Machine learning Nanodegree Captone Udacity Jiahao Cui

20 newsgroups Documents classification Report

Ⅰ.Definition

Project Overview

Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages and, in particular, concerned with programming computers to fruitfully process large natural language corpora. Documents classification is a one of the main application of NLP. The task is to assign a document to one or more classes based on the words,sentences and semantics it contains. The most frequently-used model to represent documents is the model called "Bags-of-words".In this model, a text is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.However,in recent years,with the help of powerful deep learning platforms, people have got decent outputs of word embedding,where words or phrases from the vocabulary are mapped to high dimensional vectors of real numbers.This gives people more choices to represent documents for classification tasks.

In this project,the 20 newsgroups dataset from CMU Text Learning Group Data Archives was given to classify,which has a collection of more than 18000 messages from 20 different newsgroups.I tried both the "Bags-of-words" model and word embedding model for this problem and various machine learning classification algorithms such as Naive Bayesian,Support Vector Machine and neural network were employed.

Problem Statement

The 20 newsgroups documents classification is a typical supervised multi-class classification problem.There are 20 classes and each document exactly belongs to one category.The goal is to extract representative features and build a model assigning the correct category for each messages.

I will explore two models to represent these documents for this task.The first model is *Word2Vec*.With the help of gensim python package,I can easily train on the *text8* corpus and get the vector representation of each word in a given dimension(100 typically) Then,I will try to convert each documents to vectors based on Word2Vec results(Discussed later). Another model is the "bags of words",where each document will be converted to a sparse TF-IDF vector.After getting the vectors representation of each messages,I can fit them into machine learning algorithms (including ensemble methods) to train the model and get predictions for testing data. Finally,I will optimize parameters and combinations of those algorithms in order to achieve the best performance.

Metrics

There are two main metrics for evaluating a multi-class classification model.

accuracy: the ratio of correct labels we made to the total number of predictions,ranging from 0 to 100%

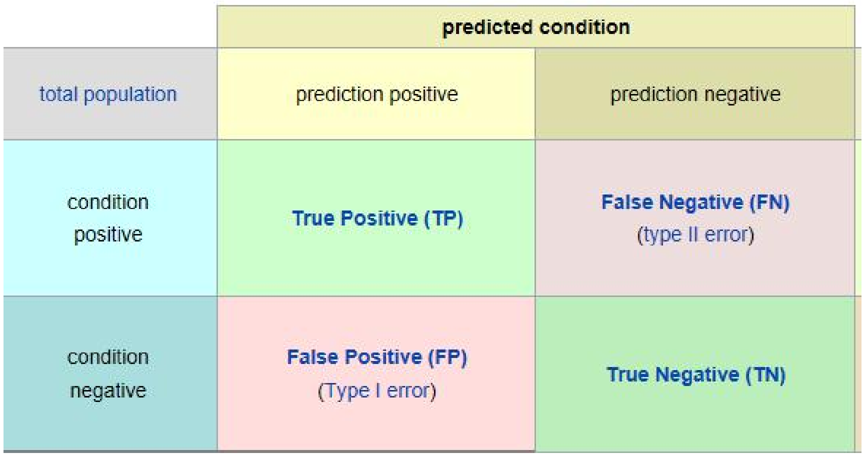
$$accuracy = \frac{\sum_{i=1}^{|V|} I(yi == y\hat{i})}{|V|}$$

V is the testing set, yi is the prediction of the model made and $y\hat{i}$ is the true label

time: the time algorithms consumed to train and classify.An efficient algorithm is expected to train on the training set and classifiy on the testing set within a few seconds in the corpus.

confusion matrix: specific layout that allows visualization of the performance of algorithms.

Here is an example of binary classification confusion matrix.



	predicted condition	
total population	prediction positive	prediction negative
condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)
condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**

Ⅱ Analysis

Data exploration

The dataset can be acquired online easily by the sklearn tools package.The dataset has been split into training set and testing set previously.The training set contains 11314 samples and the testing set contains 7532 ones.

