Predicting Similar Quora Question Pairs Using Machine Learning Model and Further Comparison with Deep Learning Method

Final Project Report for
LING 131A Introduction to Natural Language Processing

By
Guirong Liu
(guirongliu@brandeis.edu)
Linqi Wang
(linqiwang@brandeis.edu)
Linxuan Yang
(linxuany@brandeis.edu)
Yu Huai
(yhuai@brandeis.edu)

Under the guidance of
Professor Marcus Verhagen

Department of Computer Science
BRANDEIS UNIVERSITY, WALTHAM
December 2018

1. Introduction

1.1 Background and Project Description

Quora is a platform to ask questions and connect with people who contribute unique insights and quality answers. Just like other platform, there are many people asking comparable questions with similarly words. Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question and make writers feel they need to answer multiple versions of the same questions. We are going to tackle this NLP problem by applying advanced techniques to classify whether question pairs are duplicates or not. By finding duplicate questions, the algorithm will cut down on possible replicates, making it easier to find related answers for the seekers.

1.2 Dataset Description

The data comes from Quora, which includes a training data: 255027 non-duplicates and 149263 duplicates pairs.

	id	qid1	qid2	Question 1	Question 2	is_duplicate
0	0	1	2	What is the step by step guide to invest in share market in india?	What is the step guide to invest in share market?	0
5	5	11	12	Astrology: I am a Capricorn Sun Cap moon and cap rising… what does that say about me?	I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What does this say about me?	1

Also includes a testing data: 2345796 question pairs.

	test_id	Question 1	Question 2
0	0	How does the Surface Pro himself 4 compare with iPad Pro?	Why did Microsoft choose core m3 and not core i3 home Surface Pro4?
1	1	Should I have a hair transplant at age 24? How much would it cost?	How much cost does hair transplant require?
2	2	Which food not emulsifiers?	What foods fibre?

1.3 Design Process

Our team solved this problem with the following steps:

1. Data Cleaning
2. Feature Engineering
3. Additional Features

4. Classification Using SVM model
5. Classification Using Neural Network Method

2. Data Cleaning

2.1 A Whole View of the Data

First, we need to have a whole view of data.

```
df = pd.read_csv('train.csv', encoding = "ISO-8859-1")
print(df.head())
print(df.tail())
```

```
   id  qid1  qid2                                                question1  \
0   0     1     2  What is the step by step guide to invest in sh...
1   1     3     4  What is the story of Kohinoor (Koh-i-Noor) Dia...
2   2     5     6  How can I increase the speed of my internet co...
3   3     7     8  Why am I mentally very lonely? How can I solve...
4   4     9    10  Which one dissolve in water quikly sugar, salt...

                                           question2  is_duplicate
0  What is the step by step guide to invest in sh...             0
1  What would happen if the Indian government sto...             0
2  How can Internet speed be increased by hacking...             0
3  Find the remainder when [math]23^{24}[/math] i...             0
4              Which fish would survive in salt water?             0
            id    qid1    qid2  \
404285  404285  433578  379845
404286  404286   18840  155606
404287  404287  537928  537929
404288  404288  537930  537931
404289  404289  537932  537933

                                                question1  \
404285  How many keywords are there in the Racket prog...
404286           Do you believe there is life after death?
404287                                  What is one coin?
404288  What is the approx annual cost of living while...
404289               What is like to have sex with cousin?

                                                question2  is_duplicate
404285  How many keywords are there in PERL Programmin...             0
404286          Is it true that there is life after death?             1
404287                                  What's this coin?              0
404288  I am having little hairfall problem but I want...             0
404289       What is it like to have sex with your cousin?             0
```

2.2 Choose a Meaningful and Unique Index

According to the view above, we guessed the 'id' column may be unique, and it can be used as index:

```
In [3]:
df['id'].is_unique
```

```
Out[3]:
True
```

2.3 Deal with Missing Data

First, we determined which columns possesses missing data. Second, we decided how to fill the the missing data accordingly:

In [5]:

```
print(df.isnull().any())
df[df.isnull().values==True]
```

```
qid1            False
qid2            False
question1        True
question2        True
is_duplicate    False
dtype: bool
```

Out[5]:

id	qid1	qid2	question1	question2	is_duplicate
105780	174363	174364	How can I develop android app?	NaN	0
201841	303951	174364	How can I create an Android app?	NaN	0
363362	493340	493341	NaN	My Chinese name is Haichao Yu. What English na...	0

Luckily, there are only three rows with missing data, so we can just drop them by:

In [6]:

```
df = df.dropna()
```

2.4 Deal with Wrong Values

In this case, ‘is_duplicate’ column can only take the value ‘0’ or ‘1’. We can count the occurrences of each value first.

