Coursework 1 for Text Mining

Tingshan Shen 10486353 **Xin Fang** 10488936 **Anqi Guo** 10443031 **Meiyu Qi** 10550520

Abstract

Text classification is one of the most common tasks in the field of natural language processing. Based on multiple classification problems, this report presents two machine learning algorithms to question classifier that classifies questions into fine-grained classes. Implementing question classification includes the following tasks: data preprocessing, features extraction, model building, training and evaluation.

1 Introduction

This report aims to present a question classifier based on short text with multi-classification using two models: bag-of-words and BiLSTM. For using the classifiers, a question will be taken as input, and the output will be the result of the classification of the input question, which is a predefined class label. For example, when considering the question Q: *When is the summer solstice?*, the hope is to classify this question as type *NUM:date*.

This report dives into 6 parts: Sec.2 presents approaches that used to implement a question classifier, Sec.3 explains experience set-up, Sec.4 discusses the result, Sec.5 describes ablation study, and Sec.6 carries out some in-depth analyses.

2 Approaches

2.1 Initializing word vectors

For initializing word embeddings, call torch.nn.Embedding(m,n), which m represents the total number of words, and n represents the dimension of Embedding words. In our experiment, we used n=300. If the dimension is too low, the accuracy of the result will be reduced due to the low information fusion. On the contrary, if the dimension is too high, more time and space resources will be required. Through this step, we can get a large matrix, and each line of the matrix represents a word.

2.2 Using pre-trained word vectors

We can use the pre-trained word vector of Glove. The advantage is that each word can be expressed in a lower dimension, avoiding the curse of dimension. Another benefit is the introduction of prior knowledge to improve accuracy.

2.3 Models

a) bag-of-words

In this model, a sentence is represented as the bag of its words. Instead of calculating word frequency, using the following formula to calculate the sentence vectors:

$$\mathrm{vec}_{bow}(\mathrm{s}) = \frac{1}{|bow(s)|}\sum_{w \in bow(s)} vec(w) \qquad (1)$$

In this formula, s represents a sentence or a question, vec(s) is the sentence vector representation, w represents words, vec(w) is the word vector representation.

In this way, we can compute the sentence vector from the word vector.

b) BiLSTM

BiLSTM(Bi-directional Long Short-Term Memory) is a combination of forward LSTM and backward LSTM, LSTM is a recurrent neural network. PyTorch RNN classes have a bidirectional keyword flag, so we only need to add bidirectional=True when defining the model.

3 Experience set-up

3.1 data

| - data
    | - train.txt
    | - dev.txt
    | - test.txt
    | - glove.txt
    | - output.txt

3.2 preprocessing steps:

1. lowercase tokens

To unify the text, use words.lower() to lowercase tokens.

2. remove stop-words

Some stop words like "what/why/how/when..." are very important in this classification problem, so we remove part of stop words like "a/but/on...".

3. remove punctuations

Commonly used punctuations include "?,`'" are put in a list. By using for loop and put if-else judgment on every word in a sentence, we remove these punctuations.

3.3 performance metric:

In our models, we use time consumption, log-loss, and accuracy to evaluate the effectiveness of the model. It is an essential part to find out how effective the model is based on the performance metric. For such a classification problem, AUC (Area under Curve) is also commonly used to evaluate the performance.

4 Result

After parameter tuning and training, we obtained the final accuracy of the two models.

Model	Accuracy
bow	69%
bilstm	74.2%

Table 1: Classification accuracy on the test set using two models

It can be seen from the result that the accuracy of the two models is similar. For the impact on accuracy, there are other factors that can slightly affect the result, like using initialized word embeddings or using pre-trained word embeddings and choosing fine-tuning or freezing. These different settings can affect the accuracy of the result.

5 Ablation study

5.1 Fine-tuning vs. Freezing

When loading word embeddings, set "freeze=False" for fine-tuning and set "freeze=True" for freeze word embeddings. We can get a comparison below.

Model	freeze=False	freeze=True
bow	55.80%	66.0%
bilstm	75.20%	74.2%

Table 2: Classification accuracy on the test set using fine-tuning and freeze

5.2 Initialization vs. Pre-trained

In the word embedding step, we compared initializing word vectors to using pre-trained word vectors.

Model	initialize	Pre-trained
bow	66.0%	55.8%
bilstm	74.2%	69.8%

Table 3: Classification accuracy on the test set using two word embedding methods

From the results, the accuracy by initializing word embeddings is higher than using pre-trained word vectors.

6 In-depth analyses

6.1 Using part of the training set

If using only part of the training set, we will miss some labels, and the accuracy of the results will be greatly reduced.

6.2 The difficulty of classification

Generally speaking, the accuracy of our classifier is satisfactory. However, in some cases, the classifier does not perform well.

Here are some examples misclassified by the classifier.

Q: *What river runs through Rowe, Italy?*

This sentence is classified as *DES:def*, but the correct label is *LOC:other*. Since the classifier failed to relate the word “river” to a location

because the word river was not in the location list.

Q: *What is the life expectancy for crickets?*

The correct label is *NUM:other*, but the result returns *HUM:ind*. This example illustrates the shortcomings of the classifier in analyzing syntactic structure. Our feature sensors focus on "crickets" instead of "life expectancy".

Q: *What person's head is on a dime?*

The classifier outputs *ENTY:other* instead of *HUM:ind*. The classifier does not sufficiently identify "person's head". In fact, both categories are reasonable, other entity is also acceptable, but individual is closer to the meaning of the sentence.

6.3 Confusion matrix

The Confusion matrix is one of the most intuitive and easiest metrics used for finding the correctness and accuracy of the model. It is used for the classification problem where the output can be of two or more types of classes.

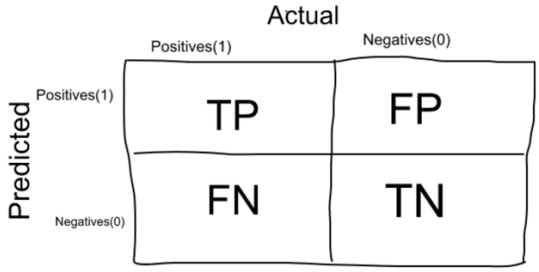

Figure 1: Confusion matrix

By applying the confusion matrix to the multi-classification problem, each category is treated as "positive" and all other types as "negative". We can calculate the accuracy and recall rate for each category based on the test set.

Class	ABB	ENT	DES	HUM	LOC	NUM	Total	recall
ABB	6	0	1	0	0	0	7	0.86
ENT	0	28	0	2	5	0	35	0.80
DES	3	37	133	3	13	24	213	0.62
HUM	0	18	2	59	6	5	90	0.66
LOC	0	10	1	0	57	4	72	0.79
NUM	0	1	1	1	0	80	83	0.96
Total	9	94	138	65	81	113		
precision	0.67	0.30	0.96	0.91	0.70	0.71		

Table 4: Confusion matrix on the test set (based on broad headings of each class)

Calculating the data in the table, we can get the average precision is 0.71, average recall is 0.78. For demonstration purposes here, we only show the confusion matrix for large labels. We can also calculate the precision and recall of each subdivided class through confusion matrix.

References

Chen, T., Xu, R., He, Y., & Wang, X. (2017). *Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN.* Expert Systems with Applications, 72, 221-230.

Fujisaki, T., & Srinivasan, S. (1999). *U.S. Patent No. 5,963,666.* Washington, DC: U.S. Patent and Trademark Office.

Subramanian, V. (2018). *Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch.* Packt Publishing Ltd.

Xin Li, Dan Roth. (Aug., 2002). *Learning Question Classifiers*. COLING'02.