Most messages contain 100-500 words,here is an example of a sample:

From: guykuo@carson.u.washington.edu (Guy Kuo)

Subject: SI Clock Poll - Final Call

Summary: Final call for SI clock reports

Keywords: SI,acceleration,clock,upgrade

Article-I.D.: shelley.1qvfo9INNc3s

Organization: University of Washington

Lines: 11

NNTP-Posting-Host: carson.u.washington.edu

A fair number of brave souls who upgraded their SI clock oscillator have *shared their experiences for this poll. Please send a brief message detailing your experiences with the procedure. Top speed attained, CPU rated speed, add on cards and adapters, heat sinks, hour of usage per day, floppy disk functionality with 800 and 1.4 m floppies are especially requested.*

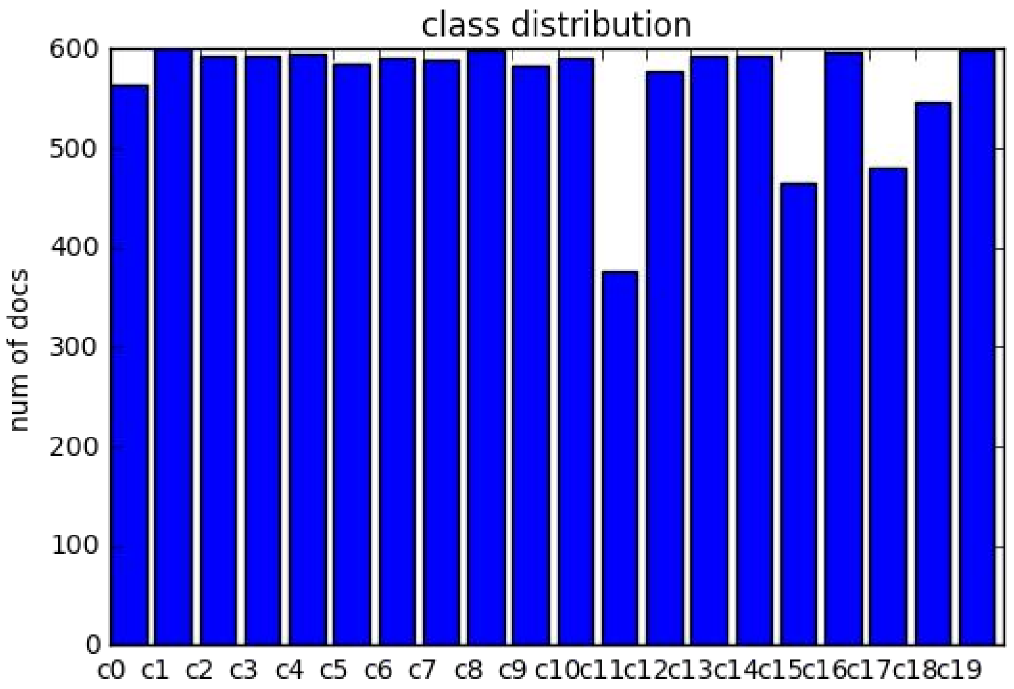
I will be summarizing in the next two days, so please add to the network knowledge base if you have done the clock upgrade and haven't answered this poll. Thanks.

Guy Kuo <guykuo@u.washington.edu>

It looks like an email with subjects,organizations and email address at the end. The contents may contain some implying terms for classification such as *'CPU', 'disk'* and *'network'.*But there are also much noisy information in this message like many meaningless punctuation and quotes.

