|  |
| --- |
|  |
| AI project report |
| Comparison of two models |
| **Group 9** |
| **崔颖琦 Kyrie 1730004002** |
| **王若彤 Edwina 1730026110** |
| **2019/12/5** |

Content

[Introduction 2](#_Toc26482143)

[1.Similarity 2](#_Toc26482144)

[1.1 Cosine similarity 2](#_Toc26482145)

[1.2 Implement 3](#_Toc26482146)

[1.3 Result 6](#_Toc26482147)

[1.4 Advantage and disadvantage 6](#_Toc26482148)

[2. Seq2seq 6](#_Toc26482149)

[2.1 RNN overview 9](#_Toc26482150)

[2.2 RNN model structure 9](#_Toc26482151)

[2.3 LSTM 10](#_Toc26482152)

[2.4 Analysis of LSTM model structure 11](#_Toc26482153)

[2.4.1 LSTM Forget gate 12](#_Toc26482154)

[2.4.2 LSTM Input gate 12](#_Toc26482155)

[2.4.3 Cell state update of LSTM 13](#_Toc26482156)

[2.4.4 LSTM Output gate 13](#_Toc26482157)

[2.5 LSTM forward propagation algorithm 14](#_Toc26482158)

[2.6 Seq2seq in Auto-reply robot 14](#_Toc26482159)

[2.7 Result 14](#_Toc26482160)

[2.8 Advantage and Disadvantage 15](#_Toc26482161)

[3. SVM model 16](#_Toc26482162)

[3.1 Implement 16](#_Toc26482163)

[3.2 Hypothesis 17](#_Toc26482164)

[4. Conclusion 18](#_Toc26482165)

[Reference 18](#_Toc26482166)

# Introduction

In the auto reply robot project, we used two different models to implement the function and an unfinished model just by test One is use similarity to fit the questions which is not a machine learning model and the other is use seq2seq model. The unfinished model use SVM to classify he questions. The following sections will explain each model and compare the results of different models in this project.

# 1.Similarity

When we got the project, we consider a simple way to finish it at first. At the beginning, I want to divide each sentence in the question set into words, and then give each question a number. For example, the first question is 1, the second question is 2 and so on. All the words in the sentences become the row of a matrix and the column is the label of the question. In each question, if the word appears in this sentence, then the corresponding position in the matrix is 1, otherwise is 0. So the matrix are formed by 1 and 0. Each column is a vector of a question and it looks like [0 0 1 1 0 1 …]. After that, when we input a new question, we change it into the vector at the same way. And then calculate the number of 1 in the vector. At the same time, we calculate the number of 1 in each question in the matrix. For example, if the number of 1 of the new question is 5, and we can fit it to the question in the matrix which the number of 1 is 5. However, this method is wrong and unreasonable. Because there are many sentences have the same number of 1 in the vector, the question we input may fit the wrong question in the matrix. So we use a new method called cosine similarity.

## 1.1 Cosine similarity

Cosine similarity measures the similarity between two vectors by measuring the cosine value of the angle between them. When two vectors have the same direction, the value of cosine similarity is 1; when the angle between two vectors is 90 °, the value of cosine similarity is 0; when two vectors point to the opposite direction, the value of cosine similarity is - 1. The result is independent of the length of the vector, only the direction of the vector.

For text matching, attribute vectors a and B are usually word frequency vectors in documents. Cosine similarity can be regarded as a method to normalize the file length in the process of comparison. And in information retrieval, each word item is given a different dimension, and a dimension is represented by a vector. The value of each dimension corresponds to the frequency of the word item in the document. Cosine similarity can give the similarity of two documents in their subject.

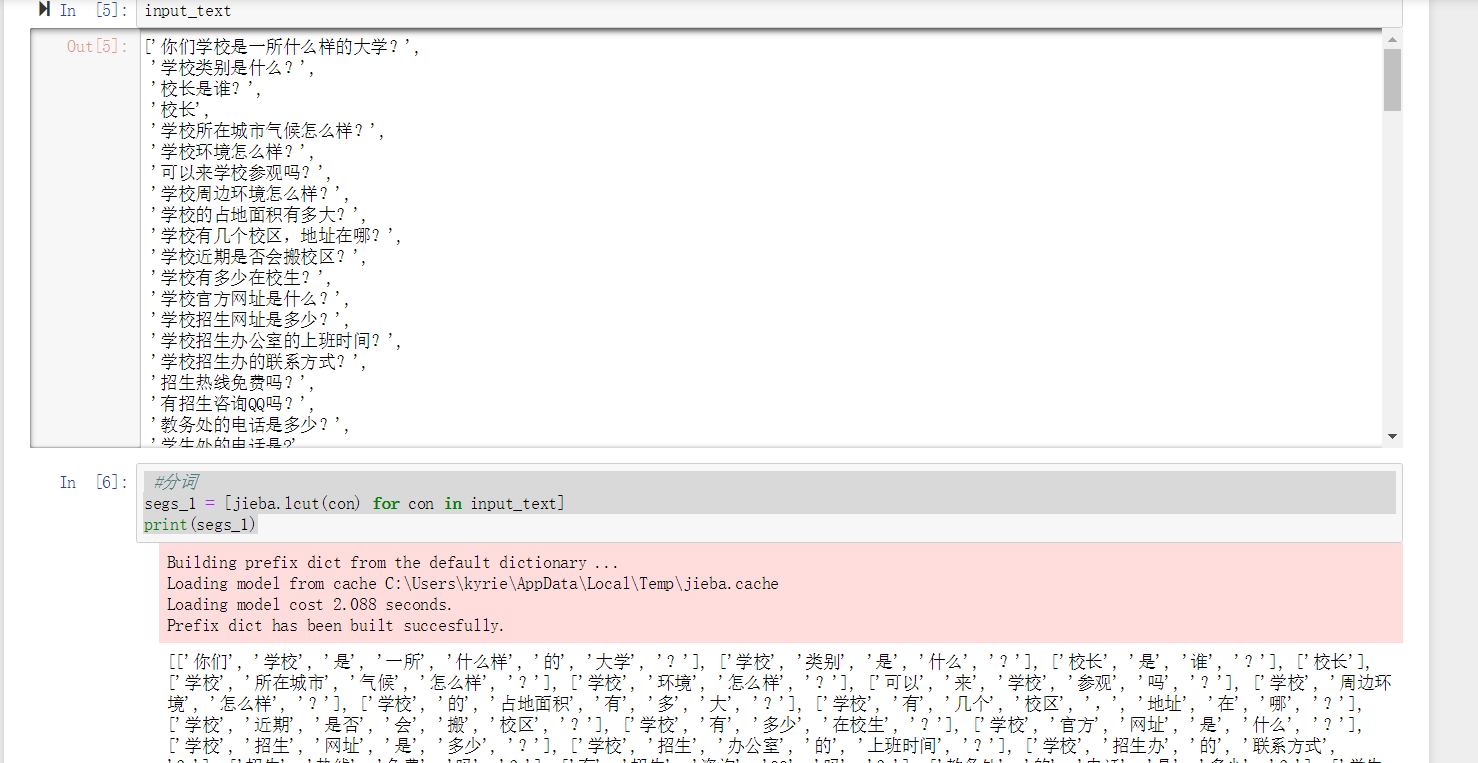
By building two vectors according to their words and calculating the cosine of these two vectors, we can know their similarity. Cosine similarity is suitable in our project.

