CSE 543T Project Report

Comparison and Optimization for Algorithms in Text Classification

Jiayang Tian, author email: jiayang@wustl.edu

Engineering and Applied Science School,

Washington University in St. Louis

Abstract

For this project, I mainly test different machine learning algorithms' performance on a Chinese news text dataset for text multi-classification. Two major class of algorithms are used in this project. (Linear model and nonlinear model). For each model, we track the time efficiency and performance metrics, then an comparison among them are proposed. Besides, two major text feature extraction methods are used in the project. Interestingly, one of the method could significantly improve the performance of some certain algorithms.

1 Introduction

1.1 Motivation

One of the classic tasks for NLP(Natural Language Processing) is text classification, also known as document classification[1]. This kind of tasks' purpose is to assign a pre-defined label to the document. It has a very wide range of application, such as spam email detection, sentiment polarity analysis and so on.

Usually, two stages are involved in the process, which are feature extraction and classification. In the first stage, some particular word combinations such as n-grams, term frequency, and inverse document frequency of the phrase can be used as features.[2]. In the second stage, different classification algorithms have their own hypothesis and decision boundaries. Hence, those methods have their own advantages and suitable use case.

This report will focus on the second stage, which is classification stage. Two major categories of classification algorithms are implemented, which are linear algorithms and non-linear algorithms.

1.2 Existing Research Overview

A common approach for text classification is to use N-gram[2] and their term frequency-inverse document frequency (TF-IDF)[3] as feature representation of the raw text, and some linear classification models such as Linear SVM , Logistic Regression, Naive Bayes[25] as classifiers.

However, recently many researchers[4][5] using nonlinear algorithms to solve the problem, especially deep learning model. The convolutional neural networks (CNN) and recurrent neural networks (RNN) has also been pronounced as a very good solution for text classification problems.

1.3 Dataset Description

In this project, I used an very popular open-source Chinese corpus – Sohu News Data(SogouCS)[http://www.sogou.com/labs/resource/cs.php]. It consists of approximately 1.6GB raw news-related text in .xml format. FigureBelow is an example of the original dataset.

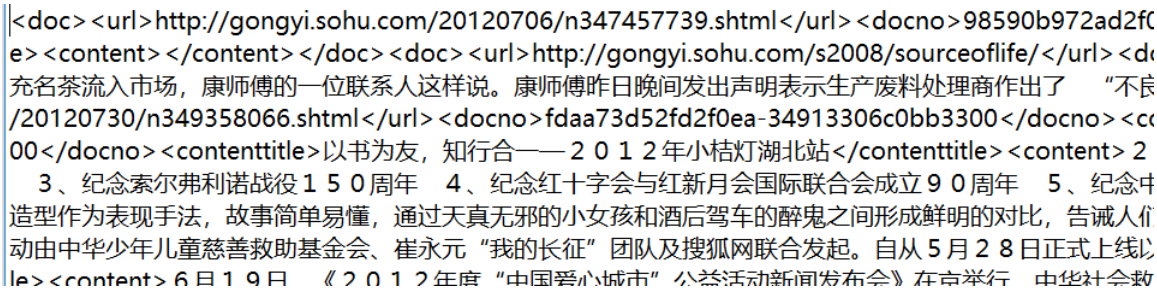


Figure: Original Dataset

For the purpose of research, I did some pre-processing of the raw dataset, such as labeling, text data extraction, stopping words removing, word segmentation(With Jieba library) and so on.

Finally I obtained the original training materials like the figure below.