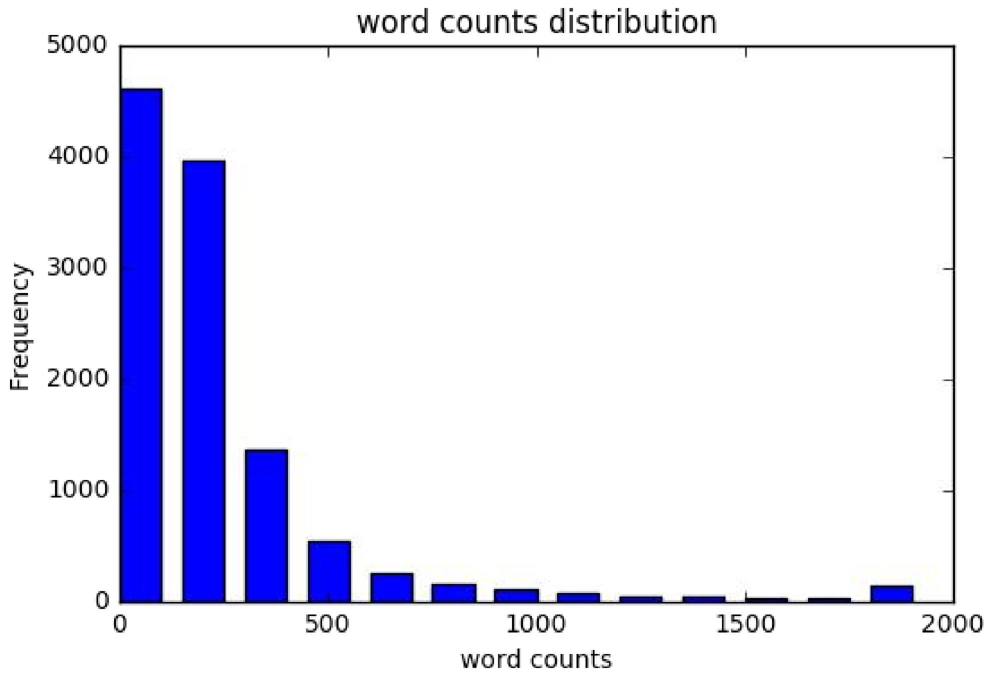
Exploratory Visualization

Let's first explore the class distribution of the corpus in training set.



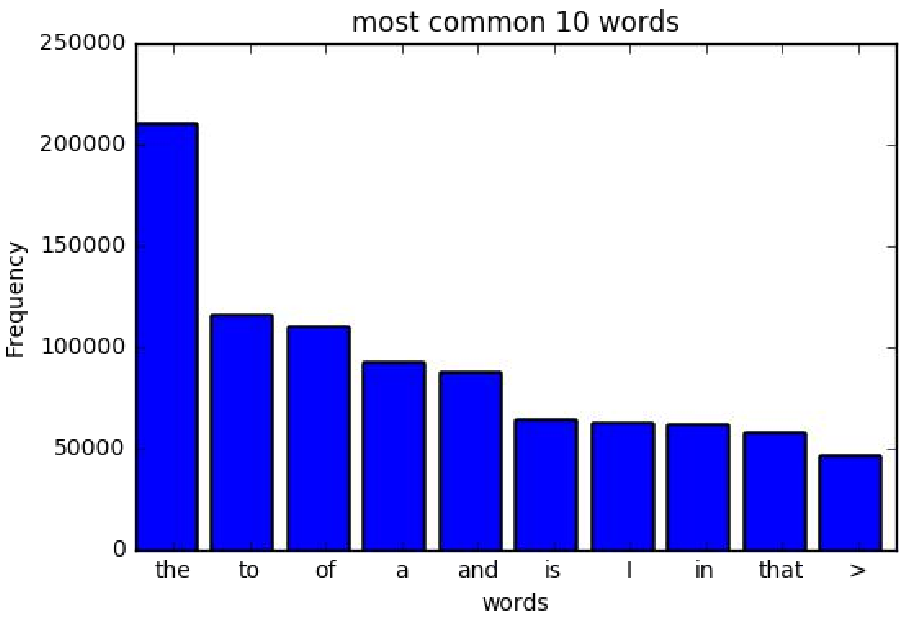
The number of samples in each class distributes relatively balanced.We do not need to apply any class weights methods.

Then,Here is the word counts distribution below



It can be seen that most samples have less than 500 words,but there are still some very long messages with more than 2000 words. Since more words can provide more information for us to classify,we do not need to remove those long messages as outliers.

There are 386410 words in the entire corpus in total,but most of them are special nouns and useless stop-words.Here is the show of most common ten words.



Most of those terms that occurred too many times in the corpus provide no information for classification.So these words will be removed as stop-words in natural language processing tasks.The sklearn and nltk packages provide English stop-words list for us and they will be removed automatically when computing TF-IDF vectors.

Algorithms and Techniques

I explored two models to vectorize texts,Word2Vec and TF-IDF, both of them have pros and cons.

Word2Vec model

Pros: Word2Vec can be trained by deep neural network on large corpus.So the semantic of terms can be reflected by the vectors.That means words closed semantically will also be closed in vector space.This property may enable the classifier algorithm to learn topics from vectorized texts more accurately. The dimension of vectors we get is much lower than that of “Bags of words” model,which will save time for classification algorithm.

Cons: Training and clustering word vectors is really time-consuming. Moreover, the performance of clustering in word vectors is bad because of too much noise and the number of clusters is hard to determine.In consequence,the doc-vectors may not reflect the topic of messages accurately.

