Default Project: FakeNews Detection

Arushi Rai [A20396974]
arai4@hawk.iit.edu
Aakef Waris [A20420535]
awaris@hawk.iit.edu
Illinois Institute of Technology
Chicago, IL

Abstract

In this project we investigated multiple models to article pairs to classify agreement, disagreement, and unrelated. We first tried Logistic Regression and RandomForest on GoogleNews pretrained word2vec embeddings. Ultimately out of the neural network approaches MLP A on SentenceBERT performs the best.

1 Introduction

One of the greatest problems of our time is culling the spread of misinformation, specifically in the form of fake news. Often times, stories are picked up by larger outlets despite the original news originating from an unknown blog source. This project's goal is to detect whether two news sources are agreeing or disagreeing, or if they are unrelated. Potentially, this could track the spread of misinformation given a repository of fake news.

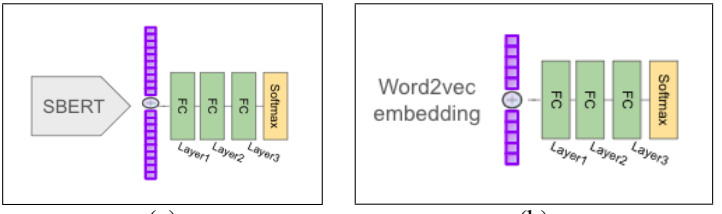
The format of the dataset include two text titles, with the first being fake news. The class label a categorical data type and is from the following set: "agree", "disagree", "unrelated". This class label is converted into numerical form and then a one-hot vector.

2 Preprocessing and Feature Engineering

We were given two csv files, with the test.csv file containing no labels. This meant to compare the models, we needed a holdout set. We did an 80:20 split on the train data to get a smaller train set and our own test set.

When analyzing the data, we noticed there was great imbalance in the class labels, with it being skewed towards unrelated samples. While one trivial approach is to sample across labels equally, we would lose a lot of data. To combat the flaws of this approach we tried upsampling. However, for our Kaggle submission we didn't include approaches using those models since they performed poorly because the test data on Kaggle is also imbalanced.

label	number of instances
unrelated	175598
agreed	74238
disagreed	6606



(a) (b)

Figure 1: Two Multilayer Perceptrons (MLP) are used: (a) MLP A; (b) MLP B;

Using the NLTK package [1] we tokenized the titles and removed stopwords. For the labels we used LabelEncoder to convert nominal data into a numerical form. Then using Keras [2] we converted into one-hot encoding. We tried two word2vec [4] models and the SentenceBERT model [5] to extract the title embeddings. The first step is to preprocess the text data so that stopwords (frequent words) and punctuations are removed. Then the title is tokenized and the tokens in the word2vec vocabulary for each respective word2vec model is turned into a [# of tokens]x300 vector. This multidimensional vector is average pooled into a 1x300 vector. We also trained word2vec on our corpus because many words were not in the GoogleNews vocabulary. One explanation for this domain gap is that the GoogleNews embeddings were trained on largely reputable sources whereas our dataset might be from tabloids or blogs where highly sensationalized language maybe used.

We turned to SentenceBERT [5] to get better embeddings. The issue with word2vec is that we are average pooling to get the title embedding which loses out on sequential or semantic knowledge present in the structure of a sentence. Look in the Experiments section to see how SentenceBERT embeddings compares to average pooled word2vec embeddings.

3 Model Description

We tried variations of an MLP. MLP A took in a concatenated embeddings from the two titles as a 1x1578 vector. MLP B concatenated the title embeddings into a 1x600 vector. Both MLP contained 3 fully-connected hidden layers with dropout and batch normalization after every FC layer. Both of these models were trained using a categorical cross entropy loss. We also tried classical data mining methods such as fitting a Random Forest and Logisitc Regression Models on the Google News word2vec embedding.

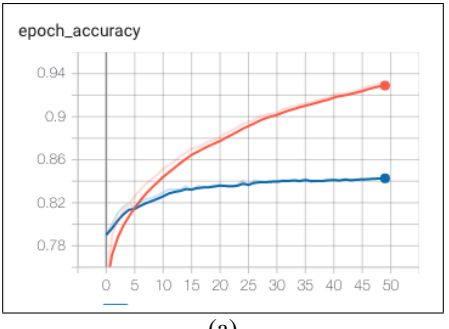
4 Experiments

Take a look at Table 1 displaying the performance of MLP A against the other architectures. We used the accuracy to compare against the other models but. Figure 2 depicts the the precision and recall which are approximately the same for MLP A with ReLU.

