Text Mining

Final assignment: Sentiment Analysis

Group 33: Wei Chen and Jialiang Wang

1 Introduction

Sentiment Analysis is the process of analyzing, processing and extracting subjective text with emotion by using natural language processing and text mining techniques. In this assignment, the task is using sentiment analysis technique to judge a tweet whether is positive, negative or neutral. In this assignment, we use three methods, Word2vec[3], Bi-LSTM and BERT[1]. By classifying the word embeddings generated by word2vec, to achieve the sentiment analysis task in tweets.

2 Data

2.1 Data description

Each data set has the same structures, with a string of numeric labels, the sentiment which consists of three criteria: neutral, positive, and negative (three-point scale). Also, the final columns is the tweets. For instance, the data set *Spending my Saturday getting my car serviced is definitely the most enjoyable thing I could do with my time.* is negative in sentiment analysis. *It's Super Bowl Sunday, pastors. Get your Jesus Jukes ready! Guilt is an awesome motivator!* is positive, and *#day #God #known #Sunday #week Question and Answer http://t.cotz18q9Brvy #prayer #church* is neutral.
But there are some duplicated items during our data pre-processing. In the datasets 2016, the 2016-testA has 51 duplicated items and 2016-trainA has 19 duplicated items. In the datasets 2015, the 2015-testA has 7 duplicated items. In the datasets 2013, the 2013-devA has 4 duplicated items, the 2013-testA has 2 dupicated items and the 2013-trainA has 29 duplicated items.

Table 1: Twitter2016 sentiment data description

	testA	trainA	devtestA	devA
Neutral	10314	2036	681	765
Positive	7046	3082	994	843
Negative	3221	863	325	391
Sum	20581	5981	2000	1999

Table 2: Twitter2015 sentiment data description

	2015-trainA	2015-testA
Neutral	253	985
Positive	170	1034
Negative	66	364
Sum	489	2383

Table 3: Twitter2014 sentiment data description

	2014-sarcasmA	2014-testA
Neutral	13	669
Positive	33	982
Negative	40	202
Sum	86	1853

Table 4: Twitter2013 sentiment data description

	2013-devA	2013-testA	2013-trainA
Neutral	737	1513	4568
Positive	573	1473	3633
Negative	340	559	1454
Sum	1650	3545	9655

2.2 Exploratory data analysis

Initially, only data from 2016 is used. Then, we found that classifiers we applied to larger data led to better results. For this reason, we stitched together the data from 2013 to 2016 manually for simplicity, which means that for our sentiment analysis task, all tweet data given is considered.

This dataset for sentiment analysis has three categories: (1)positive, (2)neutral, (3)negative. The distribution of the labels is shown as figure 1.

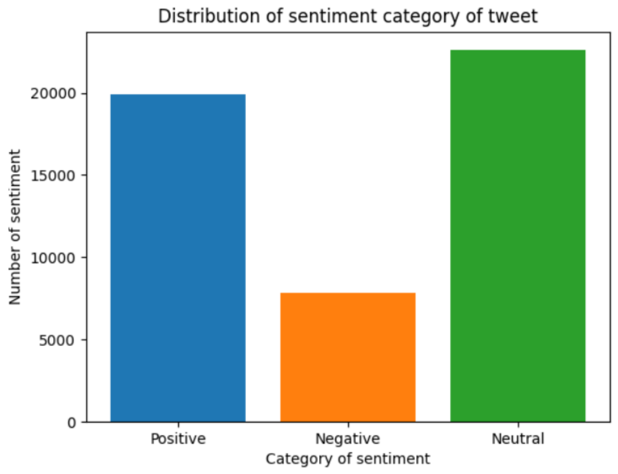

Figure 1: Distribution of the sentiment category of tweet

There are **19903** positive tweets, **7840** negative tweets, and **22591** neural tweets. In the pre-processing step, we apply several regular expressions to make the messy tweet data clean. Basically, they are : (1)remove hyperlinks (e.g., http/https) (2)remove mentions (e.g., @Trump) (3)remove special characters (e.g., punctuation, backlash, and Ampersand) (4)remove numbers (e.g., 123, 0001 and telephone) (5)remove words with number (e.g., 45tem and ok111) (6)lowercase (7)remove words whose length is less than three. (8) keep hashtag (e.g., #satire) Additionally, stemming, lemmatization, and stopwords technologies are introduced initially, but there is only a slight improvement and even worse sometimes. Therefore, we do not use them.

