Course Project Report

Group#2

1. Introduction

Fake news and other misinformation do harm society and need to be detected by effective methods. In this project, we implement Naive Bayes (NB), Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) to classify a bunch of claims provided by DataCup contest “Leaders Prize: Fact or Fake News?”, to detect whether the given claim is True, partly True or False. Then we can get the model that is more capable of detecting fake news.

2. Data description

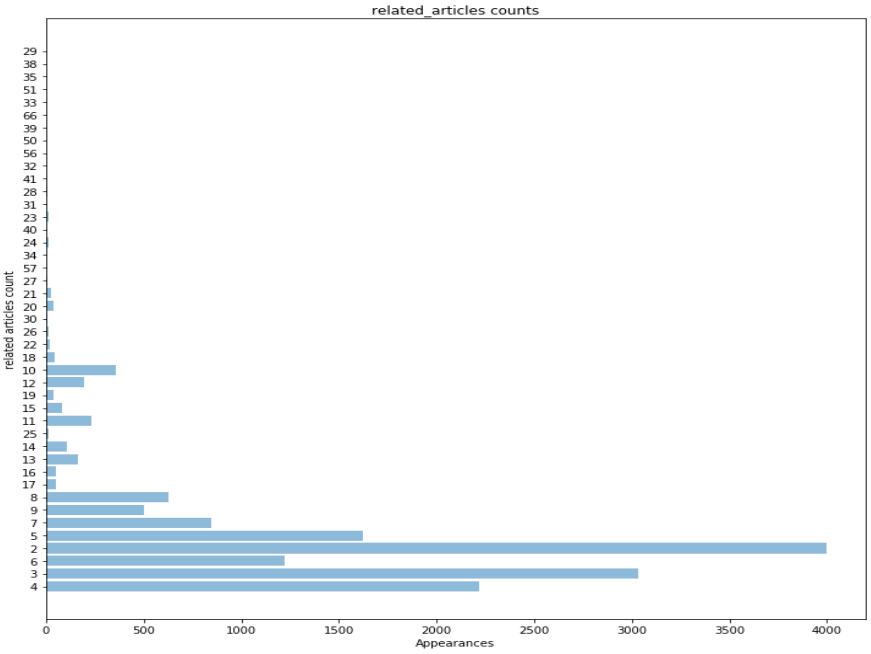
claim	claimant	date	label	related_articles	id
Says U.S. Sen. Ron Johnson "doesn't even belie...	Russ Feingold	2016-10-06	1	[1088, 26723]	4107
“By denying climate change, [Stephen Harper] d...	Justin Trudeau	2015-07-09	0	[97541, 97901, 98022, 100860]	14091
Says President Donald Trump "has signed more l...	Mike Pence	2017-07-18	1	[6007, 28240, 28245, 88386, 47172]	4210
"Obama's Private 'Security' Company Sets Up M...	Various websites	2018-04-30	0	[26939, 37849]	11405
“I was against the war in Iraq. Has not been d...	Donald Trump	2016-10-09	0	[58855, 57620, 58877, 58858, 76663, 58736, 588...	5347

2-1 Parts of data in metadata file

The data set provided by DataCup contains one metadata file and one article directory: The metadata file contains thousands of claims, which are the key components that need to be detected as true or false. Each claim has its detailed information like claimant, data, id, label (true - 2, partly true - 1, false - 0) and the id of related articles that generate or support it. The article directory contains a large number of articles, and each article has its id as its filename. We can easily connect each claim with its related article by the article id in each claim’s information and the filename of each article.

After analyzing the data, we found the claims’ detailed information in the metadata file was appropriately filled and no NAN entry was found.

The distribution of the number of related articles is plotted as figure 2-2. From that we can know that most of the claims are generated or supported by four to eight articles.



2-2 Distribution of the number of related articles

3. Preprocessing

Based on the theory of Machine Learning, we choose the label (true, partly true, false) as our target for classification, and choose claims and their related articles as our features.

For NB, we use the *Bag of Words* model to convert sentences (claims) into vectors. In Natural Language Processing, this model can be used to extract features from the text. The first step is to generate a vocabulary, which is a list of distinct words extracted from our claims. Then each claim is vectorized based on the number of times that each term occurs in the claim. Sometimes the vocabulary size is too large to train the model so we can set a maximum number of features ordered by term frequency across the vocabulary. The output for this model is a numerical vector representing each sentence. Then each claim has its words as features and has its label as the training target.

For LSTM, we use *Word Tokenizer* to tokenize sentences into a list of indexes of its words. We use two kinds of training sets here. The features of the first training set are the tokens of claims, and the features of the other training set are the tokens of the claims and the tokens of the relative articles. Since one claim may have more than one related article, we separate the related articles into multiple rows of training data and make the tokens of one claim and one of its related articles form one row of features. Then we use *Word Embedding* to convert words into vectors and generate word embedding (weight) matrix as the input of the neural network. In this project, we would use the pre-trained embedding from Glove.

For CNN, we use word2vec to train words into vectors, and we use the resulting weights as the initialization weights of the embedding vector. Then we applied the CNN model with ten filters, sized (3, 8), 50 embedding dimensions, 50 hidden dimensions, and three output dimensions.