## 1.2 Implement

Required packages: numpy, pandas, jieba, word\_token



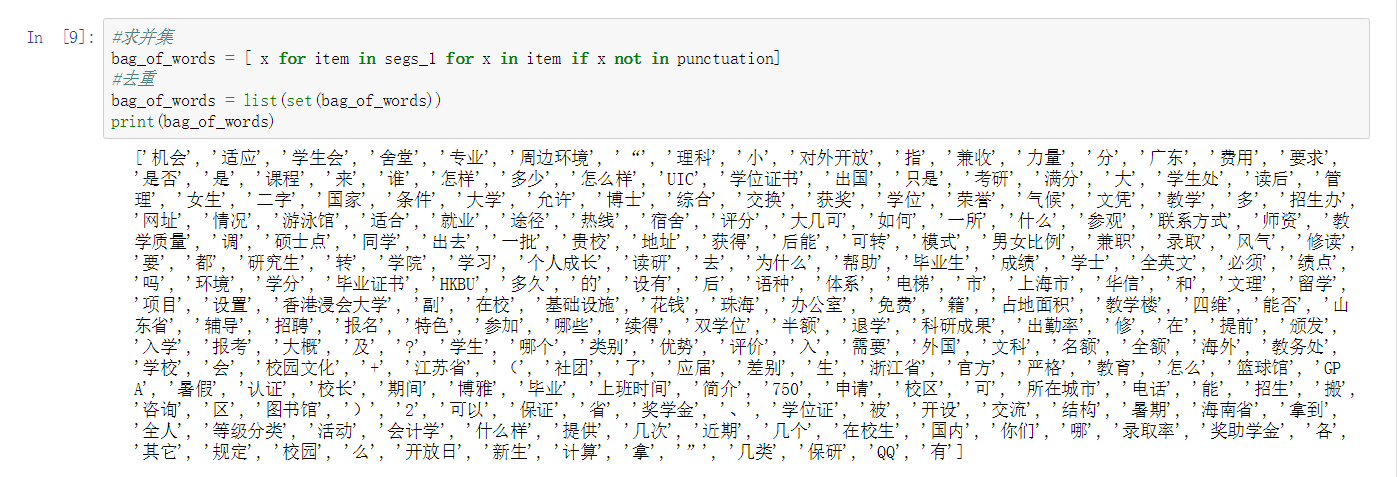
Read the question from txt file and use jieba the cut the word.



And then remove the blank space and punctuations in each sentence.



Because there are many repeated words in different sentences, we use a list to store all the words with no duplicate words.