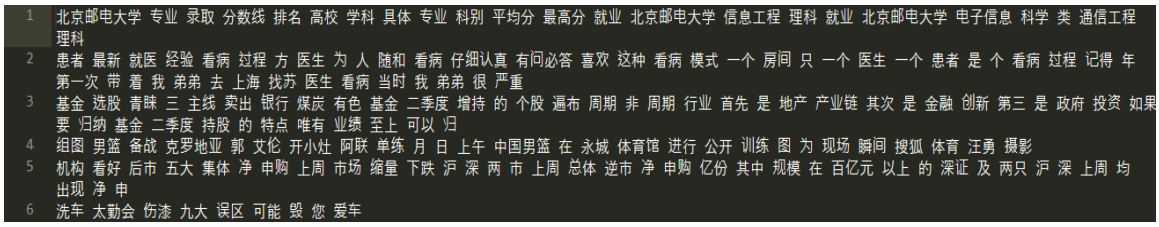


Figure: Original Training Materials

1.4 Report Structure

In the first and second chapters, structure of the project and background knowledge are provided for the report. In the third chapter, I go through how I implement the project with details in data processing and model building. In the fourth and fifth chapter, experiment results are listed, and I did some comparison and discussions based on their performance. Finally, conclusions and further research directions are listed in the sixth and seventh chapter.

2 Related Theory and Practice

2.1 Text Feature Extraction

2.1.1 TF-IDF

In information retrieval, TF-IDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word to a document in a collection or corpus. The TF–IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.[6]

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}} \qquad \mathrm{idf_i} = \lg \frac{|D|}{|\{j : t_i \in d_j\}|} \qquad \mathrm{tfidf_{i,j}} = \mathrm{tf_{i,j}} \times \mathrm{idf_i}$$

tf – term frequency, idf – inverse document frequency, n – term amounts, D – document amount

2.1.2 Word2Vec

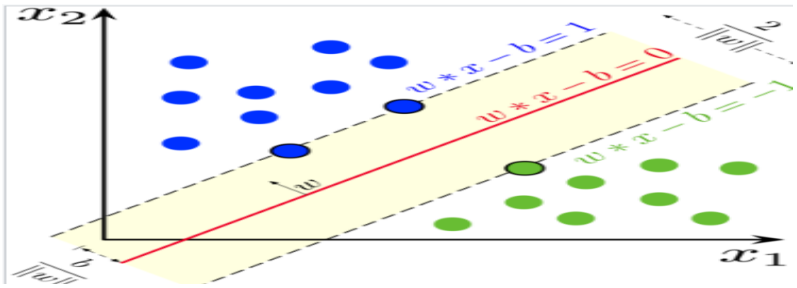
Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred 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 in close proximity to one another in the space.[7]

2.2 Linear Algorithms

2.2.1 Linear SVM

Linear SVM aims to construct a hyperplane or set of hyperplanes in a dimensional space, which can be sperated by with maximum margin between different classes. Minimizing the Objective function of SVM below is the core idea of this classifier.

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(\vec{w}\cdot\vec{x}_i - b)\right)\right] + \lambda\|\vec{w}\|^2$$



2.2.2 Naïve Bayes

In this project I will use multinomial naïve bayes classifier. The multinomial naive Bayes classifier becomes a linear classifier when expressed in log-space[8].

2.3 Nonlinear Algorithms

2.3.1 Convolutional neural networks

Convolutional neural networks are usually used for image classification. Recently some papers[9] show that CNN is also a good solution for short text classification.

The figure below is a typical CNN structure: Convolutional layer for local feature extraction, max-pool for reducing parameters, and the fully connected layer is responsible for final classification.

