Political Ideology Detection with Textual Data

Makoto Kinoshita
Colby College
Waterville, ME
Email: mkinoshi@colby.edu

Tatsuya Yokota
Colby College
Waterville, ME
Email: tyokota@colby.edu

***Abstract*—In this paper, we examine various methods to detect political ideology from textual data. Out of the models that we have explored, the optimized k-means method performs the best, surpassing results presented by previous papers on the same dataset. Our research reveals the possibility of traditional machine learning methods, and the difficulty of constructing an appropriate neural network systems.**

I. Introduction/Motivation

With the growth of the internet, many news sources have also started publishing articles online in addition to their traditional medium, and an increasing number of people's primary source of information is now online news. According to one study, 49.5% of Americans between the ages of 18 to 49 often get news from online, and this percentage has been shown to increase over the years (Pew Research Center, 2016). Online news is easier to access than traditional news mediums for people comfortable with technology in general because of low access cost, and this increases the opportunity for readers to be exposed to more diverse opinions. In such a situation, it may be beneficial to know the political biases of the particular article you are reading in order to be a well-informed reader. Also, because of the busier daily lifestyle of the average human in modern-day society and personalized news-feeds based on the preference of the reader, it is getting increasingly harder for us to read a wide range of perspectives. If readers were informed of the political biases of an article before reading it, they may be able to make balanced and efficient decisions of what to read and what not to read. Thus, there are many use cases for detecting political biases in a news article and its importance in modern-day society seems to be increasing by the minute. Yet, this is a hard problem to solve. In this paper, we examine various methods to detect political bias from text in order to solve this problem. We focus on detecting biases based on the usage of words and phrases in our dataset, which allows categorization of articles without the use of domain knowledge.

II. Related Literature

There have been many studies on political bias detection from textual data, which have helped us decide and develop our methodology.

Iyyer, Enns, Boyd-Graber, and Resnik (2014) have taken inspiration from work in sentiment analysis and applied a recursive neural network (RNN) framework to identify the political position evinced by a sentence. They used two datasets, the Convote dataset (Thomas et al., 2006) which consists of US Congressional floor debate transcripts from 2005 in which all speakers have been labeled with their political party (Democrat, Republican, or independent) and the ideological book corpus (IBC), which is the dataset we used. Their model was shown to outperform existing models such as bag of words and logistic regression on both datasets. We also decided to test the performance of the bag of words and RNN models.

Jelveh, Kognut, and Naidu (2014) implement ideology extraction on economics papers. They focus on detecting biased words from the textual dataset to improve the accuracy of the system. As the first step of the preprocessing the data, they first link each economist to their political activity, and create the dataset of liberal economics paper and conservative economics paper. Using those two datasets, they extract words and phrases which are highly used in the liberal dataset and conservative dataset. They utilize the χ^2 statistics from a Pearsons test of independence which is used by Taddy (2013) and Gentzkow and Shapiro (2010). They further improve the algorithm by detecting a topic of each article and extracting biased words for each topic. We use the χ^2 statistics they use in their paper to extract biased words in one of our models.

III. Dataset

The Ideological Books Corpus (IBC), developed by Gross et al. (2013), was used to train and test the models. This is a collection of books and magazine articles written between 2008 and 2012 by authors with well-known political leanings. Each document in the IBC has been manually labeled with political ideologies (right, left, and center) by political science experts.

IV. Predictive Models

A. Bi-gram model

Method:

The bi-gram model is created as follows:

1) Create three corpora of sentences that each have only liberal, conservative, and neutral sentences respectively.
2) Used maximum likelihood estimation (MLE) to generate a bi-gram (two word co-occurrence) probability dictionary. The dictionary has the the bi-gram as the key and the probability of that bi-gram occuring in a specific

corpus as the value. This probability was derived as follows:

$$P(wn|wn-1) = \frac{count(wn-1w)}{count(wn-1)}$$

Result: When predicting if a sentence is liberal, conservative, or neutral, the model's accuracy was 17.24%, which was far worse than random guessing. However, when letting the model predict if a sentence is either liberal or conservative, the accuracy was 65.56%.