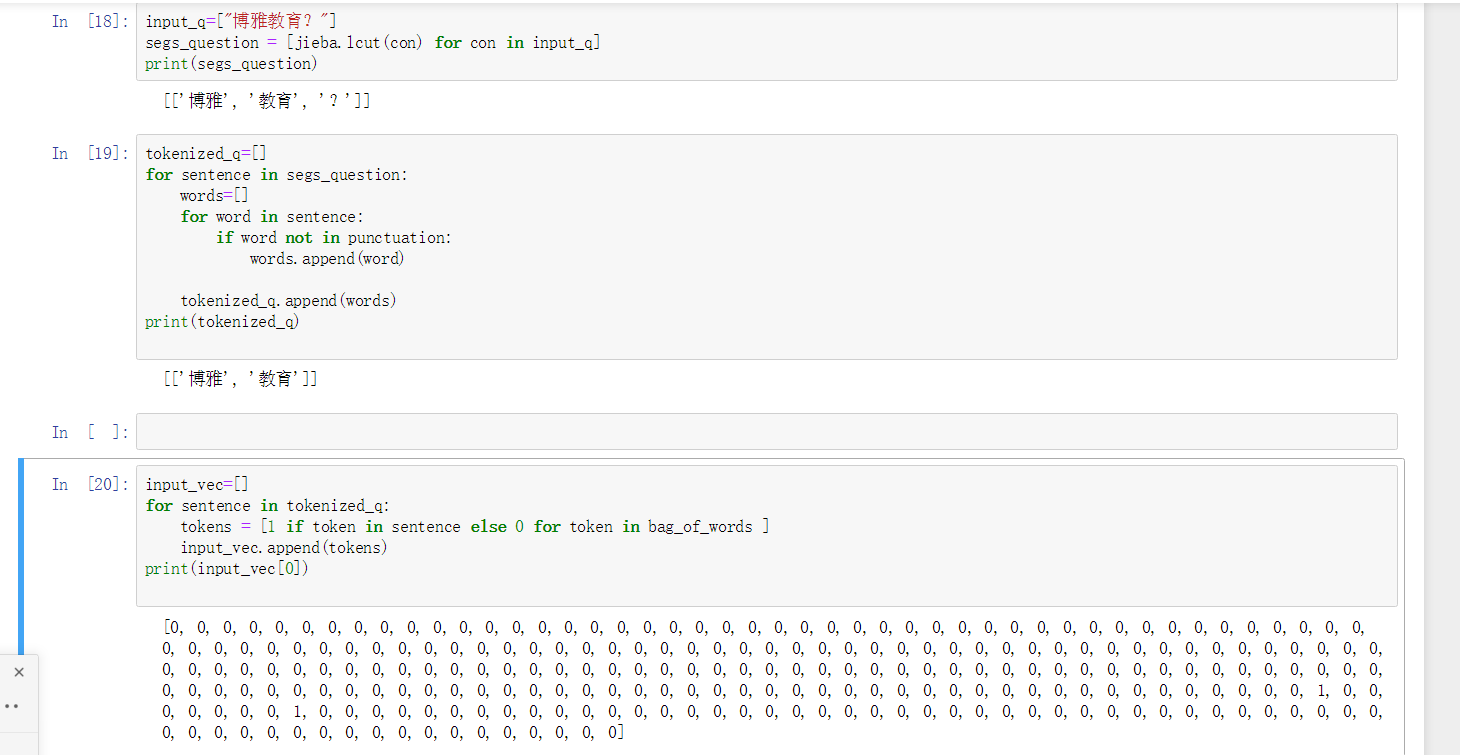
Now we can turn each sentence into vector form. If the cut word in each sentence in the set of all the words, the corresponding position in the vector is 1, otherwise is 0.



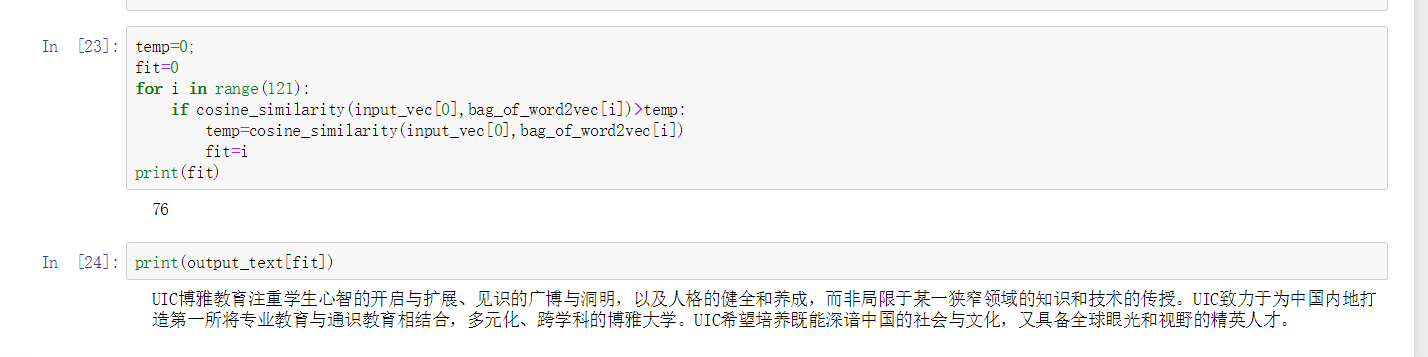
Read the answer txt file into a list and define the cosine similarity.



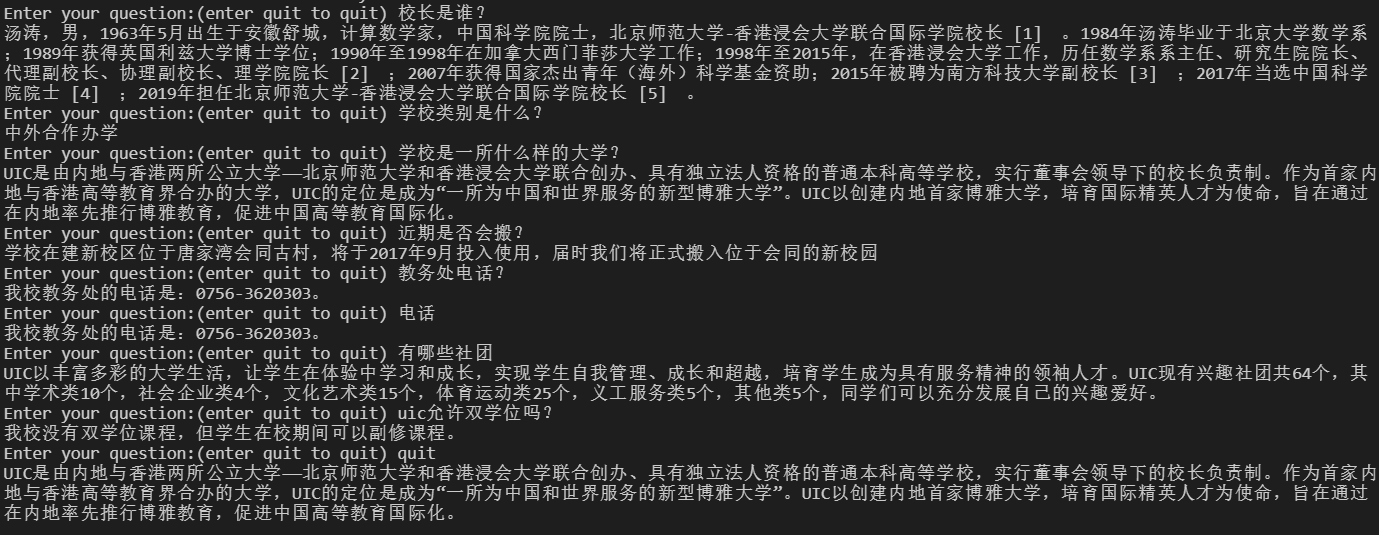
After that, when you input a question. Cut the sentence into words and then turn it into vector with the same way.