After the process of pre-processing, the size of the vocabulary for the tweet data drops greatly than originally. The distribution of the length of tweet in this dataset is shown as figure 2.

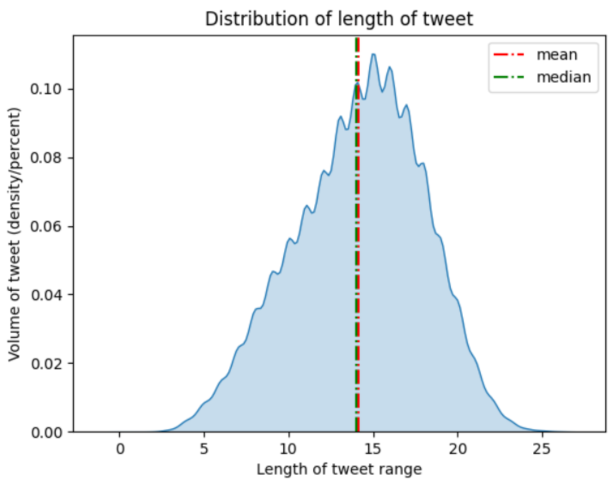

Figure 2: Distribution of the length of tweet

Additionally, the statistics of tweets is as table 5 shown, where the count of words (not unique) in all tweets is **713295**.

Length of tweet				
Max	Min	Mean	Median	Standard deviation
26	0	14.17	14	3.79

Table 5: Statistics of tweets where the count of all words (not unique) is 713295

Finally, WordCloud is used to visualize the frequent word (top 50) for all three categories. See figure 3, 4, and 5 for positive, negative, and neutral tweets.

As expected, positive tweets tend to have positive words, e.g., good, hope and excited, while negative tweets generally have negative words, e.g., hate, shit and bad.

3 Related work

In this section, we will introduce the background knowledge of sentiment analysis and the related work in sentiment analysis. We will also give a brief introduction about the methods we use in the final assignment.

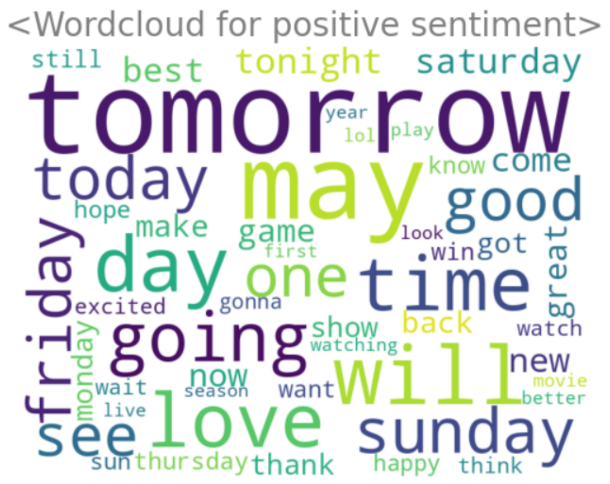
<Wordcloud for positive sentiment>
still best tonight saturday
tomorrow
year lol play
today may know come good
hope make game look win got great
friday day one time
excited gonna
monday going show watching back watch
wait now season want will new movie better
live see love sunday
sun thursday thank happy think

Figure 3: Wordcloud for positive sentiment

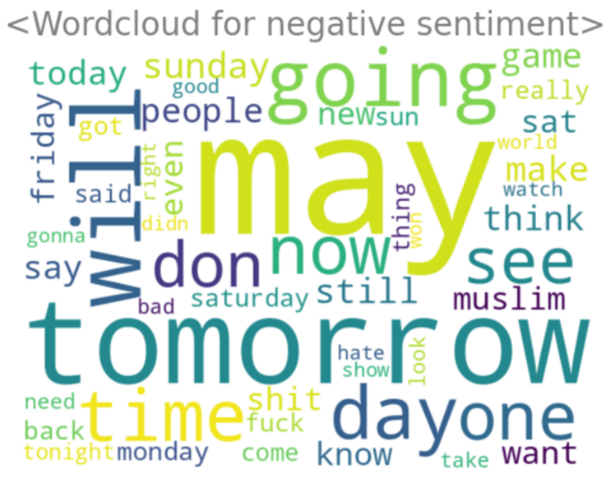
<Wordcloud for negative sentiment>
today sunday going game
good really
friday got people newsun sat
world
will may make
said right even watch
gonna didn thing won think
say don now see
bad saturday still muslim
tomorrow
hate show look
need shit
back time fuck dayone
tonight monday come know take want