When investigating why the results were as so, we noticed that the model was giving neutral the highest probability to most sentences in the test set. This seemed to be the limiting factor when letting the model classify the sentence into either one of the three categories. The reason why the model behaves like this has yet to be confirmed from this research.

Also, notice that because the bi-gram model returns the probability of the given input falling under a certain category, it may be more useful for analytic purposes compared to other models explored in our research, which can only tell us which category the input is most likely to be in. It is important to note that this does not get reflected in the accuracy.

The aim here was to create a baseline model, thus we did not fully optimize it. For example, the method of how we handle unknown words, i.e. words that have not appeared in the training set, was a simple unsophisticated method which simply creates a token called $< UNK >$, which all unknown words will be labeled as.

B. Bag of words + Principle components analysis + K-means

Method: We trained this model as follows:

1) Create a vocabulary dictionary by storing all unique words that appear in the training set, associating each with a unique id.
2) Vectorize each document, i.e. sentence, in the training set using the bag of words method. The column number corresponds to the word id in the dictionary and the value in the vector is the frequency of the word in the document.
3) Derive the eigenvectors of the training set matrix using singular-value decomposition.
4) Conduct principle components analysis (PCA) by projecting the training set matrix, which are created by stacking up the vectorized documents, onto the eigenvectors.
5) Take the mean vector of each liberal and conservative documents, and train the k-means model.

Result: This model ends up predicting with the highest accuracy at around 75% on average, although we initially intended it to be just a baseline model. PCA plays a big role in increasing the accuracy, as was shown to increase the accuracy by around 10% after adding it onto a bag of words only k-means model. When attempting to reduce the dimensions, i.e. columns, of the input matrix by keeping enough dimensions in eigenspace that explain $n\%$ of total variance in the data, the model was not able perform as well as when keeping all dimensions in eigenspace. Therefore, we did not use PCA to reduce the dimension of the input matrix, but only to decorrelate features from each other.

We further explored the result to see which features, i.e. words, influences the data more, by looking at which features influences the eigenvector with the highest eigenvalue.

Table 1 shows the top 10 words of the top 10 eigenvectors. The words were stemmed when processing the data, thus some words are not complete.

TABLE I: Top 10 influential words of top 10 eigen vectors

Eigenvector	Top 10 Influential Words
1	corpor, american, rate, will, benefit, incom, credit, make, tax, cut
2	public, use, economi, peopl, care, govern, creat, job, make, health
3	product, tax, new, energi, will, can, make, job, creat, govern
4	market, reform, worker, will, creat, govern, energi, insur, health, car
5	less, credit, econom, peopl, cut, govern, make, creat, will, tax
6	care, health, produc, econom, product, use, govern, make, creat, energi
7	social, econom, american, product, creat, peopl, energi, make, use, wil
8	creat, market, problem, health, peopl, use, make, energi, free, wil
9	use, benefit, market, make, free, secur, social, econom, will, can
10	energi, interest, financi, produc, will, can, peopl, use, market, produ

This alone does not lead to any conclusions on whether these words influence political ideology of a sentence.

C. Bag of words + Decision Tree

Method:

The decision tree method works the following way

1) We create a word dictionary by extracting unique words which are used in the training data set. The dictionary contains 6916 words
2) We vectorize each sentence into matrix by counting the frequency of each work in the dictionary created on the step 1.
3) We create and train a decision tree model whose node is represented by each word

Result: The model predicts whether each sentence is liberal or conservative with the accuracy of 56.35% on average. It is better that the random guess, but the accuracy difference is not as big as other models. Especially compared to the K-means and Bi-gram methods, the accuracy is not as significant.

There are several limitations in this model. First, there are several ways to improve a traditional decision tree model such as ensemble learning. It important to remove some of the nodes to construct a model which only includes informative nodes. This model utilizes the most basic decision tree model, so there are further room for optimization.

Second, each word embedding matrix has 7000 features in this mode, and it is hard for a decision tree to pick up the most informative node as I mentioned above. Instead of improving the model by changing how it trains a model, we try to optimize the input, the matrix of vectorized sentences. Therefore, we use PCA and try to reduce the noise in the features in the next model.