TF-IDF model

TF-IDF is a widely-used model for representing documents.The tf-idf value increases proportionally to the frequency of a term in the document but is discouraged by times it appeared in other documents.

Pros: TF-IDF vectors can be extracted efficiently by sklearn tools.Previous work proved that TF-IDF model can reflect documents features effectively and achieve decent performance in text classification tasks.

Cons: The number of features for TF-IDF model equals to the number of words in corpus,so the feature matrix of a document is a sparse matrix with extremely huge dimension.Too many features can easily cause over-fitting.

Now Let’s discuss classification algorithms I employed.

Multinomial Naive Bayes

Pros: computationally efficient,easy to implement and understand,very suitable for text retrieval.

Cons: simply assume all features are independent each other,which may not be true.

Support Vector Machine

Pros: strong classification ability, high accuracy, effective for most classification tasks.

Cons: time-consuming to train and predict, easy to overfit.

Stochastic Gradient Descent

Pros: consuming less RAM, suitable for large scale datasets.

Cons: may fall into local minimum,may slow to converge.

Multi-layer neural network

Pros: most powerful classifier, highest accuracy, can be used for almost all classification tasks.

Cons: very computationally intensive,very easy to overfit

Rigde Classifier

Pros: applicable to both classification and regression tasks, can use penalty to prevent overfitting

Cons: sensitive to outliers.may be unstable for large scale data.

All of these methods mentioned above can be implemented by *sklearn* tools package.The TF-IDF vectors can also be extracted by functions built in *sklearn* package.The Word2Vec model can be trained by *gensim package.*

Chi-squared test

In statistics,the chi-squared test measures the dependence of two stochastic variable.High chi-squared stats between feature X and label y means X is predictive to y. The chi2 function build in sklearn tools can help me remove those irrelevant features with low chi-squared stats.

Benchmark

A group in Stanford university reached micro accuracy 0.85 on 20 categories using their Stanford classifier jar. After a series of optimization and improvements,the Stanford group reached the highest accuracy 0.888 on all 20 categories.Therefore,I set the accuracy threshold at 0.85 on testing set.

Ⅲ.Methodology

Data Processing

When using Word2Vec model, I wrote a function to remove punctuation,stop-words and turn texts to lower case.Then I counted the frequency of each terms appear in corpus and removed those appear less than 3 times,like ‘aaa’,perhaps skips.

When using TF-IDF model,the step mentioned above also needed to be done.Fortunately,the sklearn tools help me do the processing automatically when building TF-IDF vectors.

Implementation

Word2Vec model

I used word2vec model build in gensim package to train word embedding vectors on a large corpus called *text8*(the 20newsgroups corpus may not sufficient enough to train word2vec). I set the commonly-used vector dimension equals to 100, window size to 5 and employed the Skip-gram model to train the word vectors. After training the Word2Vec model,I employed the KMeans algorithm to cluster those vectors to a given number of clusters(500 for example). Then each documents can be represented as a vector of length 500,where each element in the vector implies how many words in the document belong to a certain cluster.

Finally,I fit the doc vectors to various of classification algorithms with default parameters settings to compute. The result turned out to be not very satisfying.Not matter how I change the cluster numbers and classification algorithms , the accuracy never reached 85%. I also tried to adjust the parameters of classification algorithms such the alpha of MultinomialNB ,but it gave me few improvements. With the number of clusters increased, the accuracy tended to increase too, but this strategy is extremely time-consuming.I reached highest accuracy at 60.7% with

500 clusters, much lower than the benchmark performance, So I have to mainly focus on the TF-IDF model below.

TF-IDF model

The TF-IDF vectors I got from sklearn tools is a sparse matrix but I can still fit them into classification algorithms directly. I explored many algorithms and the good performance ones were RidgeClassifier, SGDClassifier, LinearSVC, MLPClassifier as well as MultinomialNB. This is easy to understand because those models except Multinomial are all linear models that have a mechanism to prevent overfitting.Linear models handle every feature in the same way so they are more suitable for large amount of features. As I expected,the feed-forward neural network with one hidden layer reached the highest accuracy at 0.868 on the 7532 testing set,but it is certainly the most complex and most computationally intensive algorithm.Other classification algorithms all reached similar accuracy between 0.83-0.86.I set most parameters of those classification algorithms by default but introduced some parameters to inhibit overfitting.For example,I used *l2 penalty* for both SVC and SGDClassifier to restrict the magnitude of coefficients and set *alpha=1.25* to prevent overfitting for RidgeClassifier. I also tried to ensemble them using the voting strategy but the performance failed to improve obviously either. The best accuracy I got is 0.868 finally on all 20 categories.