And then calculate the cosine similarity with each vector in the question set and find which one has the highest similarity. Record the position and print out the answer in the answer set.



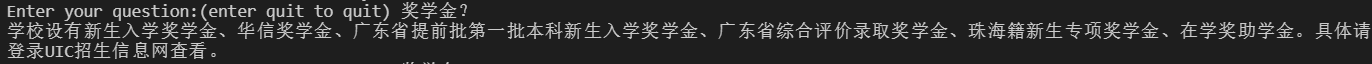
## 1.3 Result



## 1.4 Advantage and disadvantage

The advantage of this function is it don’t need a lot of time to train the data, and its accuracy is 100% if your question is in the question set or very close to the question in the question set.

The limitations are it can’t answer a new question which is not in the question set. In addition, if you ask a question which has the same similarity with different questions in the question set, it will answer you the answer which the index is smallest fit the sentence in the set. For example, if you input 奖学金, it will reply the answer of question “学校奖学金有哪几类？” because this question is No. 37 question which is earlier than other questions contains “奖学金”

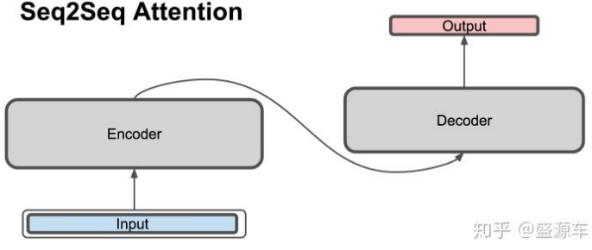


# 2. Seq2seq

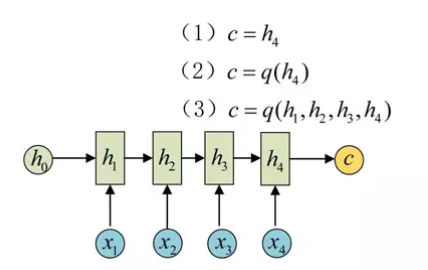
This model uses deep learning and it’s not our original model. The code is from the Internet and we just fix the parameters which make it work well on our data set.

**The seq2seq model is the most important variant of RNN: N vs M (the length of input and output sequences is different). This structure is also called encoder decoder model.**

**Encoder changes a variable length signal sequence into a fixed length vector expression, and decoder changes this fixed length vector into a variable length target signal sequence**

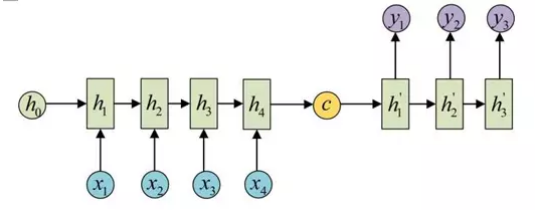


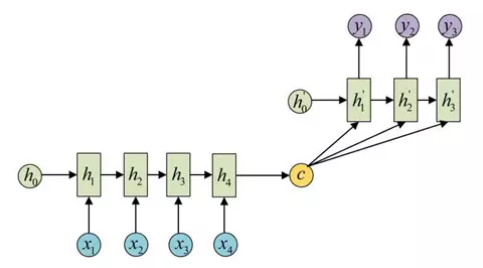
The original N vs N RNN requires the same length of the sequence, but most of the problem sequences we encounter are unequal. For example, in machine translation, the sentences of the source language and the target language are often not the same length.

Therefore, the encoder decoder structure first encodes the input data into a context vector c:  


There are many ways to get c. The simplest way is to assign the last hidden state of encoder to c. you can also change the last hidden state to get c, and you can also change all the hidden states.

After getting c, another RNN network is used to decode it. This part of RNN network is called decoder. The specific method is to input c as the previous initial state h0 into decoder:



Another approach is to use C as the input of each step:

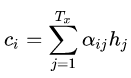
This encoder decoder structure does not limit the input and output sequence length, so it is widely used.

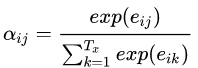
Input：x = (x1, ……, xTx）

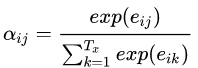
Output：y = (y1, ……, yTx)

 The encoder side accepts the word embedding and the hidden state of the previous time point. The output is the hidden state of this time point.

 The decoder side accepts the word embedding of the words in the target sentence and the hidden state of the previous time point.

 Context vector is a weighted average of hidden states output by encoder.

 The weight corresponding to the hidden states of each encoder.

 Use the hidden states of decoder and the hidden states of encoder to calculate a score, which is used to calculate the weight

 String the context vector and the hidden states of the decoder.

 Calculate the final output probability.

On the other hand, context vector and decoder's hidden state combine to calculate the probability through a series of nonlinear transformations and softmax.

## 2.1 RNN overview

RNN (cyclic neural network) is a special neural network structure, which is based on the idea that "human cognition is based on past experience and memory". It is different from DNN and CNN in that it not only considers the input of the previous moment, but also gives the network a 'memory' function to the previous content

The reason why RNN is called cyclic neural network is that the current output of a sequence is also related to the previous output. The specific expression is that the network will memorize the previous information and apply it to the current output calculation, that is, the nodes between the hidden layers are no longer connected but connected, and the input of the hidden layer includes not only the output of the input layer but also the output of the previous hidden layer.