D. Bag of words + Principle components analysis + Decision Tree

Method:

1) We run step 1 to 4 of model B. Since reducing the dimensions of the matrix do not improve model B, we keep all of the eigenvectors in this model
2) We construct a decision tree model using the matrix which is yielded as a result of project the training dataset onto the eigen space.
3) To get predictions of sentences in the test test, we also project the test dataset onto the eigen space using the same eigen matrix same as step 2 in this model.

Result: This model yields the 51.79% accuracy, which is lower than the standard decision tree model above. Using the matrix projected onto the eigen space does not improve the accuracy in this case. In theory, PCA can improve the accuracy of decision tree model, so it is unclear why projecting to the eigen space does not improve the accuracy.

Another approach of creating an informative node is to somehow choose informative features from the dataset. In this case, it means to choose words or phrases that are used differently in liberal sentences and conservative sentences. In the next model, we utilize the method which is used by Jelveh, Kognut, and Naidu (2014), and introduced by Taddy (2013) and Gentzkow and Shapiro (2010) to choose most informative features.

E. Extracted Biased Phrases + Decision Tree

Method:

1) The χ^2 statistics from a Pearsons test of independence introduced by Taddy (2013) and Gentzkow and Shapiro (2010) calculates the χ^2 as follows:

$$\chi^2_{pl} = \frac{(f_{plr} f_{\sim pld} - f_{pld} f_{\sim plr})^2}{(f_{plr} + f_{pld})(f_{plr} + f_{\sim plr})(f_{pld} + f_{\sim pld})(f_{\sim plr} + f_{\sim pld})}$$

2) A word with a higher χ^2 value represents a word which is heavily used in liberal or conservative sentences. These words can be a good indicator of a biased word or phrase in a sentence. Therefore, we extract top 100 phrases with the highest χ^2 value in the liberal dataset and conservative dataset.
3) We vectorize each sentence using those biased words extracted in the previous step, and run a decision tree model on the vectorized sentences.
Since the χ^2 statistics can be used for any phrase length, I repeat step 1 to 3 using a single word phrases, two word phrases, and tree word phrases.

Result:

Table II represents top 10 biased single words for each dataset, and they are the same for each dataset. Even though these words do not seem to improve the amount of information each node has since every word is used heavily in both dataset, the model using these words achieves the 57.92% accuracy. One of the reasons that this model improves the standard decison tree is because a combination of those biased phrased words can differentiate between the liberal and conservative dataset very well.

TABLE II: Biased one-word phrases

Liberal	**Conservative**
Government	Government
Tax	Tax
Federal	Federal
Workers	Workers
Liberty	Liberty
Rich	Rich
Liberal	Liberal
Freedom	Freedom
Health	Health
Working	Working

TABLE III: Biased two-word phrases

Liberal	**Conservative**
the federal	the federal
the rich	the rich
federal government	federal government
child care	state to
tax cuts	tax cuts
federal reserve	federal reserve
freedom and	freedom and
the wealthy	individual liberty
tax credit	the wealthy
when the	tax credit

Table III represents top 10 biased two-word phrases for each dataset. Similar to the top 10 biased single word phrases, phrases which are heavily used in liberal sentences and conservative sentences are very similar. As you can see, there is a subtle difference in frequently phrased in the two datasets, which is different from the case of single word phrases. The model using 100 of these two words phrases achieves 56.01%.

Table IV represents top 10 biased three-word phrases for each dataset.Compared to single word phrases and two-word phrases, three-word phrases often used in liberal and conservative dataset differ a lot. This makes sense since as the word length increases a phrase contains more meaning. Even though there is more distinct difference between the tow datasets, this model yields 54.3 % accuracy.

In the extracted biased phrases + decision tree model, using single word phrases gives us the highest accuracy. This seems contradictive sine many of the single word biased phrases are used in both of the datasets. One of the limitations here is also the limited number of sentences in each dataset, and the validity of this χ^2 statistics should be examined more.

TABLE IV: Biased three-word phrases

Liberal	**Conservative**
the federal government	the federal government
the federal reserve	the federal reserve
earned income tax	earned income tax
the private sector	the state to
for the rich	the first place
a way that	way of life
income tax credit	would have been
the free market	federal government to
tax cuts for	the private sector
in a way	income tax credit

F. Convolutional neural network

Method: Here we used tensorflow to create a standard convolutional neural network.

