CMPT 310 NLP Project

Simon Fraser University

Group member:
Jianing Sun, jianings
Xin Bei, xbei
Ming Chen, mca176
Lai Wei, laiw
Yingwen Ren, yingwenr
Jiaran Yu, jiarany

November 2018

1 Abstract

The goal of this project is to build a program for a natural language processing (NLP) task. The problem is to predict the validity of a news article, that is, whether the news is real or fake. Two approaches are applied in this project to solve the classification problem: neural net learning and decision tree learning. We experimented with different options to increase accuracy (cross-validation score) and insight. The details are introduced in the following sections.

2 Dataset

The dataset contains 1500 real and 1500 fake cases. Real and fake articles are stored in two separate directories called “real” and “fake”. The type of the data files is all plain text. Since the testing data are not provided, we split our dataset into training set and validation set and evaluated the model using 10-fold cross-validation.

3 Neural network learning

3.1 Approaches

The neural network learning method is written in python. We used Natural Language Took Kit (NLTK) for data pre-processing, and we built the neural network models with TensorFlow Keras.

The first neural network we built is Word2Vec Continuous Bag-of-Words model (CBOW), which is an NN-based word embedding method. It is able to represent words in a continuous, low dimensional vector space (i.e., the embedding space) where semantically similar words are mapped to nearby points [1] . The main idea of word2vec is to uses an NN with only 1 linear hidden layer to use a word to predict its neighbors, as shown below:

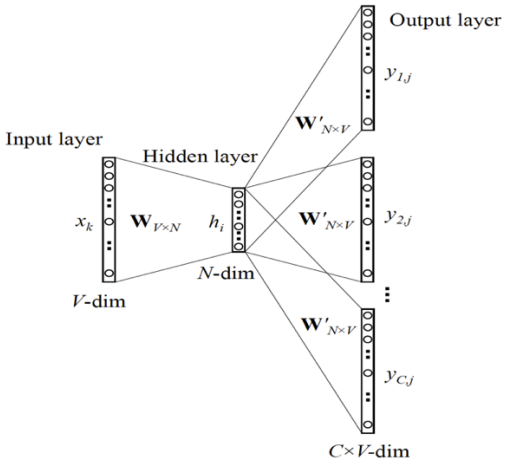


Figure 1: the structure of CBOW model

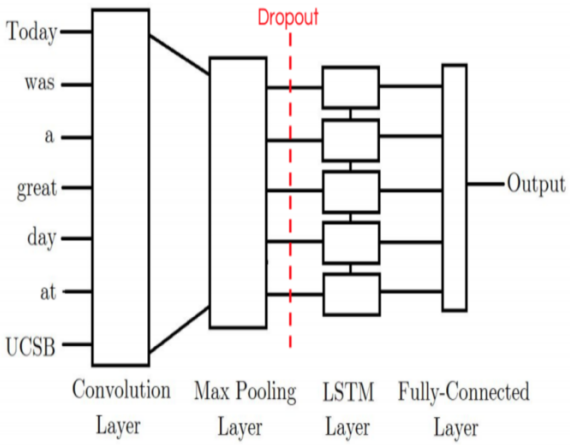


Figure 2: the structure of CNN-LSTM model

We have also implemented a more advanced model, the Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) Neural Network. Our CNN-LSTM model combination consists of an initial convolution layer which will receive word embeddings as input. Its output will then be pooled to a smaller dimension which is then fed into an LSTM layer. The intuition behind this model is that the convolution layer will extract local features and the LSTM layer will then be able to use the ordering of said features to learn about the input's text ordering [2]. The overview of CNN-LSTM model is shown as figure 2 above: After the model implementation, we explored some options to improve the cross-validation score.

3.2 Exploration of Options

We experimented with two different models and compare their results to see which one gives a better result. As the optimizer for our neural network models is Adaptive Moment Estimation (Adam), which is one kind of improved stochastic gradient descent, the 10-fold cross-validation score would increase as the number of epochs increases as shown below:

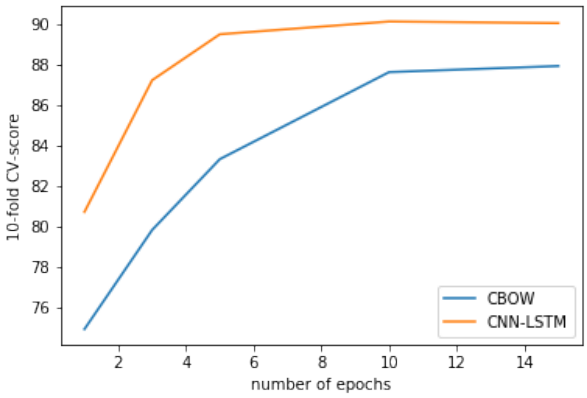


Figure 3: cross validation score vs. input length

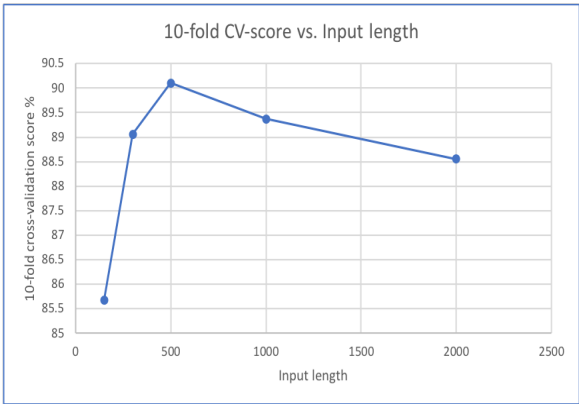


Figure 4: cross validation score vs. input length with CNN-LSTM

It is clear to see that CNN-LSTM model does way better than CBOW model, to get a better 10-fold cross validation score, we should choose CNN-LSTM.As the CNN-LSTM model performs better than CBOW model, we kept experimenting with it. Figure 4 shows that as the input length increases, the score of CNN-LSTM increase at first and then decreases. We also tried different number of nodes in the convolutional layers. As the figure shown below, the 10-fold cross validation score actually increases as the number of nodes in convolutional layers increases.

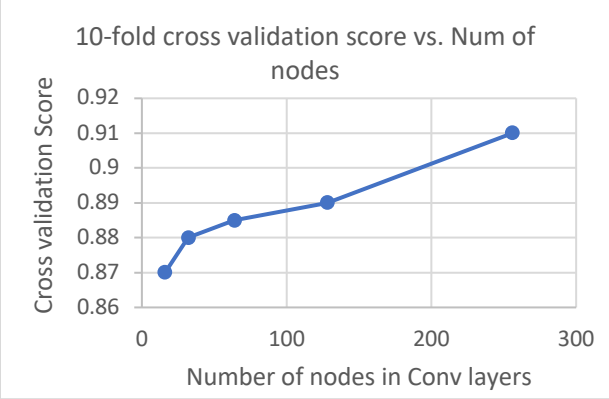


Figure 5: cross validation score vs. number of nodes.

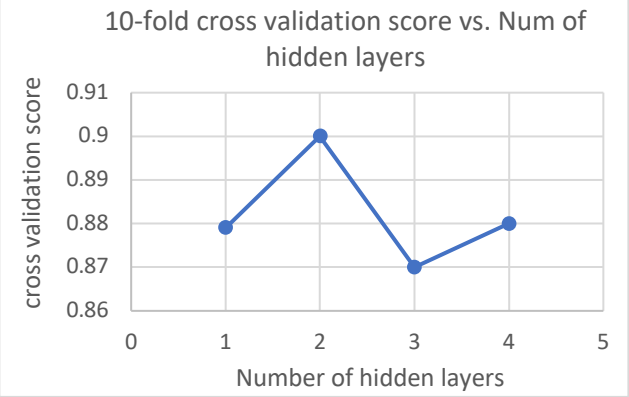


Figure 5: cross validation score vs. number of hidden layers

Another option we have explored is changing the number of hidden convolutional layers. As the plot shown above, two hidden layers give the model the best 10-fold cross validation score, and the model performance does not necessarily depend on the number of convolutional layers.

3.3 Insight

Decision Trees should be faster once trained. This is because a decision tree inherently "throws away" the input features that it doesn't find useful, whereas a neural net will use them all unless you do some feature selection as a pre-processing step. If it is important to understand what the model is doing, the trees are very interpretable. As to the Neural Nets, it is comparatively slower and less interpretable. But it can model more arbitrary functions (nonlinear interactions, etc.) and therefore might be more accurate, provided there is enough training data.

4 Decision tree learning

4.1 Approaches

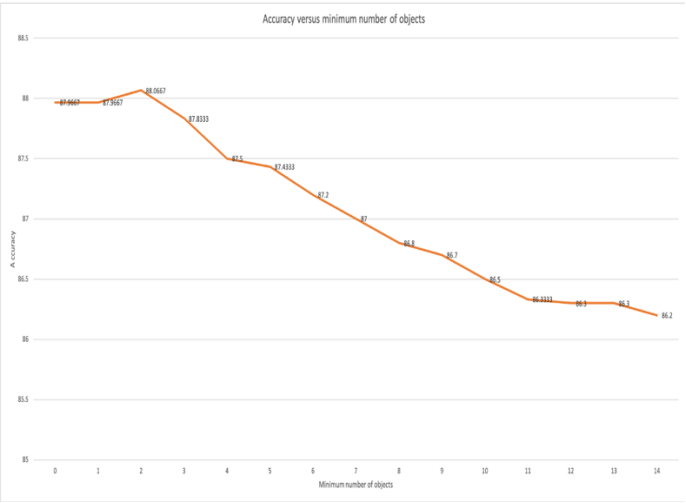
In this project, we used Waikato Environment for Knowledge Analysis (Weka) for decision tree learning. Preprocessing data and training decision tree algorithm were involved in out project.