In [7]:

```
df['is_duplicate'].value_counts()
```

Out[7]:

```
0    255024
1    149263
Name: is_duplicate, dtype: int64
```

As the data had already been pre-processed, there is no wrong value for ‘is_duplicate’ column.

2.5 Remove White Spaces and ‘\n’ of Strings

In [8]:

```
df['question1'].str.strip()
df['question2'].str.strip()
```

2.6 More Specific Cleaning

Observing that ‘qid1’ column and ‘qid2’ column are comparable, and they are related to each other closely. We can sort the data such that qid1 is always smaller than qid2 for each row.

```
print(len(df[df['qid1'] > df['qid2']]))
print(len(df[df['qid1'] < df['qid2']]))
# help(df.iterrows)
df1 = df[df['qid1'] < df['qid2']].copy()
df2 = df.loc[df['qid1'] > df['qid2']].copy()
c1,c2,c3,c4= df2['qid1'].copy(),df2['qid2'].copy(),df2['question1'].copy(),df2['question2'].copy()
c2 = df2['qid2'].copy()
df2['qid1'],df2['qid2'],df2['question1'],df2['question2'] = c2, c1,c4, c3
df3 = pd.concat([df1,df2])
print(len(df3[df3['qid1'] > df3['qid2']]))
print(len(df3[df3['qid1'] < df3['qid2']]))
print(df3.set_index(['qid1','qid2']).index.is_unique)
df = df3
```

First, we count the number of qid1 that are bigger than qid2, and switch the position of qid1 and qid2 to make all qid1 are smaller than qid2. Second, we test whether all pairs of qid1 and qid2 are unique:

Number of qid1 > qid2 = 88201

Number of qid1 < qid2 = 316086

After Switching:

Number of qid1 > qid2 = 0

Number of qid1 < qid2 = 404287

If all pairs are unique = True.

3. Feature Engineering

Feature engineering is the process of using data to extract features that can be used in machine learning to specific label assigning. Good features can increase the percentage of accurate labeling. In this project, according to the data and project goal, the features we create cover the following areas: questions without stop words, character length ratio, number of same words or synonyms, SOW model, BOW model, and syntactic features.

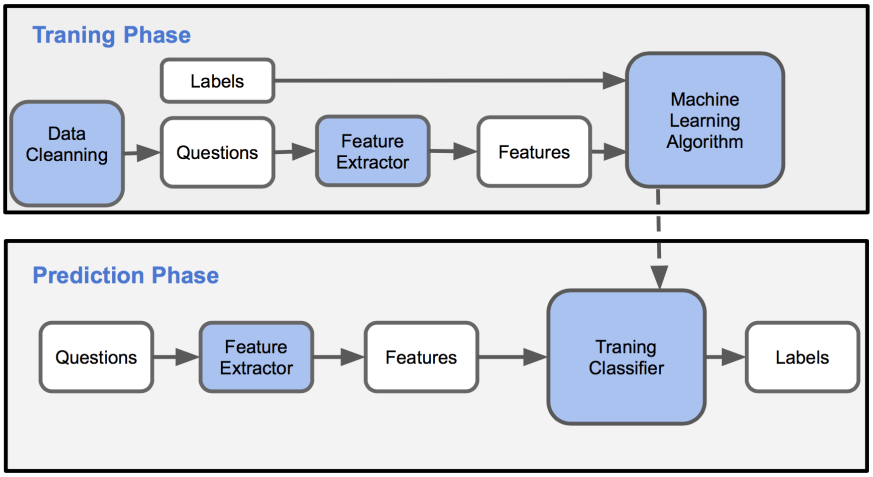

Fig. 1. Training and prediction process

3.1 Questions without Stop Words

The goal of this method is to take a string of question sentence and return a sentence without stopwords in it. For example:

Input Question: "**Do we have to** take the final exam?"
"Do", "we", "have", "to" are for stop words.
Output: "take final exam ?"