Refinement

All the algorithms mentioned above reached nearly 100% accuracy on training set but none of them reached accuracy higher than 88% on testing set so overfitting happened on all algorithms.The main reason was that I extracted too many(129791) features from the TF-IDF model. I used *chi2 test* to do feature selection and found that remaining 60000 features can output best performance.After feature selection,my accuracy improved about 0.2% for those algorithms,not very obvious,but it can also save time for training.

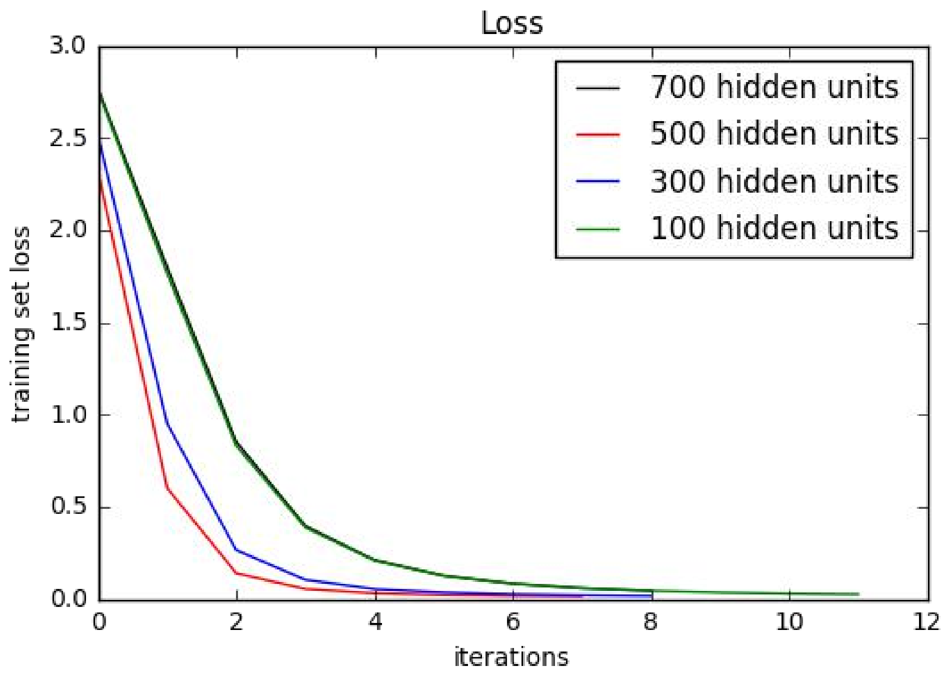
I also tried GridSearch method to select best parameters for all classifiers,but it seemed no apparent progress had been made.I think that also means the parameters I set before are suitable enough.

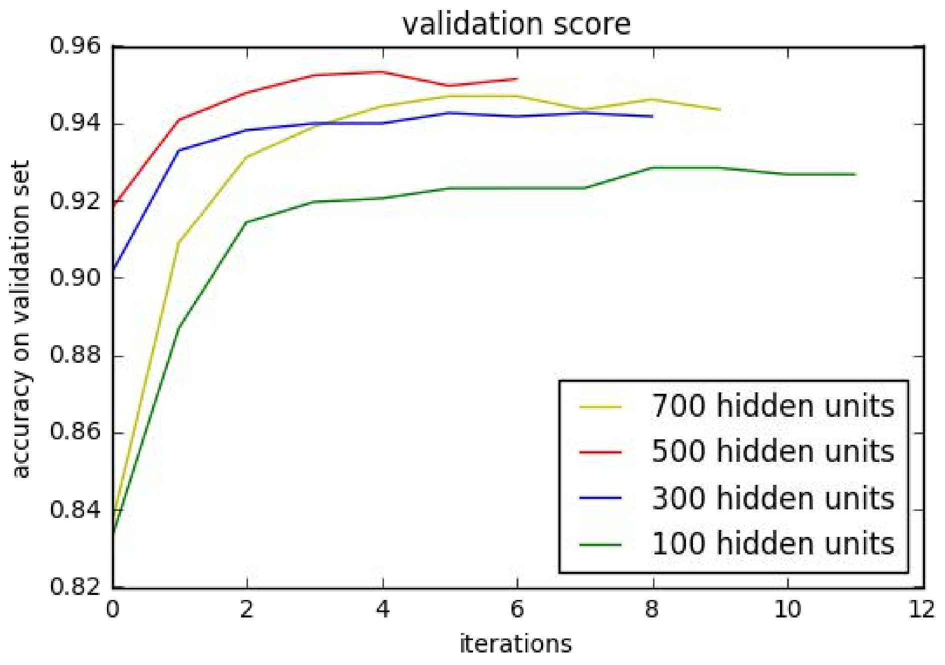
Training the neural network is too time-consuming, thus I applied a early stop strategy. When the accuracy on a validation set did not improve at all,the training step will stop early.It saved over 50% time for me.