We first applied StringToWordVector filter to transform each text into a n-dimensional boolean vector form, where each element represents the occurrence of a word. A commonly used 630-stopwords were removed in the preprocessing, in order to eliminate frequent usage words that are useless for text classification, such as conjunctions and prepositions. We used snowball stemmer for reducing inflected words to their root form. Attribution selection is also applied here to eliminate the poorly characterizing characters.

J48 classifier was adopted in our experiment, we aimed to get the result that does a decision tree perform better if it is pruned. One way is to impose a minimum on the number of training examples that reach a leaf. We tested the J48 classifier with 0.1 confidence factor and set the range of minNumObj from 1 to 14. Confidence factor was also considered in our experiment, ranging from 0.05 to 0.7.
We also attempted to delete some irreverent words from input, and drop low frequent words based on different values.

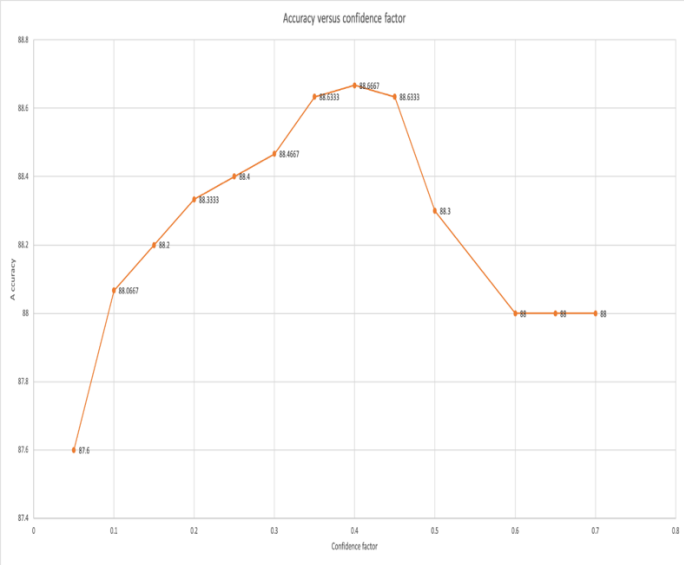
4.2 Exploration of options

We tried to explore many options to improve the cross-validation score. The significant ones are to control the size of decision tree by controlling minimum number of objects and confidence factor.



Minimum #objects	#leaves	size of the tree	Accuracy
1	62	123	87.9667%
2	53	105	88.0667%
3	47	93	87.8333%
4	44	87	87.5%
5	42	83	87.4333%
6	37	73	87.2%
7	33	65	87%
8	29	57	86.8%
9	26	51	86.7%
10	26	51	86.5%
11	26	51	86.3333%
12	23	45	86.3%
13	20	39	86.3%
14	19	37	86.1%

The graph above shows the relationship between accuracy and minimum number of objects. We can see that the accuracy reaches the highest value 88.6667% when minimum number of objects is 2 and it keeps decreasing as minimum number of objects increases.



Confidence	#leaves	size of the tree	Accuracy
0.05	47	93	87.6%
0.1	53	105	88.0667%
0.15	53	105	88.2%
0.2	63	125	88.3333%
0.25	63	125	88.4%
0.3	79	157	88.4667%
0.35	78	155	88.6333%
0.4	87	173	88.6667%
0.45	91	181	88.6333%
0.5	94	187	88.3%
0.55	118	235	88%
0.6	118	235	88%
0.65	118	235	88%
0.7	118	235	88%

The graph above shows the relationship between accuracy and confidence factor. We can see that the accuracy reaches the highest value 88.6667% when confidence factor is 0.4. After that, the accuracy decreases as the confidence factor increases and it becomes stable at 88% when confidence factor is greater than 0.6.

In further attempts, we removed "//www" from all the attributes. It turned into higher accuracy of 88.7333%
We also experimented with other options such as error pruning, however they actually decreased the accuracy.

4.3 Insights

The preprocessing step in the sentiment analysis is critical for building model, sometimes it is better not removing the stopwords. Without removing stopwords, the accuracy increased to 93%, this is because commonly used stopwords might have valuable information, if take them out, the meaning of the sentences might be changed. We also tried another classifier name LMT, it leads a higher accuracy (91%) compared with J48. It should be noticed that overfitting could be a problem for both decision tree and neural net model.

5 References

[1] Eliot Barril. Kaggle: Text Classification using Neural Networks, 2018

[2] Pedro M Sosa. Twitter Sentiment Analysis using combined LSTM-CNN Models, 2018