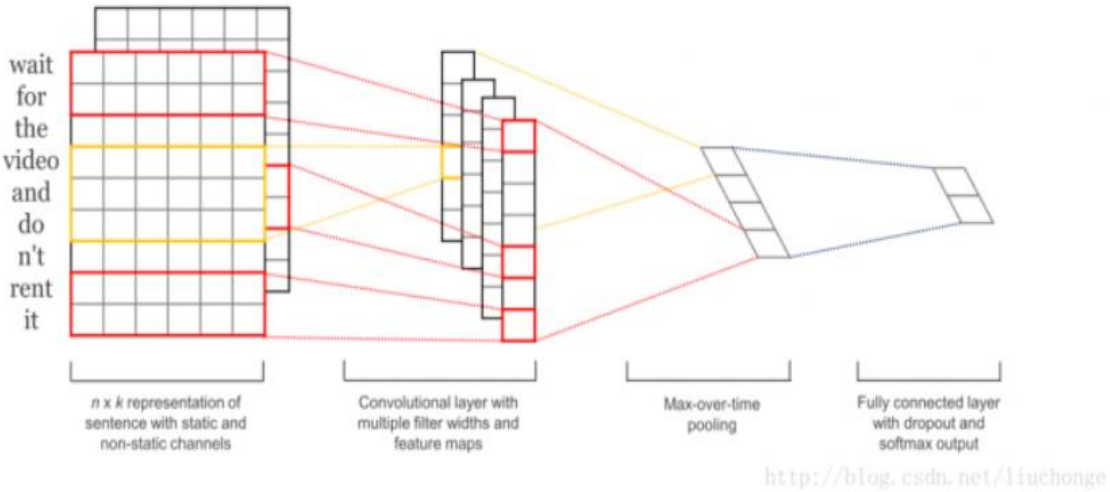


Figure: Structure of CNN for text classification[9]

3 Technical Details

3.1 Experiment Environment

3.1.1 Hardware Environment

For the hardware, I run those algorithms on my PC. The Figure below is my PC's configuration.

CPU: Intel 6th Core i7-6700HQ
GPU: Nvidia GeForce GTX 1060 6GB
RAM: 16GB

Figure: Working Environment

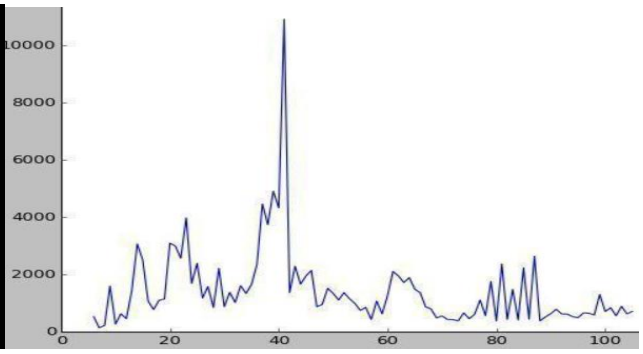
3.1.2 Software Environment

I mainly implemented the algorithms in Python 3.6.5, Scikit-Learn 0.19.1 and Keras 2.2.2 with Tensorflow as backend.

3.2 Dataset Preparation

The figures below shows the label distribution and length distributions of the raw dataset, I cut off those data points whose labels are 14,15,8,11. Then, I sampled 2000 data points from each categories for preventing label imbalance problems. Furthermore, I cut out the first 100 words to have a fixed-length training and testing dataset.

```
all:1411996
labeled:511218
label 1:138576
label 2:27489
label 3:199871
label 4:23409
label 5:44537
label 6:2179
label 7:13012
label 9:3294
label 10:1924
label 12:50138
label 13:5881
label 14:669
label 15:239
```



3.3 Feature Extraction

3.3.1 TF-IDF

Below is the code about How I implement TFIDF method to extract features.

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
count_v0= CountVectorizer();
counts_all = count_v0.fit_transform(all_text);
count_v1= CountVectorizer(vocabulary=count_v0.vocabulary_);
counts_train = count_v1.fit_transform(train_texts);
print ("the shape of train is "+repr(counts_train.shape))
count_v2 = CountVectorizer(vocabulary=count_v0.vocabulary_);
counts_test = count_v2.fit_transform(test_texts);
print ("the shape of test is "+repr(counts_test.shape))
tfidftransformer = TfidfTransformer();
train_data = tfidftransformer.fit(counts_train).transform(counts_train);
test_data = tfidftransformer.fit(counts_test).transform(counts_test);
```

3.3.2 Word2Vec

In this project, I used a pretrained Word2Vec embedding model, which is trained by full amount of text data in Sogou Corpus Database. This implies that model may provide extra data when we utilize this Word2Vec model in our further research.

3.4 Models

3.4.1 Linear Models

3.4.1.1 Linear SVM

In this project, I used Scikit-learn's SVM Classifier with default parameters (Penalty parameter C of the error term equals 1.0).

```
from sklearn.svm import SVC
svclf = SVC(kernel = 'linear')
```

3.4.1.2 Naïve Bayes

In this project, I used multinomial naïve bayes classifier, which is built-in within Sckit-learn's package.

3.4.2 Nonlinear Models

3.4.2.1 Convolutional Neural Networks

I used cross entropy as my loss function, and RMSprop as the optimizer, accuracy as the performance metric.