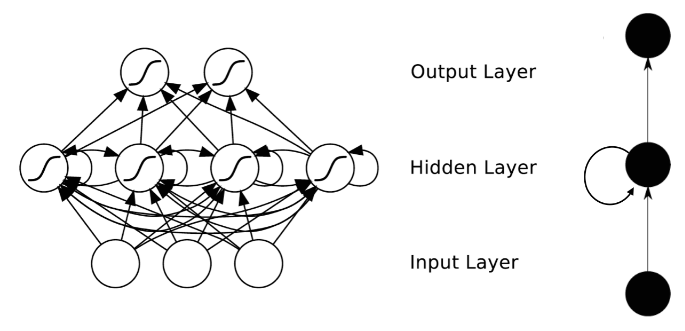
There are many application areas of RNN. It can be said that RNN can be used to solve the problem of time sequence. Here are some common application areas:

NLP: mainly includes video processing, text generation, language model, image processing.

Machine translation, machine writing novels, speech recognition. Image description generation, text similarity calculation.

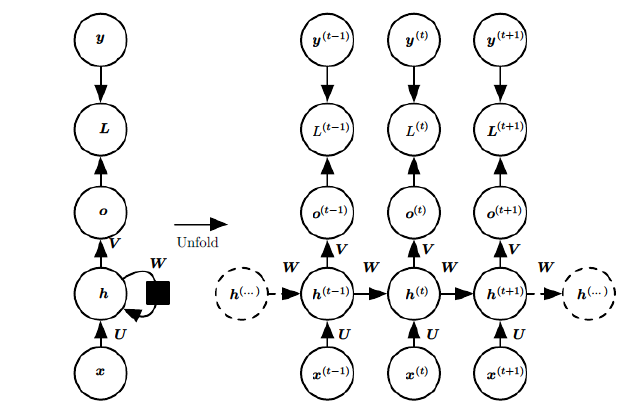
Music recommendation, online koala product recommendation, Youtube video recommendation and other new applications.

## 2.2 RNN model structure

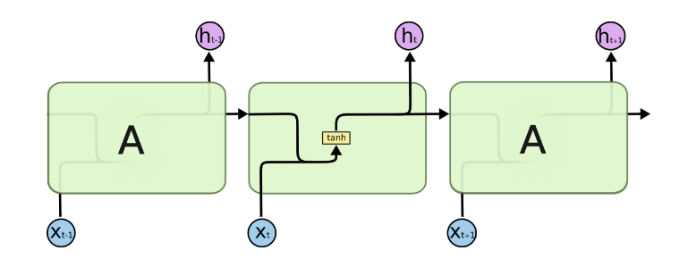
As we mentioned earlier, RNN has the function of "memory" of time. The process of realizing memory will be introduced in the following.

As shown in Figure 1, we can see that RNN hierarchy is simpler than CNN. It mainly consists of input layer, hidden layer and output layer. It also finds that there is an arrow in hidden layer to represent data cycle update, which is the way to realize time memory function.

In RNN model, we have mentioned that RNN has the following structure, each sequence index position t has a hidden state h(t)h(t).



If we omit the o(t),L(t),y(t)o(t),L(t),y(t)，RNN model can be simplified as follows:



It can be seen clearly in the figure that the hidden state h(t) h(t) is obtained by x(t)x(t) and h(t−1) h(t−1). It is used to calculate the model loss of the current layer, and on the other hand, it is used to calculate the h(t+1) h(t+1). of the next layer.

Because of the problem of RNN gradient disappearing, the big bulls have improved the hiding structure of sequence index position T. it can be said that the hiding structure is complicated by some skills to avoid the problem of gradient disappearing. Such special RNN is our LSTM.

## 2.3 LSTM

LSTM (long short-term memory) is a kind of time cycle neural network, which is specially designed to solve the long-term dependence problem of general RNN. All RNN have a chain form of repetitive neural network module. In the standard RNN, there is only one very simple structure for this repeated structure module, such as a tanh layer.

Because there are many varieties of LSTM, here we take the most common LSTM as an example. The structure of LSTM is shown in the figure below.



## 2.4 Analysis of LSTM model structure

Above we give the model structure of LSTM, and next we will analyze the internal structure of LSTM at the time t of each sequence index position.

It can be seen from the above figure that in addition to the hidden state h(t)h(t)，which is the same as RNN, there is another hidden state propagating forward at each sequence index position T, such as the long horizontal line above the figure. This hidden state is generally called cell state, which is recorded as C(t)C(t).



In addition to cell state, there are many strange structures in the LSTM diagram, which are generally called gate structures. The gates of LSTM at each sequence index position t generally include forgetting gate, input gate and output gate. Next, we will study the forgetting gate, input gate, output gate and cell state of LSTM in the figure above.

## 2.4.1 LSTM Forget gate

LSTM forget gate controls whether to forget or not. In LSTM, it controls whether to forget the hidden cell state of the upper layer with a certain probability. The substructure of forgetting door is shown in the figure below.



The hidden state h(t−1) h(t−1) of the previous sequence and the data x(t)x(t) the current sequence is input in the graph. Through an activation function, usually sigmoid, the output f (t) f (t) of the forgetting gate is obtained. Since the output f (t) f (t) of sigmoid is between [0,1], the output f ^ {(t)} here represents the probability of forgetting the hidden cell state of the previous layer. The mathematical expression is as follows:

f(t)=σ(Wfh(t−1) +Ufx(t)+bf) f(t)=σ(Wfh(t−1) + Ufx(t)+bf)

Wf, Uf, bfWf, Uf, and bf is coefficient and bias of linear relation, like RNN. σ is sigmoid activation function.

## 2.4.2 LSTM Input gate

The input gate is responsible for processing the input of the current sequence position. Its sub structure is as follows:



It can be seen from the figure that the input gate consists of two parts. The first part uses the sigmoid activation function, and the output is i(t)i(t). The second part uses the tanh activation function, and the output is a(t)a(t).

