Supervised Chinese Text Classification

Peng Li, 10175501102@stu.ecnu.edu.cn

October 23, 2019

Abstract

In this experiment, I conducted a survey on the supervised Chinese text classification task. A typical text categorization system usually includes phases such as text preprocessing, feature extraction, feature conversion, classifier selection, and evaluations. I have combed and summarized these steps in this report. In addition, I completed a classic text classification task using the Logistic Regression Algorithm. Problems such as data imbalance have been analyzed and addressed. I haven't used the Deep Learning algorithms because of the long training time, but those can be exploited in the future.

Introduction

Text classification problems have been widely studied and addressed in many real applications over the last few decades. Especially with recent breakthroughs in Natural Language Processing (NLP) and text mining, many researchers are now interested in developing applications that leverage text classification methods. Most text classification and document categorization systems can be deconstructed into the following four phases: Text preprocessing, feature extraction, feature conversion, classifier selection, and evaluations. In the first part of the report, I have outlined the general steps and common algorithms for implementing the Chinese text categorization system, which may give you a general understanding of the Chinese text categorization task.

In the second part of the report, I will detail a text classification system that I implemented. Although this system is relatively simple, it still gives you a deeper understanding of the entire process. This experiment not only uses the classical text feature representation method such as vector space model (VSM), but also uses pre-trained word vector to express the semantic information of the document. Experiments have shown that using pre-trained word vectors not only significantly improves system performance, but also speeds up the model training process.

Supervised Chinese Text Classification

In this section, I will give an overview of the general process of Chinese text categorization. Kowsari et al. (2019) gives a more comprehensive overview of English text classification algorithms.

Text Preprocessing

In the process of text preprocessing, we must first conduct exploratory data analysis (EDA). We need to know how many categories our corpus has, how many documents are in each category, how many words per document, and what types of characters are inside the document. These will have a crucial impact on the subsequent data processing. For example, if we find that the data category is not balanced, we should probably consider sampling and other technologies.

Next, it should be the steps of data cleaning, word segmentation, and removal of stop words. We should process the data into a structured form to facilitate our later processing. Techniques such as regular expressions are used in this process. These are common data processing methods, we will not go into details, but you need to pay attention to the difference between Chinese text preprocessing and English text preprocessing.

Feature Extraction

The text feature extraction algorithm can be based on levels of characters, words, sentences, documents, and the like. Next we will mainly introduce word-based feature extraction, and more methods can refer to Kowsari et al. (2019).

If we don't convert Chinese characters into continuous vectors, we can use algorithms such as Naive Bayes to classify the text. We can model the text using the n-gram model. But we tend to represent characters as vectors, because this allows more machine learning algorithms to be used. Of course, we can extract some keywords before converting them into vectors. We can use the algorithm based on TF-IDF, information gain, chi-square statistic, or mutual information to extract keywords.

Vector space model or term vector model is an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers, such as, for example, index terms. It is used in information filtering, information retrieval, indexing and relevancy rankings. Its first use was in the SMART Information Retrieval System. Vector space models can also be chosen to build based on characters, words or n-grams. Some weight-based methods are also often used, such as word frequency, entropy, TextRank value, and so on. But the vector space model has a fatal weakness - its word vector dimension is often very high.

In recent years, with the development of deep learning, the distributed word vector has been extensively studied.

Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space.

Word2vec was created and published in 2013 by a team of researchers led by Tomas Mikolov at Google and patented.The algorithm has been subsequently analysed and explained by other researchers. Models like GloVe, FastText, ELMo, GPT and BERT refreshed the list of top conferences again and again.

In order to represent a word vector as a document vector, we can use an average or weighted average operation. Of course this is just an simple way, but it tends to be faster.

Feature Conversion

Feature conversion is mainly for vector space model which doesn't contain semantic information. We can use algorithms such as PCA to convert high-dimensional vectors to low-dimensional. We can also model text using topic models, such as LSA, NMF, PLSA, LDA and so on. These algorithms can be seen as dimensionality reduction of vector space model. Auto-encoder is also a means of dimensionality reduction, and it is somewhat similar to PCA.

Classifier

After the document is represented as a vector, many machine learning algorithms can be used. Traditional machine learning algorithms include kNN, Naive Bayes, Decision Tree, Logistic Regression, SVM, Boost, DNN and so on. After the rise of deep learning, many deep learning models have also been used to complete the text classification tasks. These models include CNN Based Models, RNN Based Models, Self-Attention Based Models and so on. A few SOTA models can be found in here.

We will focus on the Logistic Regression model, which can be seen as a transition between machine learning and deep learning. For an introduction to logistic regression, you can refer to it here.