Figure 4: Wordcloud for negative sentiment

Sentiment analysis which studies the sentiment of text and sentiment analysis is an important branch of natural language processing. But in this task, we deal with the Twitter data. There are several challenges, like the informal language in tweets, or the sarcasm expressed in tweets. In sentiment analysis, we have two major tasks, one is text representation, in which word embedding[4] is representation of words which is relatively low-dimension and dense vector. If we use word embedding technique, two vectors close to each other means the original word share the similar meaning. Word2vec[3] is a better method to train vector space model, in this assignment, we use word2vec to generate the representation of tweets. The other task is classification, which consists of two

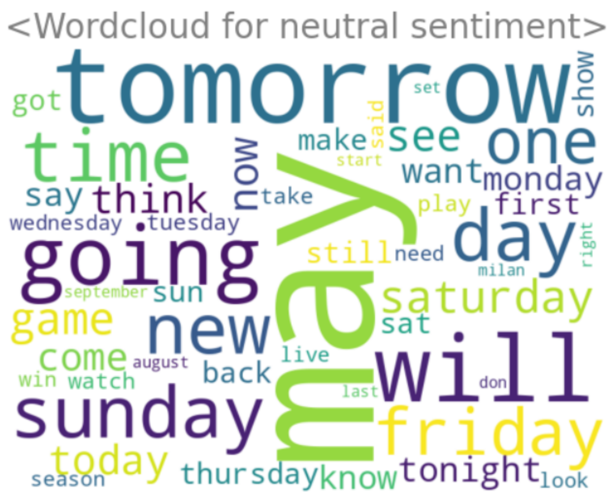

Figure 5: Wordcloud for neutral sentiment

methods, one is knowledge-based. In knowledge-based method, we can choose words or phrased in the text which could express sentiment. For example, *great, excellent or awful*, *great and excellent* could express positive feeling and *awful* could express negative feeling. But in sentiment analysis, the words might express the opposite meaning in a sarcasm way. The other type of classification method is machine learning. During the course, we learned Naive Bayes, SVM, RNN[5] and LSTM[2]. By training a classifier, we could achieve a classification task. In this assignment, we used Bi-LSTM.

LSTM is an abbreviation for Long Short-Term Memory[2] which can not encode the back-to-front information. Under stricter classification condition, for instance, strong degree of positive or weak degree of positive, strong degree negative or weak degree of negative and neutral. LSTM does not perform well. Bi-LSTM is an abbreviation for Bi-directional Long Short-Term Memory which is used to model contextual information, in which forward LSTM and backward LSTM are combined together. The forward LSTM will generate corresponding vector according to the sentence, the backward will also generate the vector. After we combine these two vectors together, we can get all information in two directions.

However, in reality, the amount of task-specific labeled data is much less than the amount of unlabeled data. We need pre-train, in which model parameters are no longer randomly initialized, but pre-trained by some tasks. Generally, a large amount of training data collected at low cost is put together, pre-trained in some way to learn the commonalities, and then the commonalities are transferred into a task-specific model, and then fine-tuned using a small amount of labeled data from the specific domain in question. In this way, the model only needs to "learn" the "special" parts of the particular task from the "commonalities". We use BERT[1] in this assignment. The Encoder structure of Transformer[6] is used, but the model structure is deeper than Transformer. The traditional CNN and RNN are discarded in Transformer, and the whole network structure is composed entirely of Attention mechanism. More precisely, the Transformer consists of and only consists of Self-Attention and feed forward neural network. A trainable neural network based on the Transformer can be built by stacking the Transformer. The authors' experiments were performed by building Encoder-Decoder with 6 layers each of encoder and decoder, for a total of 12 layers.

4 Methods

For our tweets sentiment analysis task, three classifiers (Naive Bayes, Bi-LSTM, and BERT) and three word representations (Bag of words, TF-IDF and Word2vec) are used, where we regard Naive Bayes as a baseline for evaluation.

4.1 Naive Bayes

First of all, to learn word features and represent words as vectors, Bag of words (i.e., CountVectorizer) and TF-IDF (i.e., TfidfVectorizer) are imported. Next, once word features are learned from the texts of the pre-processed tweets, Naive Bayes (MultinomialNB) is applied.