The results of the two parts will be multiplied later to update the cell state. The mathematical expression is as follows:

i(t)=σ(Wih(t−1) +Uix(t)+bi) i(t)=σ(Wih(t−1) + Uix(t)+bi)

a(t)=tanh (Wah(t−1) +Uax(t)+ba) a(t)=tanh (Wah(t−1) +Uax(t)+ba)

Wi, Ui, bi, Wa, Ua, ba, Wi, Ui, bi, Wa, Ua, and ba is the coefficient and bias of linear relationship, like RNN

## 2.4.3 Cell state update of LSTM

Before studying the output gate of LSTM, we need to look at the cell state of LSTM. The results of the forgetting gate and the input gate in front of the cell will affect the cell state C (t) C (t). Let's see how to get C (t) C (t) from cell state C (t − 1) C (t − 1). As shown in the figure below:

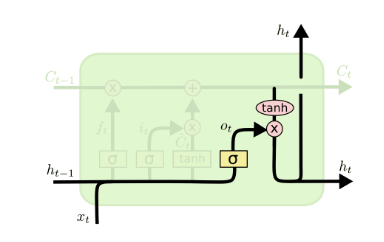


Cell state C (t) C (t) consists of two parts. The first part is the product of C (t − 1) C (t − 1) and forgetting gate output f (t) f (t). The second part is the product of I (t) I (t) and a (t) a (t) of the input gate:

C(t)=C(t−1) ⊙f(t)+i(t)⊙a(t)C(t)=C(t−1) ⊙f(t)+i(t)⊙a(t), ⊙ is Hadamard product.

## 2.4.4 LSTM Output gate

With the new hidden cell state C (T) C (T), we can see the output gate. The substructure is as follows:



It can be seen from the figure that the update of the hidden state h(t)h(t) consists of two parts, the first part is o(t)o(t), which is obtained from the hidden state h(t−1)h(t−1) of the previous sequence and the data x(t)x(t) of the current sequence, as well as the activation function sigmoid. The second part is composed of the hidden state C(t)C(t) and the tanh activation function:

o(t)=σ(Woh(t−1) +Uox(t)+bo) o(t)=σ(Woh(t−1) +Uox(t)+bo)

h(t)=o(t)⊙tanh(C(t)) h(t)=o(t)⊙tanh(C(t))

## 2.5 LSTM forward propagation algorithm

The LSTM model has two hidden states H (T), C (T) H (T), C (T). The parameters of the model are almost four times that of RNN, because these parameters Wf, Uf, bf, Wa, Ua, ba, Wi, Ui, bi, Wo, Uo, bo are available now.

The process of forward propagation at each sequence index location is as follows:

1) update forgotten door output:

f(t)=σ(Wfh(t−1) +Ufx(t)+bf) f(t)=σ(Wfh(t−1) +Ufx(t)+bf)

2) update the output of two parts of the input gate:

i(t)=σ(Wih(t−1) +Uix(t)+bi) i(t)=σ(Wih(t−1) +Uix(t)+bi)

a(t)=tanh (Wah(t−1) +Uax(t)+ba) a(t)=tanh (Wah(t−1) +Uax(t)+ba)

3) update cell status:

C(t)=C(t−1) ⊙f(t)+i(t)⊙a(t)C(t)=C(t−1) ⊙f(t)+i(t)⊙a(t)

4) update output valve output:

o(t)=σ(Woh(t−1) +Uox(t)+bo) o(t)=σ(Woh(t−1) +Uox(t)+bo)

h(t)=o(t)⊙tanh(C(t)) h(t)=o(t)⊙tanh(C(t))

5) update the prediction output of the current sequence index:

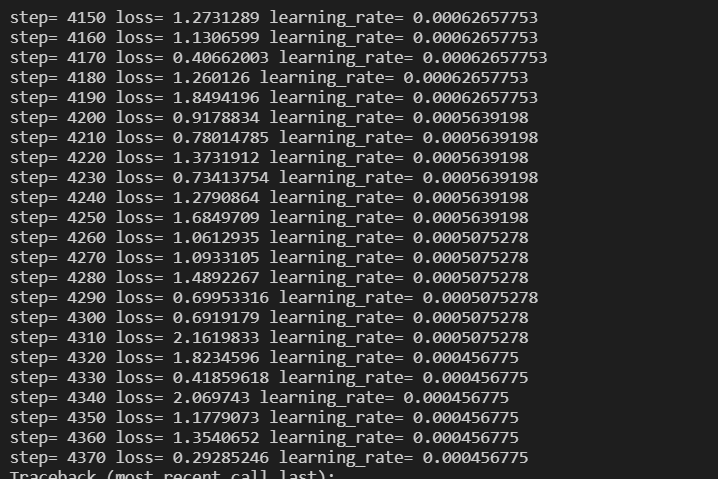
ŷ (t)=σ(Vh(t)+c) y^(t)=σ(Vh(t)+c)