Evaluation

For classification tasks, commonly used model evaluation methods include accuracy, recall rate, f1 value, etc. For multi-classification tasks, we can use macro-average and micro-average methods.

Experiment

I implemented a simple text classification system, and the details of the experiment have been described as follows.

Environment

- Python: 3.6.4
- Numpy: 1.14.2
- Pandas: 0.22.0
- Spicy: 1.0.0
- Sklearn: 0.21.3
- Imblearn: 0.5.0

Data Set

The data set contains 19,637 documents, of which 9,804 are training sets and 9,833 are test sets. There are 20 categories of documents in the entire data set. Our task is to let the machine automatically judge its category after giving a document. These ten categories are: 'Economy', 'Computer', 'Sports', 'Environment', 'Politics', 'Agriculture', 'Art', 'Space', 'History', 'Military', 'Education', 'Transport', 'Law', 'Medical', 'Philosophy', 'Mine', 'Literature', 'Energy', 'Electronics' and 'Communication'. In the experiment I corresponded each category to a number(see table 1).

Table 1: Category Name to Number

Category Name	Corresponding Number	Category Name	Corresponding Number
Economy	0	Education	10
Computer	1	Transport	11
Sports	2	Law	12
Environment	3	Medical	13
Politics	4	Philosophy	14
Agriculture	5	Mine	15
Art	6	Literature	16
Space	7	Energy	17
History	8	Electronics	18
Military	9	Communication	19

Text Preprocessing

In the data preprocessing stage, I first counted the number of various types of documents. I used the 'matplotlab' package to visualize the results(see Figure 1)).

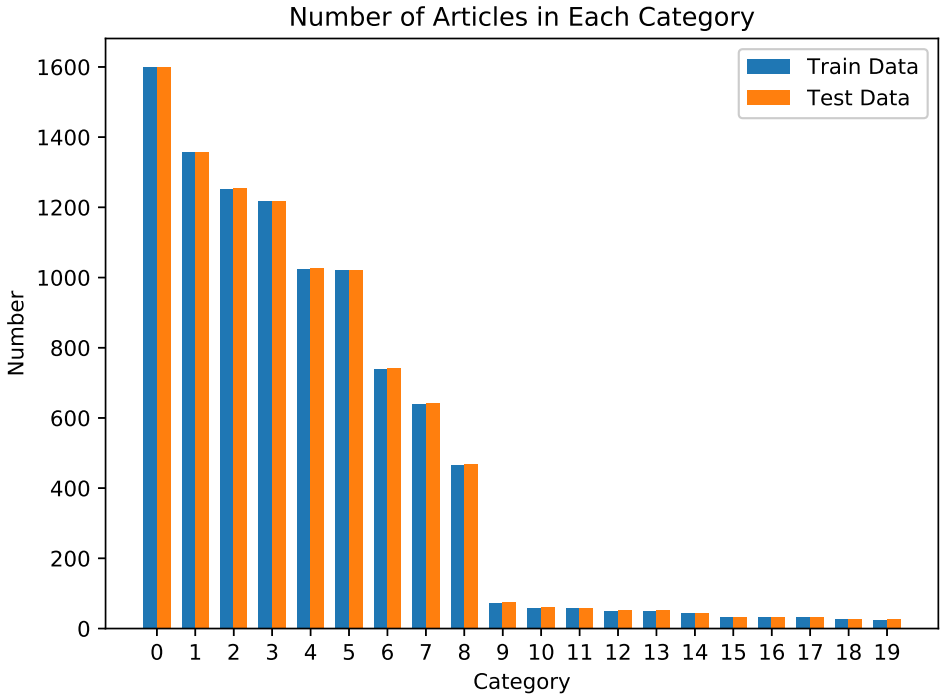

Figure 1: Number of Articles in Each Category

It can be seen from the figure that the number of articles in different categories varies greatly. So we may need some oversampling methods to solve this data imbalance problem.

In addition, during the data pre-processing phase, I also cleaned the data. I only extracted the Chinese in the document and deleted the letters and other symbols. I also removed the stopwords.

Considering that the length of the documents in the dataset is mostly long, in order to improve the training speed, I extracted the first K keywords for each article at this stage.

Feature Extraction

In this experiment I used four ways to encode the document. The first is a vector space model based on word frequency weighting. The second is a vector space model based on TF-IDF weighting. The third and fourth are based on the pre-trained word vectors. The third is to average each word vector to get the vector representation of the document. The fourth is based on TF-IDF weighted document vector representation. Pre-trained word vector comes from here.

