such models etc.

3. Proposed Solution:

We plan to train a simple Logistic Regression model that classifies the text to one of the 3 labels. We also plan to create two neural networks, which use the LSTM model which is very good in identifying patterns in sequences and we also train a LSTM model with siamese architecture that finds similarity between the inputs and labels the data accordingly.

4. Implementation:

The dataset is visualized to identify the class distribution of the data.
The training file used has the class distribution as below:
Unrelated - 68.475%
Agreed - 28.949%
Disagreed - 2.576%

As it can be seen from the above distribution of the data, we can see that the distribution is highly unbalanced and any model trained on this data might not identify the disagreed class accurately as the available training data is very limited.

The data has to be balanced and there are several ways to do that. We can try to get more data from the same source with different classes to append to our initial data or generate new data from the existing data or perform resampling on the available data.

The approach we chose is to generate data from the existing data. There are several techniques to do this, some of them include Text generation using models like GPT-2 or BERT etc. These are trained models available from a repo like huggingface that can be imported and used to generate similar text. We can use ML models to translate the data to a different language and translate it back using another ML model and this will result in change in structure and words of the data while preserving the meaning of the sentence. We can use a module like nlpaug which has a function that can replace words in a sentence with synonyms that can be used to generate similar sentences with the same meaning but different words.

We chose to do the data augmentation using the nepaug module and generate new data by replacing at least 2 words in a given sentence with a maximum of 4 words replaced in the sentence.

The class distribution of the data set after the data augmentation is as below:
Unrelated - 40.240%

Agreed - 34.0239%

Disagreed - 25.734%

This is much better than the initial training dataset and this is used to train the models

Model 1 Logistic Regression:

The training data is split into a train test split using 70% to train and 30% for validation. We use TF-IDF vectorizer to transform all the combined text of 2 columns with a minimum document frequency of 20.

This is passed to the Logistic Regression model as input with the labels passed as the true outputs.

The test and validation accuracy are very high. The F1 scores for training and validation dataset from the model is as below:

Training data:

	precision	recall	f1-score	support
0	0.85	0.87	0.86	122918
1	0.86	0.85	0.85	103919
2	0.96	0.93	0.95	78601
accuracy			0.88	305438
macro avg	0.89	0.88	0.89	305438
weighted avg	0.88	0.88	0.88	305438

Validation data:

	precision	recall	f1-score	support
0	0.82	0.84	0.83	52680
1	0.82	0.82	0.82	44537
2	0.95	0.92	0.94	33686
accuracy			0.85	130903
macro avg	0.87	0.86	0.86	130903
weighted avg	0.85	0.85	0.85	130903

As we can see from the above, the model fits the data well and has a validation accuracy of 85% with good recall and precision.

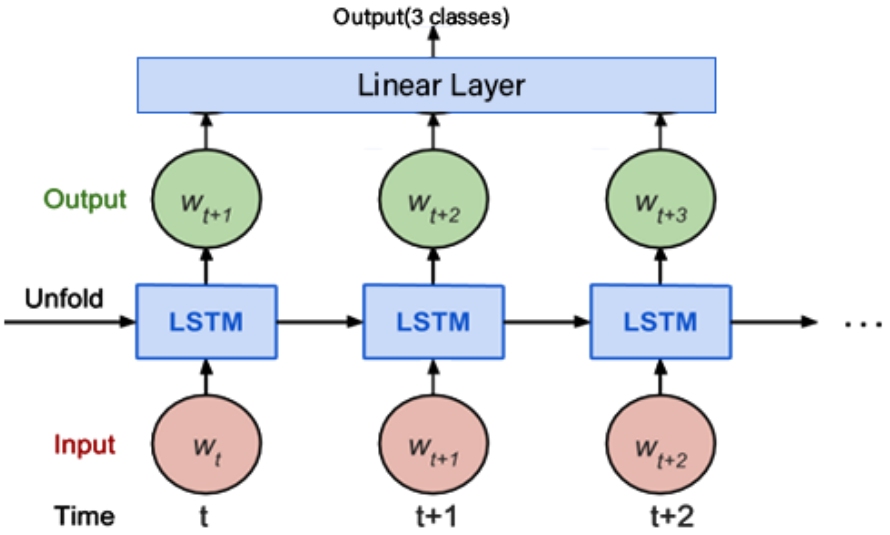
We wanted to check if a neural network can provide better results and hence we created two neural networks that use LSTM layers to label the inputs.