## 2.6 Seq2seq in Auto-reply robot

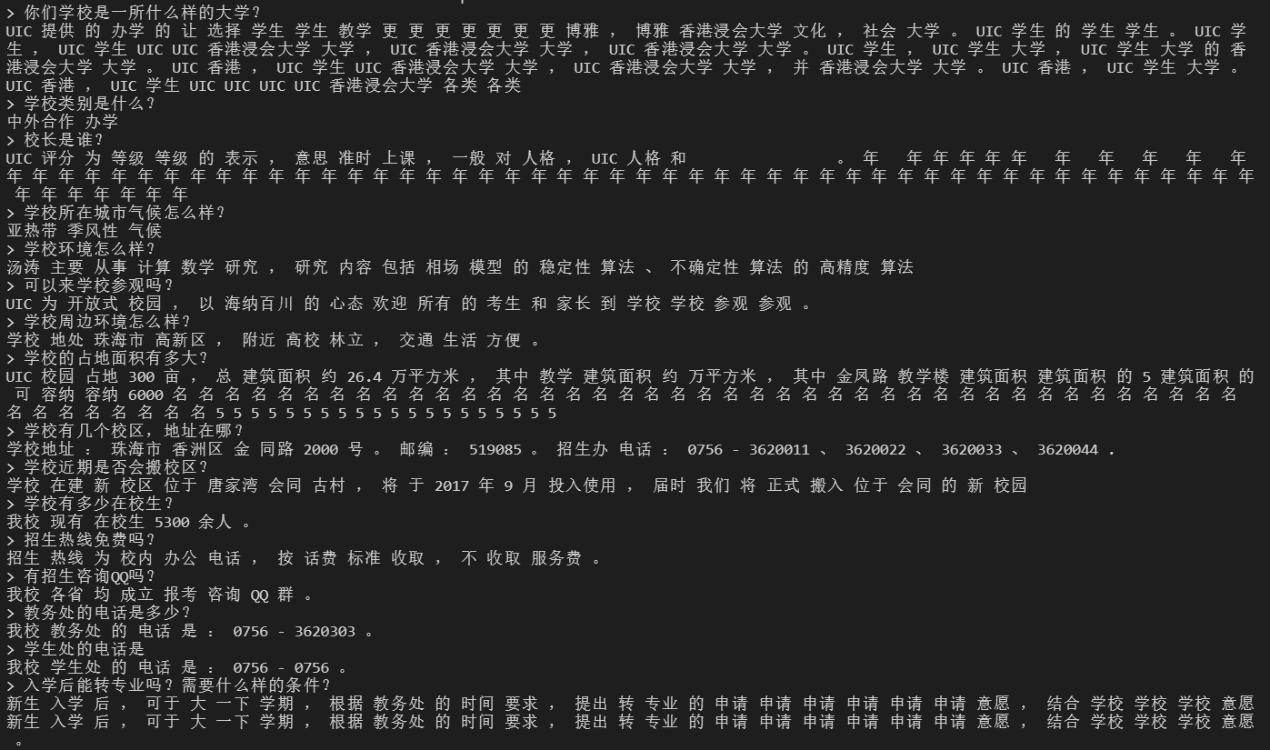
First of all, preprocess the corpus data, and use the seq2seq model to train the preprocessed corpus data to get the question answering model; then receive the questions input by users, extract the subject words according to the context information of the question answering statements and store them in the database; finally, input the processed questions into the trained seq2seq model to get the answers corresponding to the questions.

## 2.7 Result

Although this model is more complex and using deep learning, the performance is worse than the similarity model in our experiment. First, the loss of the model is too high, and the loss is not decrease step by step.



Second, the accuracy of this model is not very high although we use the same question in the training set to test. Some answers are strange and wrong.



Obviously, the loss of the model is too high, so we will get some wrong answer in the test. We think there are three reasons for the error: first is the training times is too little. We have trained the model 4370 times, if we can train over 10000 times, the loss can decrease to 0.001. Second, the parameters we fix are not the best parameters. Finally, the number of training data is not enough. We just have 120 independent data to train, however, the train data is over 10000 in normal.

## 2.8 Advantage and Disadvantage

The advantage of this model is more reliable in reality though we couldn’t prove it in our experiment. Because the model is more complex than using similarity, it will not face to the problem when you input a very simple question, the robot replies you the a not very accurate answer.

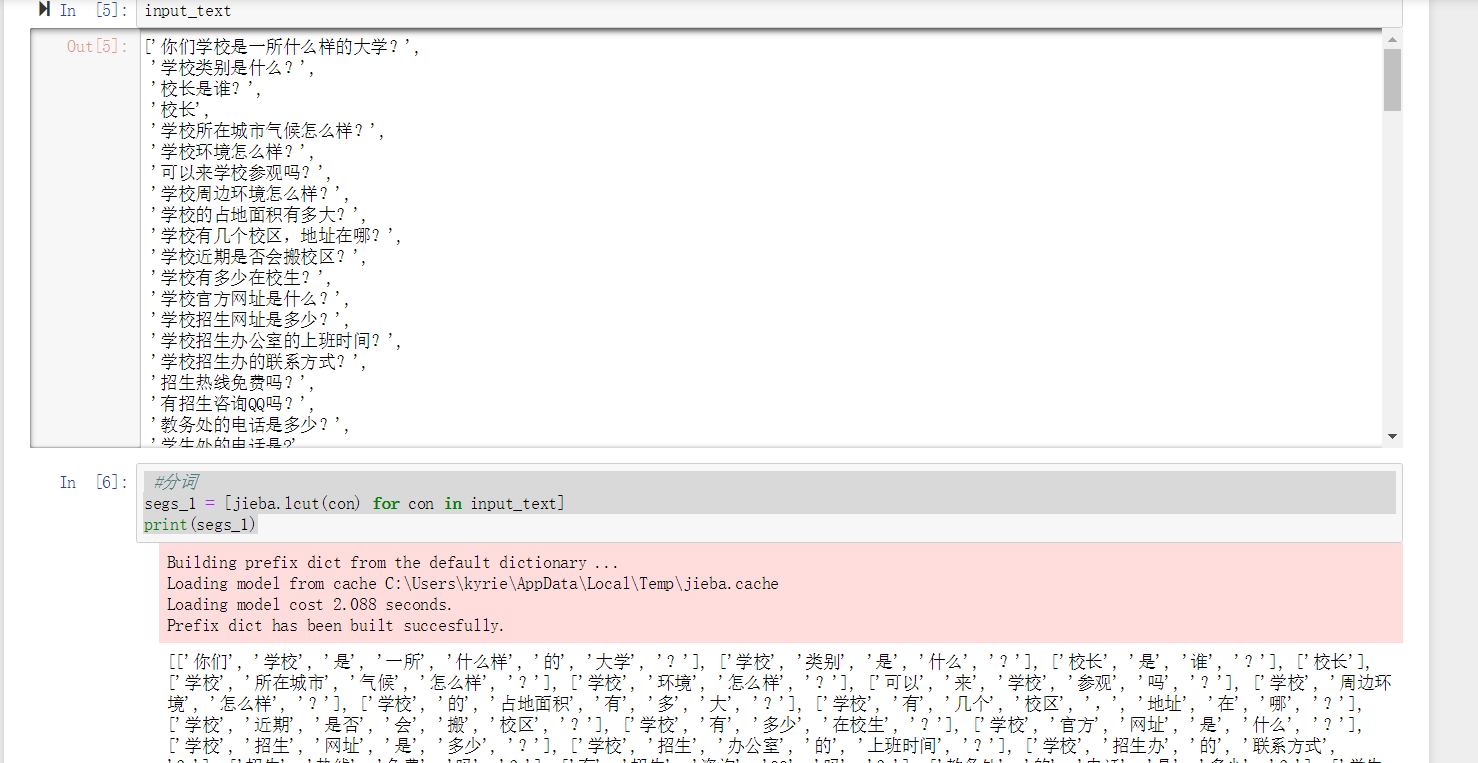
The disadvantage of this model is hard to implement. In addition, it needs lots of data and spend lots of time training. What’s more, it also has certain requirements for computer hardware configuration.