```
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['acc'])
```

Layer (type)	Output Shape	Param #
embedding_4 (Embedding)	(None, 100, 200)	13121000
dropout_4 (Dropout)	(None, 100, 200)	0
conv1d_4 (Conv1D)	(None, 98, 250)	150250
max_pooling1d_4 (MaxPooling1	(None, 32, 250)	0
flatten_4 (Flatten)	(None, 8000)	0
dense_7 (Dense)	(None, 200)	1600200
dense_8 (Dense)	(None, 12)	2412

Figure: model structure

3.4.2.2 LSTM

I used cross entropy as my loss function, and RMSprop as the optimizer, accuracy as the performance metric. The model structure is shown as below figure.

```
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['acc'])
```

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 100, 200)	13121000
lstm_2 (LSTM)	(None, 200)	320800
dropout_2 (Dropout)	(None, 200)	0
dense_2 (Dense)	(None, 12)	2412

Figure: model structure

4 Experimental Results

I calculated three times re-run of each models to measure their performance and time efficiency. The result is in the Figure below.

METHOD	ACCURACY	TRAINING TIME
CNN + word2vec	0.853	43 s/epoch* 2 epoch
CNN	0.815	66 s/epoch* 2 epoch
LSTM + word2vec	0.827	124 s/epoch* 2 epoch
LSTM	0.706	139 s/epoch* 2 epoch
Naive Bayes	0.814	< 1s
SVM	0.804	40 s

Figure: experimental results

5 Comparisons and Conclusions

For this project, I mainly test different machine learning algorithms on the Chinese news text dataset for text classification task. The research is not very comprehensive but enough to draw some conclusions.

Firstly, introducing pretrained word embedding models such as word2vec can both improve accuracy and reduce training time. Especially for the nonlinear models such as CNN and LSTM, it can help reduce parameters need to train and augment dataset indirectly, which is beneficial for the when you do not have enough labeled dataset for training.

Besides, for the text classification task, there is no absolute good or bad between linear and nonlinear methods. Linear methods have relatively quick speed and higher interpretability , while nonlinear methods have higher performance metrics and longer running time. Based on my research, CNN, LSTM, SVM and Naïve Bayes are all fine choices for the task considering of their accuracy metric against an 11-classification problems. The choice of which method to use depends on the size of the data set and the complexity of the problem itself.

6 Further Work

Firstly, we can also try different word embeddings such as Glove and Fasttext. Besides, we can try to do classification based on lower-level granularity. In this project, I only implemented word-level text classification research, which means words are smallest unit in the whole context. However, some recent research shows that char-level text classification can slightly improve the performance. Finally, there is still other algorithms that is suitable for text classification. For linear algorithm, I have not tried logistic regression. And for nonlinear algorithm, tree-based and RNN-based methods also worth a shot.

Appendix

a. Contributions of team members

I am the only one in my team, so I am the only contributor

b. References

[1] COLLOBERT, R., AND WESTON, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning (2008), ACM, pp. 160–167

[2] CAVNAR, W. B., TRENKLE, J. M., ET AL. N-gram-based text categorization. Ann Arbor MI 48113, 2 (1994), 161–175.

[3] SPARCK JONES, K. A statistical interpretation of term specificity and its application in retrieval. Journal of documentation 28, 1 (1972), 11–21.

[4] COLLOBERT, R., WESTON, J., BOTTOU, L., KARLEN, M., KAVUKCUOGLU, K., AND KUKSA, P. Natural language processing (almost) from scratch. Journal of Machine Learning Research 12, Aug (2011), 2493–2537.

[5] CONNEAU, A., SCHWENK, H., BARRAULT, L., AND LECUN, Y. Very deep convolutional networks for natural language processing. arXiv preprint arXiv:1606.01781 (2016).

[6] TF-IDF https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[7] Word2Vec https://en.wikipedia.org/wiki/Word2vec

[8] Rennie, J.; Shih, L.; Teevan, J.; Karger, D. (2003). Tackling the poor assumptions of Naive Bayes classifiers

[9] KIM, Y. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014).