3.2 Character Length Ratio

The goal of this method is to take two strings of question sentences and return the ratio of two length. For example:

Input Question one: "Do we have to take the final exam?"
Input Question two: "No, we do not have to!"

$$ratio = \frac{length\ of\ Question\ one}{length\ of\ Question\ two} = 1.55$$

Output: 1.55

3.3 Number of Same Words or Synonyms

The goal of this method is to take two strings of question sentences and return the number of occurrences of the same words or synonyms in these two sentences. To achieve it, we built a list of lemmas of all synset of each word in one of the sentence. Then, for each word in the second sentence, we checked if that word was in the prior list. For each word in sentence two, we create a running count of similar instances in sentence one, each occurrence counted for 1.
For example:

Input Question one: "good *cats* equals dogs."
Input Question two: "is *cat* dog?"
"cat" and "cat", "dogs" and "dog" are same words, and "is" and "equals" are synonyms.
Output: 3

3.4 SOW Model

In this model, we ignored the order of text words, grammar and syntax, but only recorded whether a word appears in the text. To be more specific, we have a set, which has words that is in corpus, and we mark the length of the set as V. We also mark V as the length of the word vector. So for each word, the value of the word at the corresponding position is 1, and others remain 0.
For example:

Dictionary set: ("this","cat","dog","wants","play","to","do",the","house")
Input Text: "This cat wants to play"
Output: word vector [1, 1, 0, 1, 1, 1, 0, 0, 0]

3.5 BOW Model

This model is similar with SOW, but we also considered the occurrence of each word.
For example:

Dictionary set: (“This”,”cat”,”and”,”that”,”dog”,”want”,”play”,”to”,”do”,the”,”house”)
Input Text:“This cat and that cat want to play”
Output: word vector [1, 2, 1, 1, 0, 1, 1, 1, 0, 0, 0]

3.6 Syntactic Features

This model takes a sentence of question as input and return a list of tagged tokens. For example:

Input Text: “This cat wants to play.”
Output: [('This', 'DT'), ('cat', 'NN'), ('wants', 'VBZ'), ('to', 'TO'), ('play', 'VB'), ('.', '.')]

We can further take out the verb or the noun to compare, or we can combine this method with methods above to extract other useful information.

3.7 Basic features analysis

The above six features are some basic features that can be used for further extraction and analysis. By extracting the fundamental features of a question sentence or the comparison between two question sentences, multiple features can be combined to create both practical and additional more complex features. While a single feature may be lacking, a combined set was considered better to solve our problem.

4. Additional Features

Features that measure individual sentences attributes, SOW, BOW, and syntactic tag, are insufficient to compute the (dis)similarity between two sentences. The combination to make more advanced operations can solve this problem:

4.1 Word Mover Distance

Word Mover Distance(WMD) is a measure that allows us to assess the "distance" between two sentences in a meaningful way, even when they have no words in common. It uses word2vec vector embeddings of words. It been shown to outperform many methods in *k*-nearest neighbors classification.

WMD is illustrated below for two very similar sentences. The sentences have no words in common, but by matching the relevant words, WMD is able to accurately measure the (dis)similarity between the two sentences.

The method also uses the bag-of-words representation of the documents (simply put the word frequencies in the documents), noted as d in the figure below. The intuition behind the method is

that we find the minimum "traveling distance" between documents, in other words the most efficient way to "move" the distribution of document 1 to the distribution of document 2.

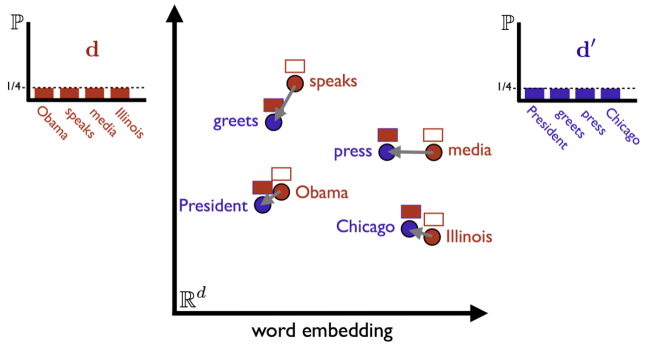

Fig. 2. Word Mover Distance Measure

To use WMD, we need some word embeddings first of all. We downloaded the word2vec-google-news-300.gz embeddings. We loaded these into a Gensim Word2Vec model class, which requires a lot of memory.

```
In [3]: import gensim

In [4]: model = gensim.models.KeyedVectors.load_word2vec_format('/Users/huaiyu/gensim-data/
   ...: word2vec-google-news-300/word2vec-google-news-300.gz', binary=True)
```