# 3. SVM model

We consider this model is not suitable for our project. And this model is unfinished in our experiment.

## 3.1 Implement

Read the question txt file and use jieba to divide each sentence into words and delete the blank space and punctuations which is the same as the step in the similarity model.

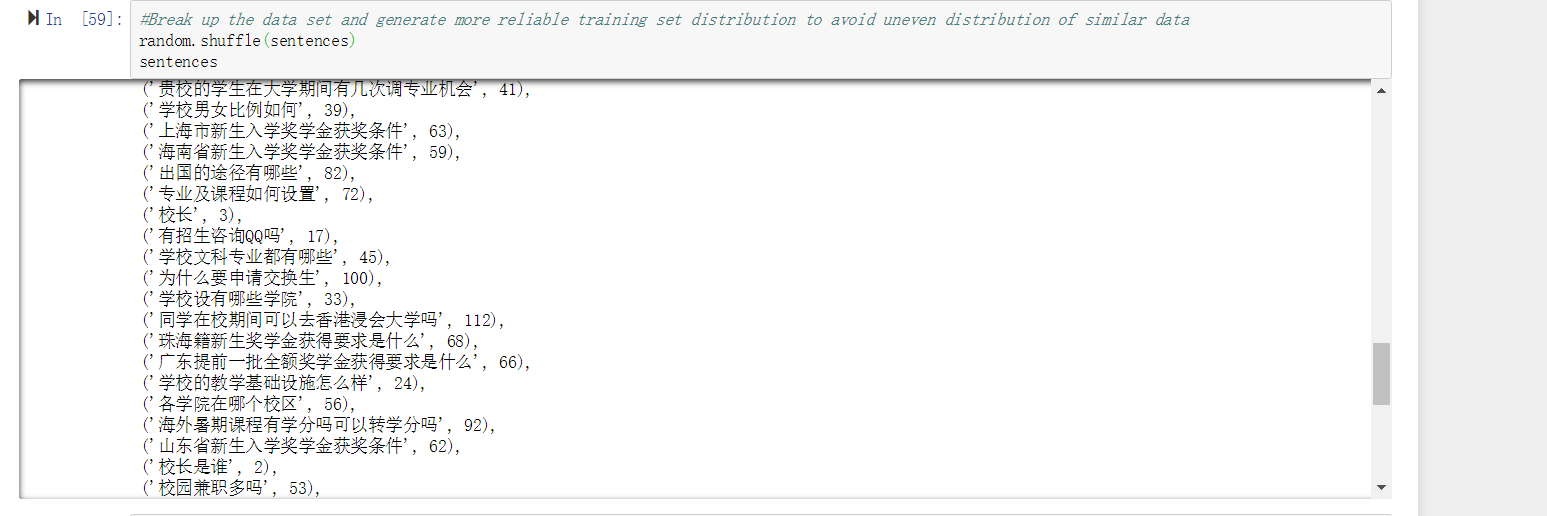




After that, we label each question.



And then reorder the data randomly and split the data into training set and test set



However, I found that SVM is a classifier which is used to classify some similar question in a same class. But our questions are independent to each other. We haven’t some questions can be seen as a class, so I think SVM is not suitable to use in our project.

## 3.2 Hypothesis

I think SVM is more likely to use to classifier the questions. For example, if we have some similar questions in the data set which can be seen as the same class, we can use SVM to do it. Maybe we can upgrade our data set to do it in the future. But I have no idea how to use SVM to implement the auto-reply robot. SVM is just a classifier, after classifying the questions, we also need to match the question inputted by users with classified data in the database which is similar with the cosine similarity.

# 4. Conclusion

Above all, we found that using cosine similarity to fit the question performs better than the seq2seq model in our experiment. So we choose the cosine similarity model as our final model in the project. In the process of doing the project, we improve our ability of writing python and simple understand a deep learning framework. What’s more, we learn some skills about how to process the data before training.

# Reference

Source: Wikipedia. (2011). *Long Short-term Memory.* From:

<https://en.wikipedia.org/wiki/Long_short-term_memory>

Reiter S. (2006). *A Combined LSTM-RNN - HMM – Approach for Meeting*

*Event Segmentation and Recognition.* IEEE. ICASSP.2006.1660362

Christopher O. (2014) *Understanding LSTM Networks by colah*. From:

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Jason B. (2017. August). Stacked Long Short-Term Memory Networks. From:

<https://machinelearningmastery.com/stacked-long-short-term-memory-networks/>

Source: CSDN. (2018). Seq2seq + Attention. From:

<https://blog.csdn.net/gt362ll/article/details/83199575>