4.2 Bi-LSTM

For this classifier (Bidirectional-LSTM), instead of representing words as high-dimensional and sparse vectors using BOW or TF-ITF, we use Word2Vec (Gensim) to generate dense vectors (or embeddings) since using dense vectors on neural networks tends to have better performance.

Firstly, we use Word2Vec to extract word features and generate our own word embedding model from the corpus rather than using pre-trained models (e.g., GloVe). Then, once each word in the vocabulary has its corresponding dense vectors (or embeddings), we create an embeddings matrix to include a feature vector of words for later bi-LSTM training. Finally, bi-LSTM combined with CNN is built on tensorflow and is trained.

The whole training process for predicting sentiment is shown in figure 6, where the embedding layer is used for reloading the word embeddings trained by Word2vec beforehand. The embedding layer is able to reload pre-trained embeddings (that is exactly what we do here without fine-tune) or retrain words represented by the index initially from scratch to generate dense embeddings (no need to use Word2Vec). For further details of our architecture of bi-LSTM, see table 6.

Embedding layer
CNN 1D (filter=32, kernal=3)
Activation (relu)
MaxPooling 1D (size=2)
Bidirectional-LSTM (32)
Dropout layer (0.5)
Dense (3)
Activation (softmax)

Table 6: Architecture of bi-LSTM

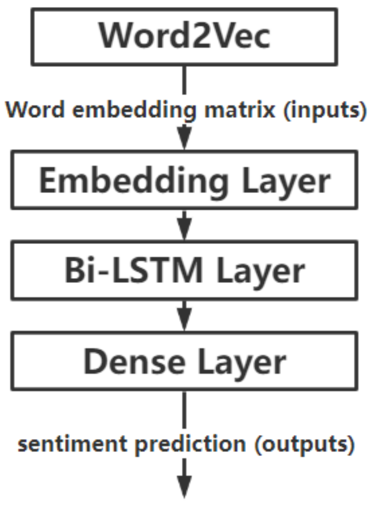
Word2Vec
Word embedding matrix (inputs)
Embedding Layer
Bi-LSTM Layer
Dense Layer
sentiment prediction (outputs)

Figure 6: Process of implementing sentiment analysis based on bi-LSTM

4.3 Bert

In recent years, BERT is a state-of-art technology for NLP tasks. For the sentiment analysis task, BERT with a self-attention mechanism is introduced to our task. In our task, we import BertTokenizer (from 'bert-base-uncased') and TFBertModel to fit the tweet data, where we use BertTokenizer to generate ids and attention mask for each tweet (or text) from the train set and test set, respectively. Then, once features are automatically built through tokenizer, we pass the id feature vectors and attention masks as inputs to the fine-tune model.

5 Results

The Twitter dataset contains about 50000 tweets and we split them into two sets, 75% tweets for training and 25% for testing.

Furthermore, the random seed is set to 123 for all situations (functions).

5.1 Baseline: Naive Bayes

As stated above, Naive Bayes classifier is seen as the baseline for evaluation. The performance of Naive Bayes with bag of words feature representation and with TF-IDF are as tables 7 and 8 shown.

Classification report for Naive Bayes (BoW)			
	Precision	Recall	F1-score
Positive	0.65	0.64	0.64
Negative	0.42	0.41	0.41
Neural	0.61	0.62	0.62
Accuracy			0.5977
Macro avg	0.56	0.56	0.56

Table 7: Classification report for Naive Bayes (BoW)

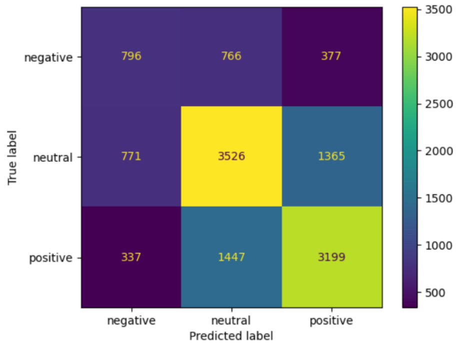

Figure 7: Confusion matrix for Naive Bayes (BoW)