1) Load a list of words and its corresponding word vectors. This vector was derived from looking at word embedding and co-occurrence in the English Wikipedia Corpus.
2) Create an ids matrix, which is a matrix where each row represents a sentence and the n-th column is the n-th word in the sentence. The value of each element in the matrix is the index number of the word in the word list. This information is later used to retrieve the corresponding word vector to create a three-dimensional matrix.
3) Create a histogram showing the distribution of the length of each sentence in order to decide the number of columns the input matrix should have. From examining Figure 1, we arbitrarily decided 61 to be a good number that covers most sentence lengths in our dataset.

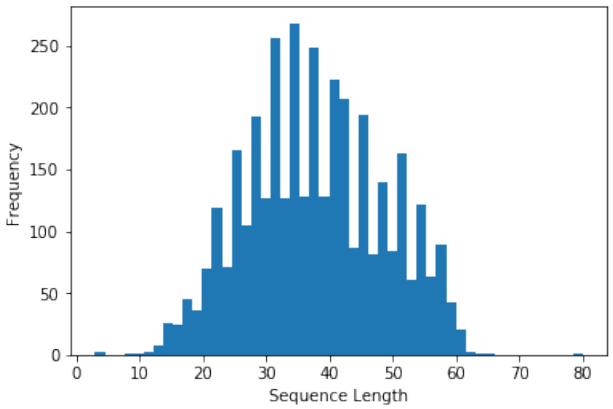

Fig. 1: Distribution of the length of each sequence

4) Set the cost function to minimize cross entropy and used the Adam optimizer with a learning rate of 0.1 to do so. The Adam optimizer is a popular optimization algorithm used in place of the standard stochastic gradient descent algorithm, which was available by default in tensorflow.
5) Set the number of hidden layers to 2 and the number of neurons in each layer to 256.
6) Trained the model with batches of 128 for 10000 iterations, optimizing weight values on each iteration.

Result: The result is too good to be true, with an average accuracy of 99.76%. One thing that we learn from this is that debugging neural networks is a completely different beast compared to the usual debugging process. It is difficult to tell if a simple mistake is made in the code or if the model happens to train in a way such that it could generalize really well to the test set. We make sure that our dataset labels are not included in the training set, a typical mistake that can lead to abnormally high accuracy, and it might be the case that the model is actually performing this well, but we are still skeptical.

G. Recurrent neural network

Methods:

1) - 3) Same step as convolutional neural network.
4) Construct a recurrent neural network, which also uses long short-term memory (LSTM). LSTM is used to take a relationship of words which are apart from each other into account. Even though there are various architectures of LSTM system, we used the standard LSTM structure. It is shown that recurrent neural network which has LSTM inside tends to perform better.
5) We also randomly dropped a node to avoid overtraining, and we used 0.5 for dropout ratio parameter.
6) In addition to the drop out parameter, we used iteration number, the number of LSTM in a neural network, and batch size as parameters. We constructed the neural network using Tensorflow.

Result: While we trained the model, we monitored the performance of each model by plotting the accuracy and loss function of each model shown in Figures 2 and 3 in the Appendix.

One of the limitations of this model is that the dataset only has 3400 sentences in total, and the recurrent neural network tends to end up being overfitting if we use the larger parameter for iteration number. Therefore, we also added the congressional debates with annotations of each authors partisanship, which gave us more than 80000 sentences in total. Since the neural network with 96 LSTM units performed the best, we trained the model with same number of LSTM units and drop out rate. We also monitor how the network is trained on training data set, and here is the result.

As you can see in Figures 4 and 5 in the Appendix, this model was not able to construct a proper network that works on the training set. There are two reasons. First, it requires more parameter adjustment, and complicated network. Second, since congressional dataset includes each sentence with the author level partisanship, the data includes more noise than the IBC data set. Obviously, it requires further improvements to asses the model, but we were not able to work on it because of the limited time and computation power.