Simple LSTM model:

The two titles are concatenated and encoded as neural networks only take numbers as inputs and provide numbers as outputs. Vocabulary of the words is generated and the word count is used to encode the data.

These encoded combined titles are passed to a bidirectional LSTM layer as input and the outputs are passed through a linear layer that outputs the 3 class probabilities of the input text. A categorical cross entropy loss function is used to calculate the loss used for backpropagation. A stochastic gradient descent is used as an optimizer with a learning rate 1e-3.

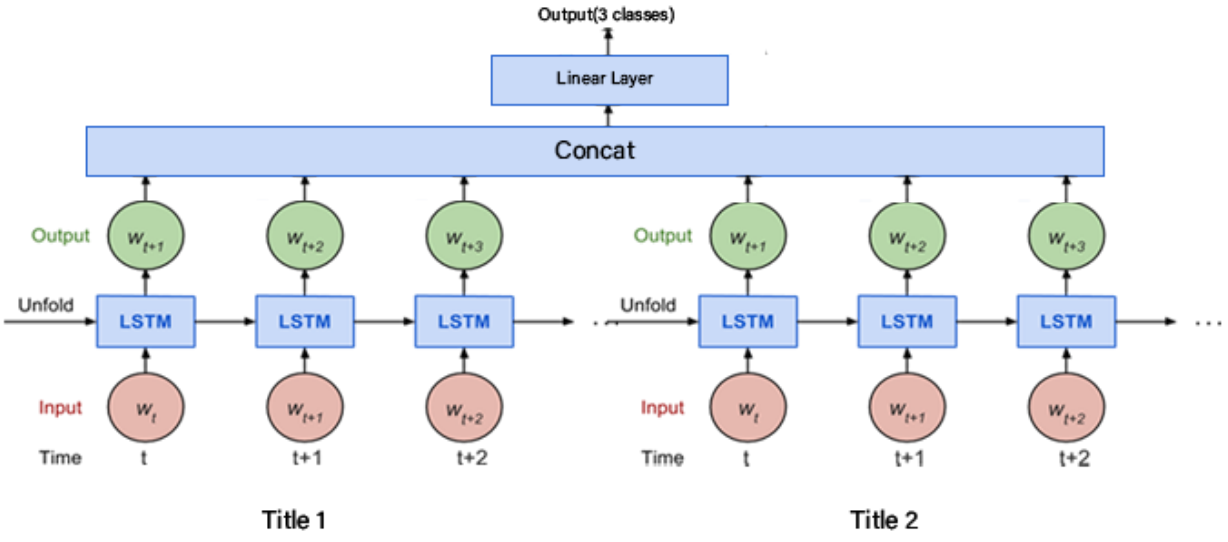
LSTM network architecture:



LSTM Network with Siamese architecture:

We created a similar network that takes 2 encoded titles separately as inputs and learn the similarities in the text to generate output labels.

Siamese architecture of LSTM network:



5. Results, Observations and issues:

These neural networks require more computational power compared to the Logistic Regression models created. These models require CUDA memory to train faster which was a limiting factor for us to train these models.

We were limited to using small batch sizes and could not run as many epochs as we'd wanted to.

After one epoch each of the above models provided a test accuracy of around 40 to 65%.

We couldn't retrain these models often with hyperparameter tuning as they take a long time to go through each epoch.

6. Conclusion and Future work:

We'd like to increase the batch size and number of epochs to train the above neural nets.

We'd also like to test our logistic regression model with data that's not similar to the training dataset and see how well it holds up with new data.

7. Acknowledgement:

This project is done as part of the course project for CS-579-01: Online Social Network Analysis - Spring 2022.

8. Libraries and tools used:

1. Python 3.8.12
2. PyTorch
3. Pandas
4. Numpy
5. Scikit-learn
6. VS Code
7. NLPAUG
8. Seaborn
9. Pyplot