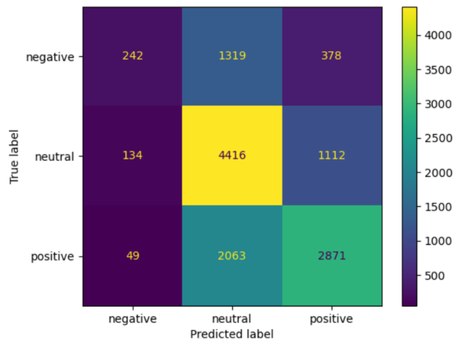

Figure 8: Confusion matrix for Naive Bayes (TF-IDF)

Classification report for Naive Bayes (TF-IDF)			
	Precision	Recall	F1-score
Positive	0.66	0.61	0.64
Negative	0.60	0.11	0.19
Neural	0.58	0.78	0.66
Accuracy			0.6085
Macro avg	0.61	0.50	0.50

Table 8: Classification report for Naive Bayes (TF-IDF) , where the hyperparameter n-gram is set to (1,1)

Additionally, we plot the confusion matrix to show the prediction performance of Naive Bayes visually (see figures 7 and 8).

It can be found that using TF-IDF to represent word features results in better precision and accuracy than using bag of words. However, it is hard for the classifier to correctly recognize negative tweets for both two word representations (BoW or TF-IDF) and we considered that this results from the distribution of the category of tweets is imbalanced, where the number of negative tweets is much smaller than the number of tweets in the other two categories (see figure 2), thereby making it difficult to say which word representation is better.

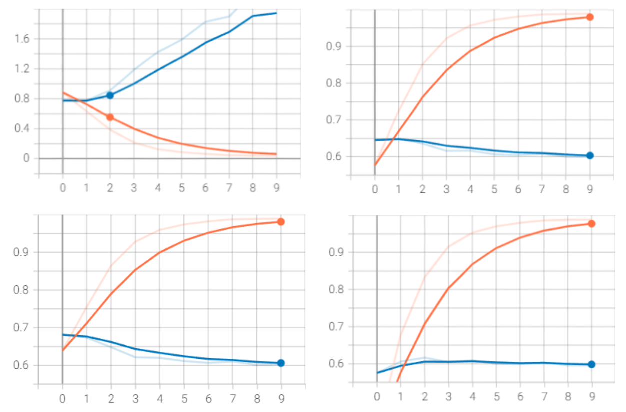

Figure 9: Classification report for Bi-LSTM on train set (orange) and test set (blue) where the top left, top right, bottom left and bottom right sub-figures are loss, accuracy, precision and recall, respectively (based on tensorboard)

5.2 Bi-LSTM

We pass the word embeddings pre-trained by our own to the Bi-LSTM layers and freeze the parameters for word embeddings, which means we take as input word embeddings without fine-tuning. The results is shown in figure 9, while the selection of (hyper)parameters for both Word2Vec and bi-LSTM is shown in table 9.

Selection of (hyper)parameters for Word2Vec				
Word dimension	Window size	Min count	Epoch	SG
300	5	1	20	1
Selection of (hyper)parameters for Bi-LSTM				
Text length	Batch size	Epoch	Optimizer	Learning rate
20	64	10	Adam	1e-3

Table 9: Selection of (hyper) parameters for both Word2Vec and bi-LSTM

It can be seen that as the epoch increases three metrics accuracy, precision, and recall for the test set gradually drop and finally reach a plateau. In addition, the final results for these three metrics are similar to that of Naive Bayes, which means it appears that using the Bi-LSTM model we defined is only a slight improvement on Naive Bayes with BoW and TF-IDF. We considered that the imbalanced distribution of our data and relatively small dataset lead to overfitting existing in our model.

5.3 BERT

Due to the performance of the bi-LSTM, we then decided to add one more experiment using BERT model based on transformers architecture. The choice of the (hyper) parameters is as table 10 shown, whereas the result of evaluation for each epoch on test (validation) set is as table 11 shown.