V. Conclusion

In this paper, we explored various methods to detect political biases in textual data. To our surprise, the optimized k-means method using bag of words with PCA ended up performing the best out of all models including the RNN. Furthermore, this model surpassed the best RNN model result from Iyyer, Enns, Boyd-Graber, and Resnik (2014) on the IBC dataset, in which their accuracy was 69.3% and ours was an average of 75%. This highlights two important points. One is the potential of more traditional methods to surpass flashy newer methods if done in the right way. Even a simple method such as k-means can surpass the now-popular RNNs. This leads to our next

point of how it is important not to be discouraged to optimize a baseline model of a paper. It very well be the case that authors of papers of new models have not optimized baselines not because they believe it will perform worse than their new model, but to emphasize that their new model performs better than traditional baseline models.

ACKNOWLEDGMENT

The authors would like to thank Professor Bruce Maxwell for his encouragement and guidance through this project.

REFERENCES

[1] A. Mitchell, J. Gottfried, M. Barthel, and E. Shearer, 1. Pathways to news, *Pew Research Center's Journalism Project*, 07-Jul-2016. [Online]. Available: http://www.journalism.org/2016/07/07/pathways-to-news/. [Accessed: 20-Dec-2017].

[2] Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik, "Political Ideology Detection Using Recursive Neural Networks" in *Proceedings of Journal of the Association for Computational Linguistics*, 2014.

[3] Matt Thomas, Bo Pang, and Lillian Lee. "Get out the vote: Determining support or opposition from Congressional floor-debate transcripts", in *EMNLP*, 2006.

[4] Z. Jelveh, B. Kogut, and S. Naidu, "Detecting latent ideology in expert text: Evidence from academic papers in economics", in *Proceedings of EMNLP*, 2014.

[5] Matt Taddy, "Multinomial inverse regression for text analysis", in *Proceedings of Journal of the American Statistical As- sociation*, 2013.

[6] Matthew Gentzkow and Jesse M. Shapiro, "What drives media slant? evidence from U.S. daily news- papers", in *Proceedings of Econometrica*, 2010.

[7] Justin Gross, Brice Acree, Yanchuan Sim, and Noah A Smith. "Testing the etch-a-sketch hypothesis: A computational analysis of mitt romneys ideological makeover during the 2012 primary vs. general elections", in *APSA 2013 Annual Meeting Paper*, 2013.

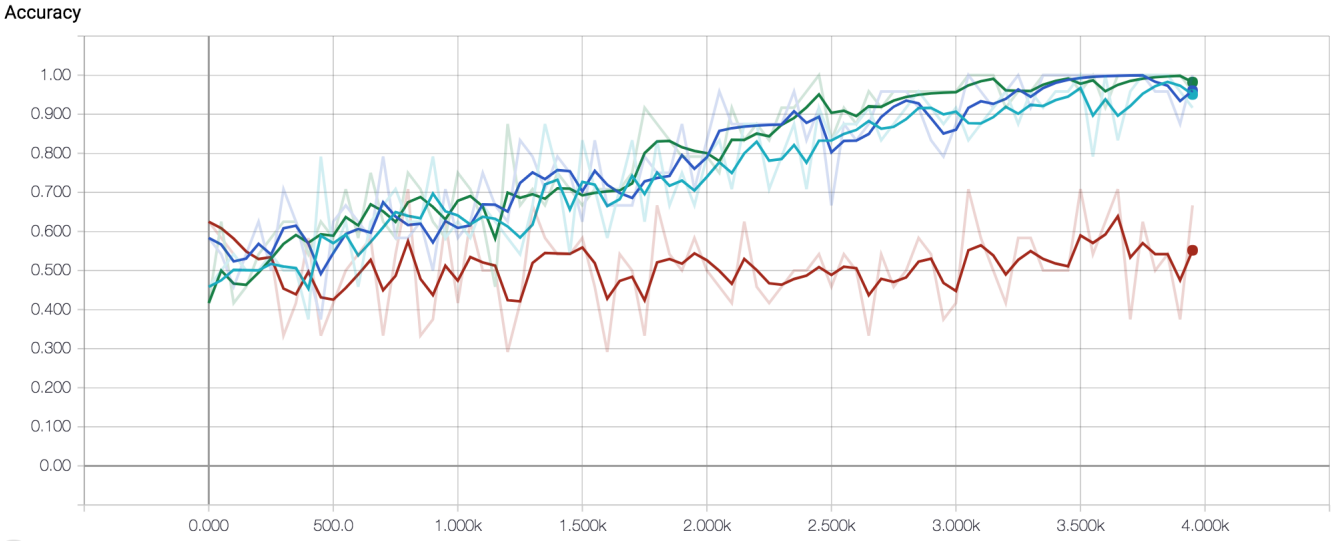

Fig. 2: Trend of the accuracy of each model.