4. Model implementation

4.1 Naive Bayes

The Naive Bayes classifier is a probabilistic classifier. It is implemented to find the best class for each claim based on the maximum likelihood. However, Naive Bayes is a generative model which models the joint distribution of feature X and target Y to predict the P(y|x). It assumes all features to be conditionally independent, which means these features don't have a correlation with each other, and each feature is corresponding to one target.

We use MultinomialNB from the sklearn library to do classification because the features are integer counts, generated by the *Bag of Word*s model and we use the accuracy result as our baseline for this project.

4.2 LSTM

Long Short-Term Memory networks (LSTMs) are special kinds of Recurrent Neural Networks (RNN) and was proposed by Hochreiter and Schmidhube as early as 1997. Compared with ordinary neural networks, the network can remember long term memories, which has proven to be essentially useful in speech recognition and sentimental analysis tasks. Compared with other existing RNN networks, it has three gates: input, output and forgets. Considering the nature of this project involves text recognition, LSTM is a natural option.

We use the LSTM model with four different implementations: General LSTM, Bi-directional LSTM, LSTM with attention, and LSTM with attention supplied with extra feature related articles.

We implement each of them with two layers of LSTM in the neural network, and each LSTM layer has 256-unit size.

LSTM model	Accuracy
General LSTM	58.76%
Bi-directional LSTM*	55.87%
LSTM with attention**	59.14%
LSTM with attention, supplied with extra feature related articles***	57.13%

4-2-1 Different models of LSTM and their accuracy on test set

*Bidirectional LSTM is an LSTM with both directions; in other words, the states in the neural network have access to both "past" and "future" information. This kind of model with the same parameters as the normal LSTM has been implemented as well.

**During the years of implementation and discussion, gradient explosion and vanishing might happen with very long sentences, since we are multiplying and processing the gradients too many times. To resolve this matter, attention modules have been designed. In traditional LSTMs or RNNs, the network has been divided into "encoding" and "decoding" process, where the decoding process will only deal with the last context vector (or "thought" vector) at the end of the encoding process. This would cause the mentioned gradient explosion or vanishing since it would be very likely that the information of the previous text would be lost along the way to generate the thought vector. Attention modules are designed such that all the intermediate states would be taken into consideration and have an impact on the decoding process.

***Apart from pure claim text analysis, it would also be valuable to take the related articles into account. Checking reference sources is also a reliable way to tell if a piece of news is valid for humans. Hence a deep neural network that takes the related article's text into consideration has been implemented as well. We average the prediction distribution result with that of the previous claim LSTM and derives the final result.

4.3 CNN

CNN uses convolution in place of general matrix multiplication. Although CNN is not widely used in natural language processing, it is powerful when handling complex classification problems. Based on the information provided in Yoon Kim's paper, our group decided to implement a CNN classifier to compare its results with those of the LSTM classifiers mentioned below. As Kim's paper suggested, we also applied an embedding layer taking advantage of the Word2vec function.

Compared with other classification methods, CNN has a lot of advantages. First of all, in each mapping of convolutional layers, the model considers the information in a small neighborhood, usually a 3×3 or 5×5 matrix, moreover, since the mappings in the convolutional layer are many-to-one mappings, so they reduce the number of units in the network efficiently, that means we need fewer parameters to learn and so that the probability of overfitting would be reduced.

Our CNN method includes a feature extraction part and a classification part. The word2vec function is used to train words into vectors, and the model is built on Keras library, adapted from the "CNN-for-Sentence-Classification-in Keras" by Alexander Rakhlin, based on the algorithm provided in "Convolutional Neural Networks for Sentence Classification" by Yoon Kim. Finally, we get a validation accuracy of 60.67% after five epochs. But we notice that our CNN model would not classify any news as "True news". This might be caused by the lack of "True news" samples in the provided dataset, making related information hard to survive when passing the convolutional layer.

5. Findings and Analysis

In sum, we have implemented in total three different models for this specific project:
1) Naive Bayes 2) LSTM and 3) CNN, where we have used the Naive Bayes model with *Bag of Words* vectorization as a benchmark of our implementation.

Our NB model has a training accuracy of almost 75%. However, the testing accuracy is just 57.9%, revealing a significant variance.

In our LSTM realization, we have compared four different LSTM models and found that the LSTM with attention model performs best on the testing data set, having an accuracy of 59.1%.

Finally, we investigated a novel CNN approach with word embeddings. The model acted fair during training and has a 60.7% validation accuracy. Nevertheless, the model cannot label True News; thus, its performance on a larger data set might be worse.

Compared without the NB benchmark, both the LSTM and CNN model behave better. After evaluation, we decided to use the LSTM (with attention) model in our final submission and got score 0.408579 on the contest platform.

29	paddle	0.408579

5-1 Score on leaderboard

6. References

How Naive Bayes Algorithm Works. (2018, November 9). Retrieved from https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/

Brownlee, J. (2019, August 12). Naive Bayes for Machine Learning. Retrieved from https://machinelearningmastery.com/naive-bayes-for-machine-learning/.

Understanding LSTM Networks. (2015, August 27). Retrieved from https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

An intuitive guide to Convolutional Neural Networks. (2018, April 24). Retrieved from https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/.

Convolutional Neural Networks (CNNs / ConvNets). (n.d.). Retrieved from http://cs231n.github.io/convolutional-networks/.