The TF-IDF-based document vector has not been trained in the end because the space required by the model is too large. But a toy test can be run successfully.

Feature Conversion

For the vector space model, I mainly want to reduce the dimension of the word vector. I used the dimensionality reduction tool in the Sklearn, but I didn't succeed. The main reason is that my computer has insufficient memory.

In order to solve the data imbalance problem, I have oversampled the training data. There are 32,000 training samples after sampling.

Classification

I used the interface of the Logistic Regression algorithm in the Skearn. It should be notice that if you set the parameter 'multiclass' as 'multinomial', you have to set the solver as 'lbfgs'. The parameter 'C' stands for 'Inverse of regularization strength', it must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

Initially I compared the effects of several word vectors using a simple method. I found that the third model is slightly better than the first and second. Then I did further training on the third model.

Results and Analysis

In the first, I set the 'max_iter' as 100 when I using the Logistic Regresion model. The results tested on the test data set can be seen in table 2.

Table 2: Comparison Between The Three Type of Vector (iter=100)

Vector	size	f1_micro	f1_macro	training time
Count2Vec	(19637, 62183)	0.8954*	0.7085	174.1270
Tfidf2Vec	(19637, 62183)	0.8789	0.4409	181.8859
avg_Word2Vec	(19637, 300)	0.8839	0.6468	2.6879*
avg_Word2Vec + oversampling	(32000, 300)	0.8704	0.7254*	9.9719

From the table 2 we find that the model based on pre-training word vector is slightly better than the other two, especially its training time is one percent of other models. In addition, we found that the macro average of the model was significantly improved after using the oversampling technique, and the training time hardly changed.

Then I split the training set and divided it into a training set and a validation set. I looked at the effect of the model by constantly changing the number of iterations. The result can be seen in Figure 2

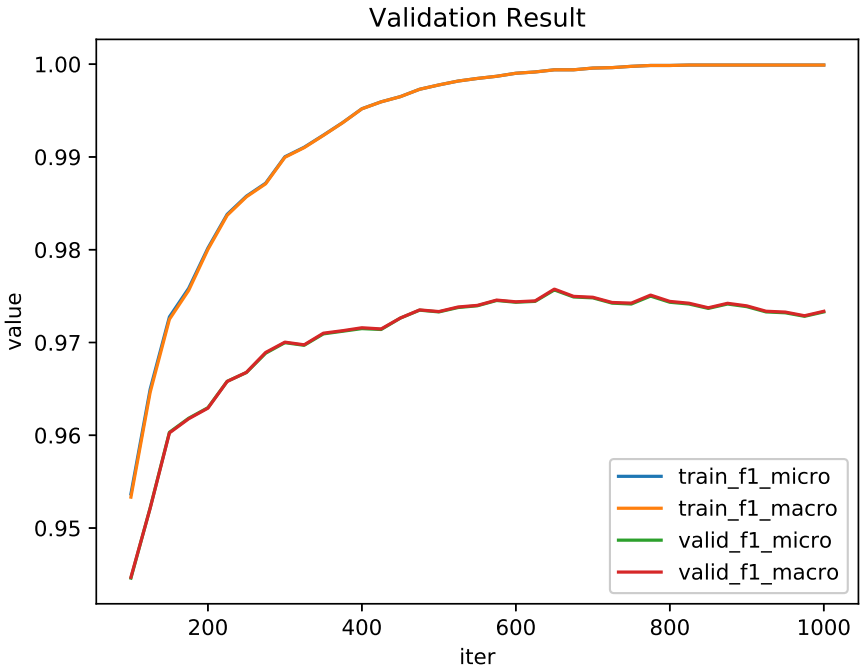

Figure 2: Validation Result

Finally, I set the number of iterations to 625, and the final experimental result is shown in the Table 3. We can see that the final model is better than the previous model in almost every indicator.

Table 3: Final Result

Vector	size	f1_micro	f1_macro	training time
avg_Word2Vec + oversampling	(32000, 300)	0.9012*	0.7964*	0.7891

Conclusions

Through this experiment, I have a more comprehensive and in-depth understanding of the text classification task. I continued to adjust the model so that the final model achieved a relatively high level. The

main difficulty in this experiment is that the model takes up too much memory and the data is not balanced. I finally solved the above problem by choosing the right storage method and using oversampling technology. Of course, this is just a toy-level system, and there are many more advanced implementations. In the future, deep learning models can be used to further improve the system's effectiveness.

References

Kowsari, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L. E., and Brown, D. E. (2019). Text classification algorithms: A survey. *CoRR*, abs/1904.08067.