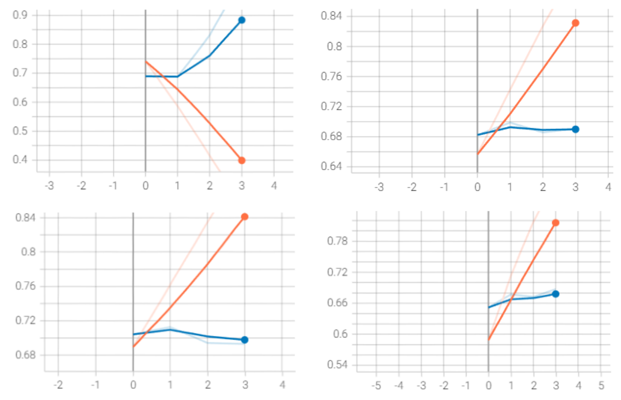

Figure 10: Classification report for BERT on train set (orange) and test set (blue), where the top left, top right, bottom left and bottom right sub-figures denote loss, accuracy, precision and recall, respectively (based on tensorboard)

Selection of (hyper)parameters for BERT			
Max length	Learning rate	Batch size	Epoch
20	2e-5	1	4

Table 10: Selection of (hyper) parameters for BERT

Classification report for BERT				
	Loss	Accuracy	Precision	Recall
Epoch 1	0.6893	0.6824	0.7041	0.6515
Epoch 2	0.6873	0.6987	0.7126	0.6774
Epoch 3	0.8307	0.6857	0.6944	0.6725
Epoch 4	1.0283	0.6908	0.6933	0.6875

Table 11: Classification report for BERT

It can be found from the curve graph of training and evaluation (see figure 10) that BERT has a great advantage over Naive Bayes and Bi-LSTM. The final recall (i.e., avg-recall) reaches 0.6875 which is higher than the bi-LSTM with around 0.6 recall and the Naive Bayes with about 0.56 recall (BoW) and about 0.5 recall (TF-IDF). In addition, using BERT yield an about 10% accuracy rate greater than the baseline on average.

6 Discussion

6.1 Limitations and implications

We find that there exist a few limitations and implications when implementing this sentiment analysis task:

- Imbalanced distribution of sentiment category of the tweet data
  For example, the number of tweets for each sentiment category is 19903 (positive), 22591 (neutral), and 7840 (negative). Obviously, the number of negative tweets is not even half

of the other categories. Therefore, in our baseline model (i.e., Naive Bayes), we believe that it is more likely the main reason resulting in bad performance of predicting and recognizing negative tweets. In addition, it equally affects other models (bi-LSTM and BERT).

- Relatively small size of dataset
  Even though all the years of Twitter data are stitched together, it still looks small intuitively, especially after pre-processing dirty data. Hence, it is considered as a factor that makes the bi-LSTM model over-fitting.
- The limited topics from the dataset
  The main content of data might be not diverse, which means the topics from the tweet dataset given are limited. For instance, the tweets might only concentrate on limited fields such as hardware and sports, to the exclusion of everything else (less diverse) and this probably impacts the generalization and robustness of the models when training.

6.2 Future work

- Use more data (need to be diverse)
- Carry out some improvements to the data
  pre-process step For our data pre-process step, we feel that some operations (we defined) of the regular expression are not appropriate for cleaning tweets. Also, it needs more effort to explore the tweet data and determine whether apply stopword, lemmatization, and stemming technology.
- Apply more methods
  For example, SVM, random forest, 'normal' RNN with dense and sparse features. Moreover, it is said that there are more advanced word embeddings methods.

7 Conclusion

To complete this sentiment analysis task, three classification methods (Naive Bayes, Bidirection-LSTM, and BERT) and three word feature learning technologies (Bag of words, TF-IDF, and word2vec) are used. Also, we take Naive Bayes as the baseline model.

The dataset we used consists of all years of tweet data given but the size of the dataset is relatively small and the distribution of the category of tweets is imbalanced, where the number of negative tweets is too few. Hence, we concluded that these factors lead to over-fitting for the bi-LSTM model and a large number of incorrect predictions of negative tweets for the baseline model.

For the results of evaluation on bi-LSTM, what we expected is that intuitively it has obviously better performance than the baseline model but the results of the two are similar. However, we achieved significant results when using BERT. Therefore, we concluded that the BERT model is best suited for sentiment analysis tasks.

8 Contributions of the team members

For this task, the contributions of each team member are rough as follow:

- Wei Chen
  - Group discussion
  - Writing code (apply methods and evaluation)
  - Writing report sections relating to code (methods, results, etc.)
- Jialiang Wang
  - Group discussion
  - Give advice on how to implement the task (e.g., which models? which evaluation methods?)
  - Writing report sections (Introduction, Related work, Data, etc.)
  - Testing the quality of the code and give feedback

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[2] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[4] Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *International workshop on artificial intelligence and statistics*. PMLR, 246–252.

[5] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature* 323, 6088 (1986), 533–536.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762