Brown = 108 LSTM units, Green = 96 LSTM units, Blue = 64 LSTM units, Light Blue = 50 LSTM units

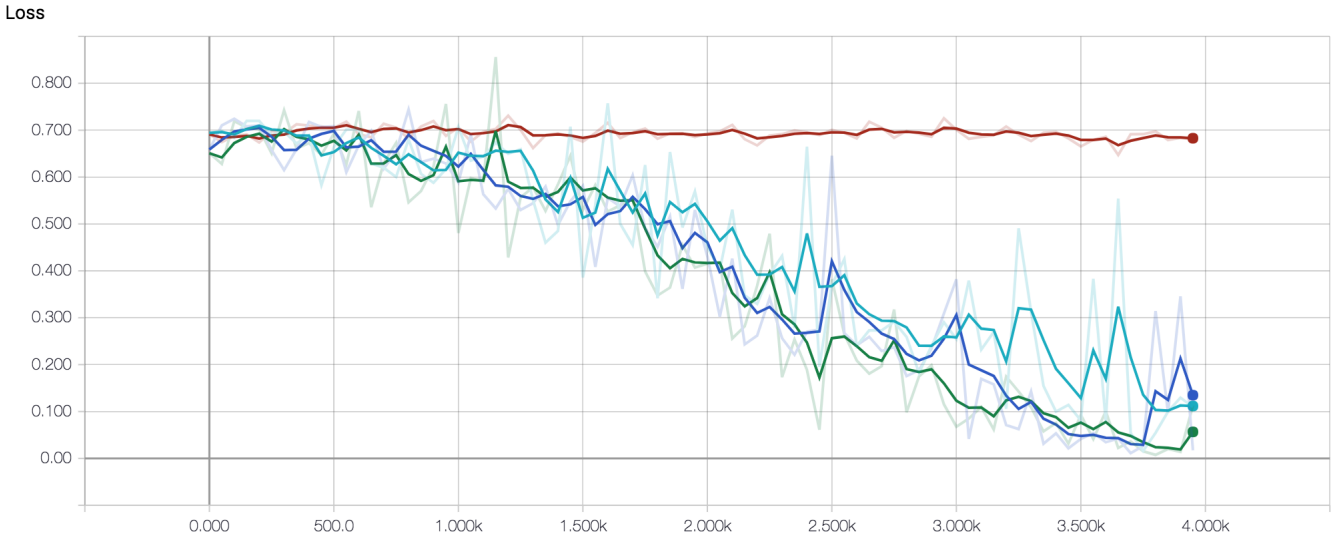

Fig. 3: Trend of the cost function value of each model.

Brown = 108 LSTM units, Green = 96 LSTM units, Blue = 64 LSTM units, Light Blue = 50 LSTM units