Using the wmd distance method, we could compute WMD easily.

```
In [5]: sentence_president = 'The president greets the press in Chicago'

In [6]: sentence_obama = 'Obama speaks to the media in Illinois'

In [7]: distance = model.wmdistance(sentence_obama, sentence_president)

In [8]: distance
Out[8]: 1.2129149287253465
```

4.2 Hamming Distance for SOW Vectors

Hamming distance applies to two equal length of strings or list of numbers. It means the number of positions at which the corresponding symbols are different. In other word, it measures the minimum number of substitutions which is required to transformed one string into the other. It is similar to exclusive-or operation.

For SOW vectors of a pair of sentences, we compute the hamming distance between them.

4.3 Euclidean Distance for BOW Vectors

For BOW vectors, we compute the Euclidean distance between two BOW vectors in a pair.

$$dist(a,b) = \left|\left|a-b\right|\right| = \sqrt{\left|\left|a\right|\right| + \left|\left|b\right|\right| - 2a \cdot b}$$

4.4 Cosine Distance for BOW Vectors

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors oriented at 90° relative to each other have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude.

$$similarity = cos(\theta) = \frac{A \cdot B}{\left|\left|A\right|\right| \left|\left|B\right|\right|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

4.5 Cosine Distance for Syntactic Tags

Similar to BOW model, we count the occurance for each tag for tokens in each sentence, and vectorize it. Then we calculate the cosine distance between two sentences in a pair.

4.6 Euclidean Distance for Syntactic Tags

The same as Euclidean distance for BOW features, we calculate the Euclidean distance for syntactic tags.

4.7 Hamming for Syntactic Tags

Similar to SOW model, we ignore the order of the tags for each token in each sentences, using 1 or 0 to represent whether the tag appears in the sentence. After vectorizing the tag feature, we calculate the Hamming distance between two sentences in a pair.

5. Classification Using Python Sklearn model

We extracted 9 features in total from each question. Feature examples are shown as below. Eventually we transferred this problem to a binary-class problem. If two sentences in a pair are similar questions, then the label would be 1, otherwise the label would be 0.

In advanced_feature.py, we extracted all features and stored features as new_data.csv.

	length_ratio	num_same_word	wmd	hamming_tag	distance_cosine_tag_count	distance_euclidean_tag_count
0	1.162162162162162	7	0.31446556117957025	0	0.0028235350472621823	0.0028235350472621823
1	0.52	6	0.8171886248612079	3	0.07532190152528395	0.07532190152528395
2	1.2105263157894737	4	0.6076749690953072	2	0.11747739187817197	0.11747739187817197
3	0.4166666666666667	1	1.9284997952986407	7	0.8356010126946427	0.8356010126946427
4	2.129032258064516	3	1.0192708493944636	5	0.14351411271558545	0.14351411271558545
5	0.9166666666666666	6	0.5380733748003504	6	0.18065350960129722	0.18065350960129722

sow_hamming	bow_cosine_distance	bow_euclidean_distance
1	0.05131670194948623	1
7	0.4343145750507621	2.6457513110645907
6	0.49290744716289014	2.449489742783178
15	0.8265780060951761	4.69041575982343

Fig. 3. Feature Examples

5.1 Model Selection

To solve this binary classification problem, we chose 3 models to train our data. They are Random Forest, Decision Tree and SVM.

Random Forest, as an ensemble method of supervised learning, always has great performance on classification and regression problems. It is flexible and easy to use. It constructs many decision trees during training with bagging method.

Decision Tree is a tree-like structure that each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. The paths from root to leaf represent classification rules. Decision tree algorithm is very easy to interpret from human perspective.

Support vector machines (SVMs) also is a very popular supervised learning model for classification and regression analysis. A SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

5.2 Implementation of Models

To implement Random Forest, Decision Tree and SVM, we use python build-in sklearn package. We seperated dataset into training data and test data with 4:1 ratio.

To run my code, use

>> python3 train.py

The prediction accuracy is about 0.79.

6. Neural network Method

Neural network methods have demonstrated their effectiveness in several area in natural language processing field. So it is natural for us to try it in our task. Considering the complexity of each sentence, we implement a relatively shallow neural network on the platform of keras and using tensorflow as backend.The framework of our network is shown in the Fig. 4.