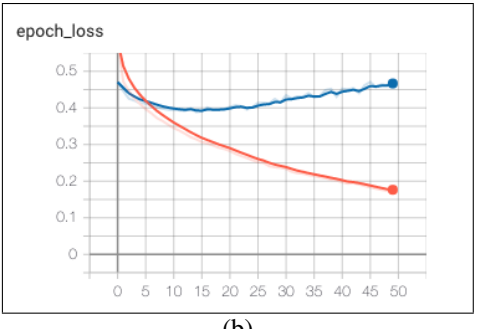
To investigate to the performance of SentenceBERT embeddings we created box plots shown in Figure 3. The agreed labels have a higher mean cosine similarity than the other models. However, the variation is pretty high and a threshold would not work. Usually for measures like cosine similarity we expect to use it to directly compare and get an idea regarding the relation however it seems like we cannot do that with SentenceBERT.

Model Architecture	Embedding	Accuracy
MLP A with LeakyReLU	**SentenceBERT**	**85.6**
MLP A with ReLU	SentenceBERT	84.6
MLP B with ReLU	word2vec	77.43
MLP B with ReLU	Google News word2vec	76.0
Random Forest	Google News word2vec	71.1
Logistic Regression	Google News word2vec	68.3

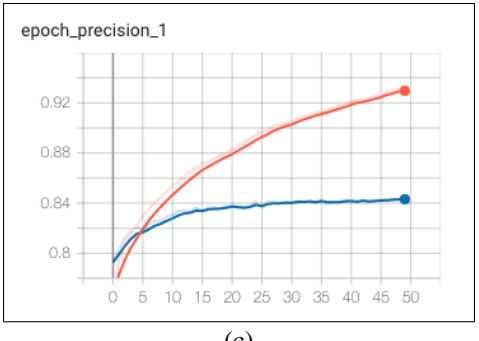
Table 1: MLP A with LeakyReLU outperforms the other architectures.



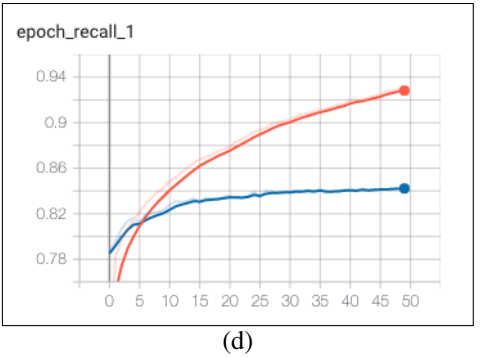
(a)



(b)

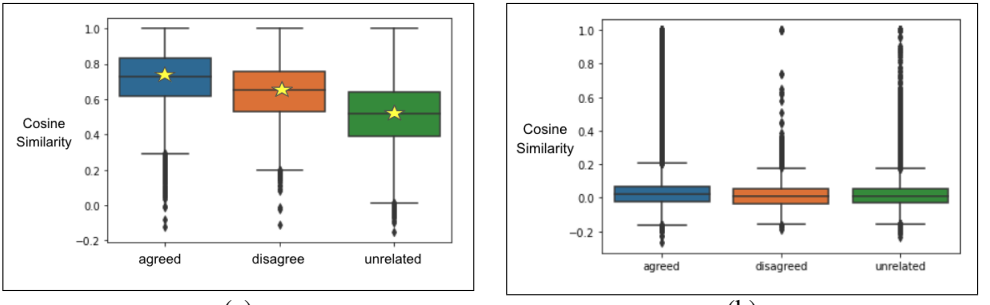


(c)



(d)

Figure 2: MLP A with ReLU (blue validation: (a) accuracy vs epoch; (b) loss vs epoch; (c) precision vs epoch; (d) recall vs epoch. All of these figures indicate the around the 20th epoch the model stops improving the validation metrics. The validation loss starts increasing, which indicates that gradient updates are not generalizing to the validation set.



(a) (b)

Figure 3: Cosine Similarity Distribution of: (a) SentenceBERT; (b) GoogleNews Word2Vec.

5 Conclusion

SentenceBERT performed better than Word2Vec as the embedding. We should also try an RNN-based model using the word2vec embeddings to get hierarchical representation. SentenceBERT has room for improvement, perhaps we could try a supervised contrastive learning [3] approach on BERT. This method explicitly trains the embeddings to be further apart. With labels we know the correct positive and negative samples instead of randomly sampling the negative samples. Random Forest Model Performed better than Logistic Regression as Logistic Regression does not perform well on imbalanced data. One key method for embeddings is tf-idf, so we should try running our classical machine learning models and neural nets on that vectorization.

6 Group Justification

6.1 Arushi

Created MLP A and B.
Preprocessing and feature engineering.

6.2 Aakef

Created Random Forest Model on Google News word2vec embeddings
Created Logisitc Regression Model on Google News word2vec embeddings

7 Source Code

Link to Code: https://github.com/awaris123/OSNA-Project2 Be sure to look at the README to know which files to run.

References

[1] Steven Bird and Edward Loper. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona,

Spain, July 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P04-3031.

[2] François Chollet. keras. https://github.com/fchollet/keras, 2015.

[3] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2020.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[5] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

7.1 Tutorials

Data Generator https://stanford.edu/ shervine/blog/keras-how-to-generate-data-on-the-fly

7.2 Libraries and Packages

Keras
Tensorflow
Scikit-learn
Numpy
matplotlib