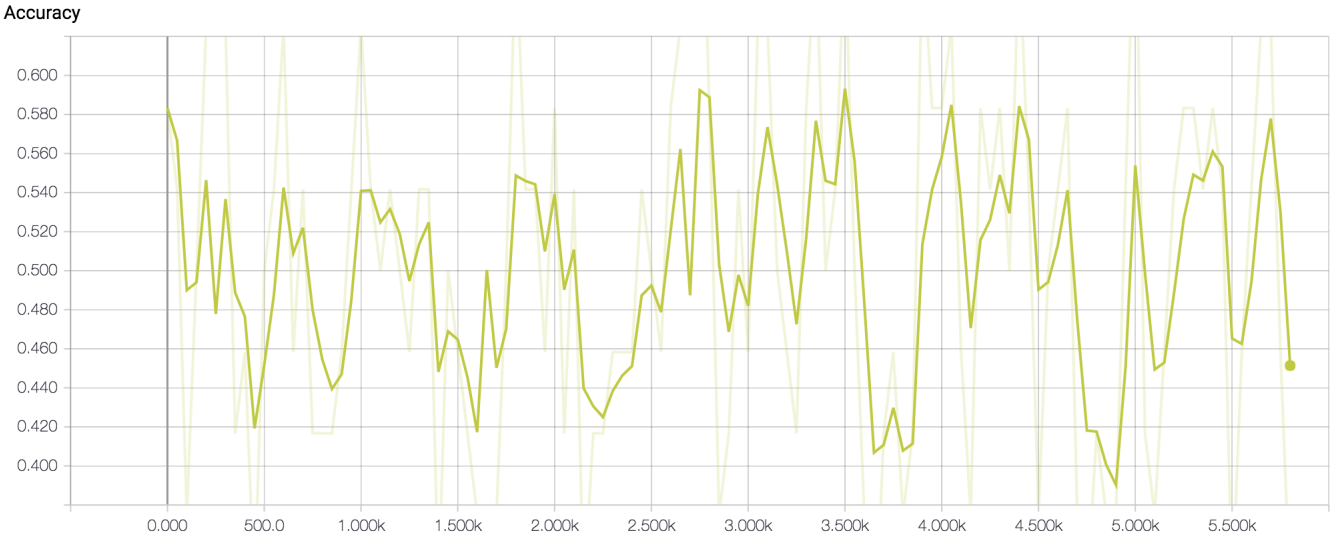

Fig. 4: Trend of the accuracy

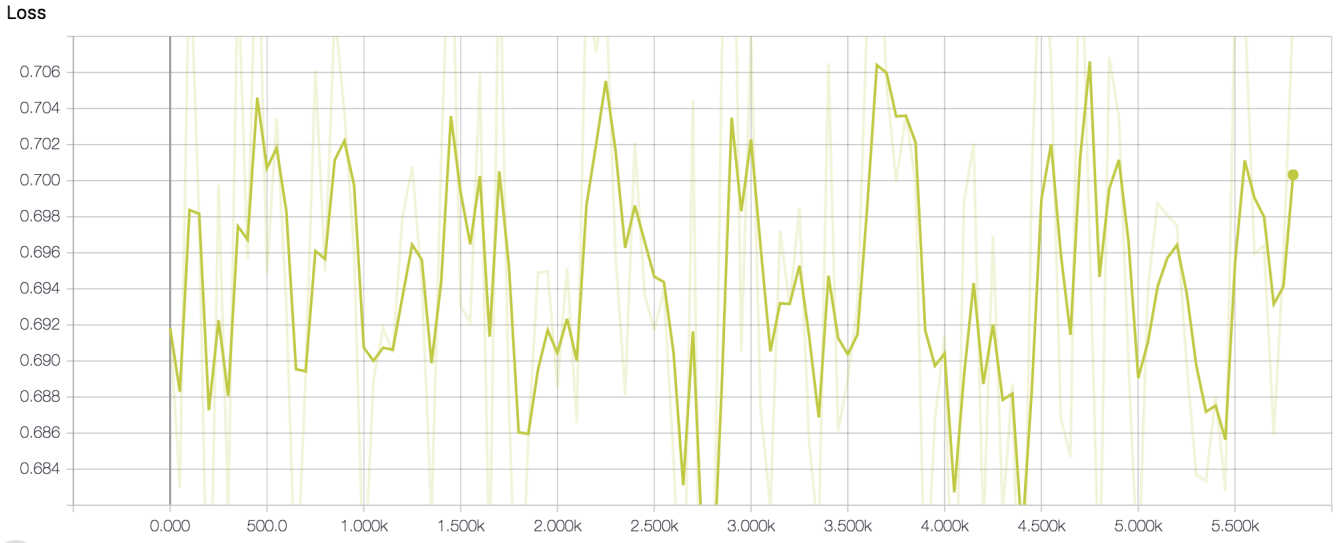

Fig. 5: Trend of the cost function value of each model