The first thing to clarify is that instead of computing the similarity of two question sentence in a pair, we convert it into a classification problem with two classes: (1) question pair with the same meaning;(2) question pair with the different meaning.

As shown in Fig. 4., we adopt a siamese neural network. Each question sentence is fed into the embedding layer to produce the embedding in vector format. Then, feature representation of each sentence is obtained by Long Short Term Memory network. Following, the feature vector of each sentence is concatenated to form a new feature vector as the representation of the question pair. Finally, the new feature vector is input into two fully connected layers to generate the final classification results. In our neural network, we use embedding layer, LSTM layer, dropout layer, batchnormalization layer and dense layer. We will briefly introduce them in 6.1.

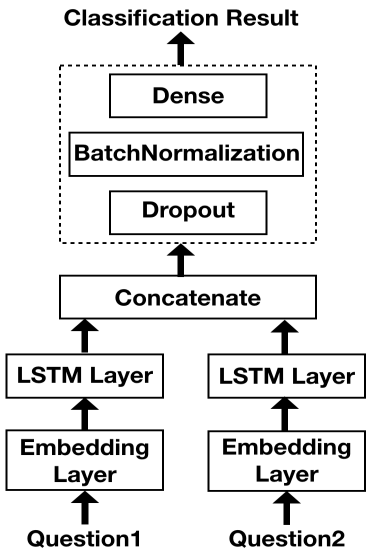

Fig. 4. Framework of our neural network

6.1 Layer Introduction

(1) Embedding layer

In this layer, we use word2vec to change original question sentence into word vectors. Considering the vocabulary size of our database and complexity to train a new word2vec model, we use the public pre-trained model which is a common way of using word2vec. We think that word2vec contains CBOW(Continuous Bag-of-Words Model) and Skip-Gram which is enough for our task. So we only use word2vec to convert words into vectors in our network.

In our project, we used the public model trained on roughly 100 billion words from Google News, which contains a vocabulary of 3 million words and phrases. The output vector length is 300.

(2) LSTM layer

Long Short Term Memory networks is a special version of Recurrent neural network which can make use of the previous information without a long time dependency problem. The structure of

LSTM is shown in Fig. 5. In the network structure diagram, we display three unit in a LSTM layer. As we can see, every unit will accept the current input and also the state from previous unit. In the unit, the current input and the previous will go through forget gate which will keep and propagate forward part of the information. LSTM is widely used in natural language processing task because the component in a sentence is not independent and we can infer the current information according to the previous information.

In our network, we input the embedding vectors into LSTM layer to extract the relations between words and integrate the independent word vectors into a sentence representation vector.

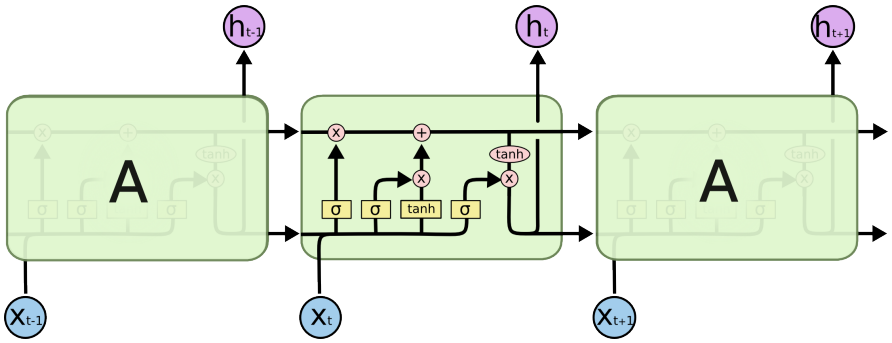

Fig. 5. Long Short term Memory network.

(3) Dropout

If the same neural network is trained on the same data too many times, it will become 'overtrained' and may doesn't have good generalization ability **(FIX SENTENCE)**. And when they see new examples, the result is not necessarily good.

Therefore, we use dropout to temporarily discard some cells in each training iteration. In this way, different neural networks are trained on each mini-batch to prevent the overfit.

(4) Batch Normalization

The usage of batch normalization is for the accelerating the convergence of neural network and reduce the complexity of neural network training. We use it before the feature vector going through the dense layer.

(5) Dense layer(Fully connected layer)

Since we have converted the problem into a two-class classification problem, our output will be a vector of 2 length. Values in the vector represent the possibility that the pair belongs to corresponding class. We need the output vector with length 2 while the sentence feature dimension is large and we need to map them to our desired output. Therefore, we use two Fully connected layer(Dense layer in keras) to realize this process.

6.2 Experiment settings

Firstly, we split the dataset into training set and test set by the ratio of 8:2. We trained 50 epoch in about half an hour on a server with GPUs. In each training process, we firstly shuffle the samples and then further split the training set into an actually training set and validation set by 9:1. After getting the model with highest validation results, we finally tested it on the test set.

6.3 Result and Analysis

In our experiment, we also perform some data preprocess such as remove stop words, retrieve the abbreviations and stemming. To verify the effectiveness of these technologies, we conduct a series of experiments:

(1) No data preprocessing
(2) Retrieve words from abbreviation
(3) Stemming
(4) Remove stop words

Table1. The result comparison of different data preprocessing technique on our neural network

Experiment Settings	Accuracy on test set
No data preprocessing	85.2%
Retrieve Words from Abbreviation	85.4%
Stemming	85.1%
Remove Stop Words	84.9%

The results in the table show that retrieve words from abbreviation boosts the performance a little, while removing stop words and stemming lower the performance. In our opinion, the reason may be that removing stop words and stemming reduce the available information for the neural network which leads to the accuracy decrease. The performance change is not so large which indicates the robustness of neural network. We also conduct a experiment introducing convolution layer, in which the training accuracy is high while the test accuracy is only 83.9%. We think the reason is that the sentence is not so long and the usage of convolution layer makes the neural network more easily overfit. Therefore, we decide to adopt our original design.

7. Conclusion

While we were designing this project, there were multiple methods and models that can be implemented. The methods and models we chose gave fairly good results(79% and 85%). We believe there are many ways to improve the outcomes. We should continue to seek effective methods for determining the similarity of question pairs. In the future, additional work can be

done on analyzing more data, extracting more features, and improving architecture for increasing accuracy of results.

References

Howardyclo. (2017). *Howardyclo/Kaggle-Quora-Question-Pairs*. GitHub. [online] Available at: https://github.com/howardyclo/Kaggle-Quora-Question-Pairs. [Accessed December 17, 2018].

Finding Similar Documents with Word2Vec and WMD. Lda_training_tips. [online] Available at: https://markroxor.github.io/gensim/static/notebooks/WMD_tutorial.html. [Accessed December 17, 2018].

Zeroassetsor. (2017).*Quora Question Pairs Organization*. Shuzigugong. Bokeyuan. [online] Available at: https://www.cnblogs.com/viredery/p/document_similarity.html. [Accessed December 16, 2018].

Colah.github.io. (2018). *Understanding LSTM Networks -- colah's blog*. [online] Available at: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ [Accessed 19 Dec. 2018].

Keras.io. (2018). *Core Layers - Keras Documentation*. [online] Available at: https://keras.io/layers/core/ [Accessed 19 Dec. 2018].

Towards Data Science. (2018). *Batch normalization in Neural Networks – Towards Data Science*. [online] Available at: https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c [Accessed 19 Dec. 2018].

Brownlee, J. (2018). *Dropout Regularization in Deep Learning Models With Keras*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/ [Accessed 19 Dec. 2018].

Workload distribution:

Guirong Liu - Neural network method implementation and experiment. I build the neural network method training and test the results. I conduct experiments comparing our architecture with neural network with additional convolution layer to choose the better architecture. And I utilize three data preprocessing method and conduct the comparison experiments to verify their effectiveness.

Linqi Wang - Data pre processing: after we got the data sets, we need to do data cleaning before other progress. Firstly, I count the total number of the data; secondly, detect missing data; thirdly detect duplicate pairs; and fourthly remove white spaces and other punctuations.

Linxuan Yang - Raw data check and experiment. I built and implemented 6 basic feature methods to extract features from the cleaned data, such as SOW, BOW, syntactic tag feature, returning sentence without stop word, length ratio as well as the word dictionary. I am also responsible for timeline organization, report format, sentence correction, grammar check, conclusion and reference citation for this project.

Yu Huai - Advanced feature engineering and training with selected models. After computing the SOW, BOW, syntactic tags for each sentence in pair, I calculated euclidean distance, hamming distance, and cosine distance between them. After extracting all features, I used several python sklearn models like Random Forest, Decision Tree Algorithm and SVM to train data and predicted whether two sentences are similar.