Ⅳ.Results

Model Evaluation and Validation

Here is the visualization of how parameters influence the performance of neural network.





I split 10% off the training set for validation.When the accuracy on validation on validation set stops to improve,the training will stop early.

Neural network performance

Models/#hidden units	100	300	500	700
Iterations	11	8	6	9

Trainning time/seconds	232.0	638.6	954.8	927.4
Testing set accuracy	0.8647	0.8681	0.8659	0.8650

Justification

From the results shown above, I decided to employ the neural network with 300 hidden units as the final model because it achieved highest accuracy and much more efficient than neural networks with more layers and units.I think the Linear Support Vector Machine is also a good choice for this documents classification tasks,with accuracy closed to neural network and much shorter time to train.

Here is the performance of Support Vector Machine on different penalty parameter of *C*

C	*0.1*	*0.5*	*1.0*	*2.0*	*5.0*
Testing accruacy	*0.8497*	*0.8624*	*0.8611*	*0.8599*	*0.8587*

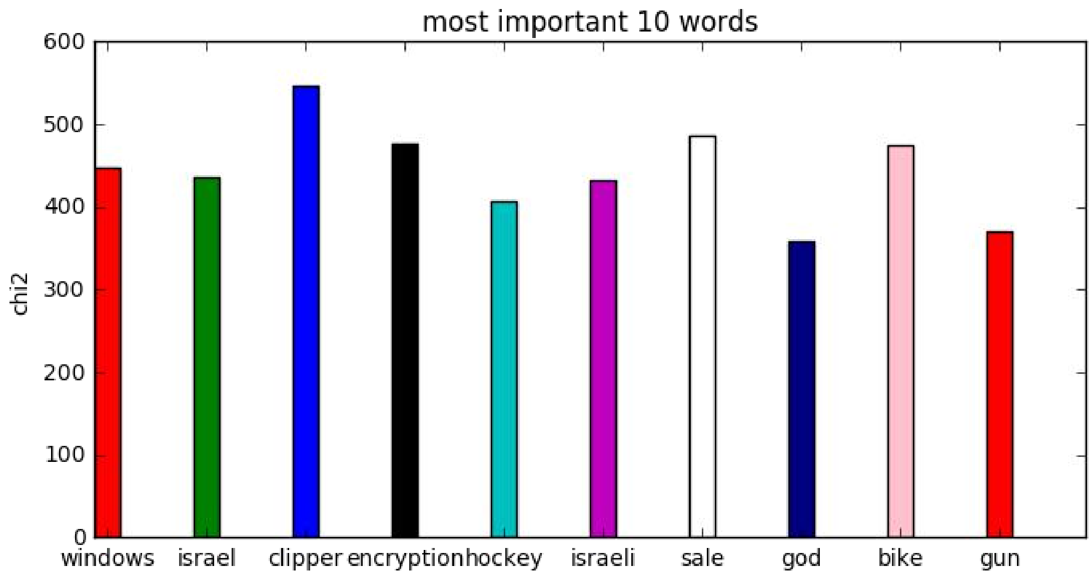
So C=0.5 was chosen as the best parameter for LinearSVC.

My final accuracy is higher than the benchmark threshold 0.85 I set before,but not great improvement.Since it is a multi-class classification task with 20 categories,the higher the accuracy we reached,the harder the performance to be improved.So I think my result is acceptable and applicable enough.

Ⅴ.Conclusion

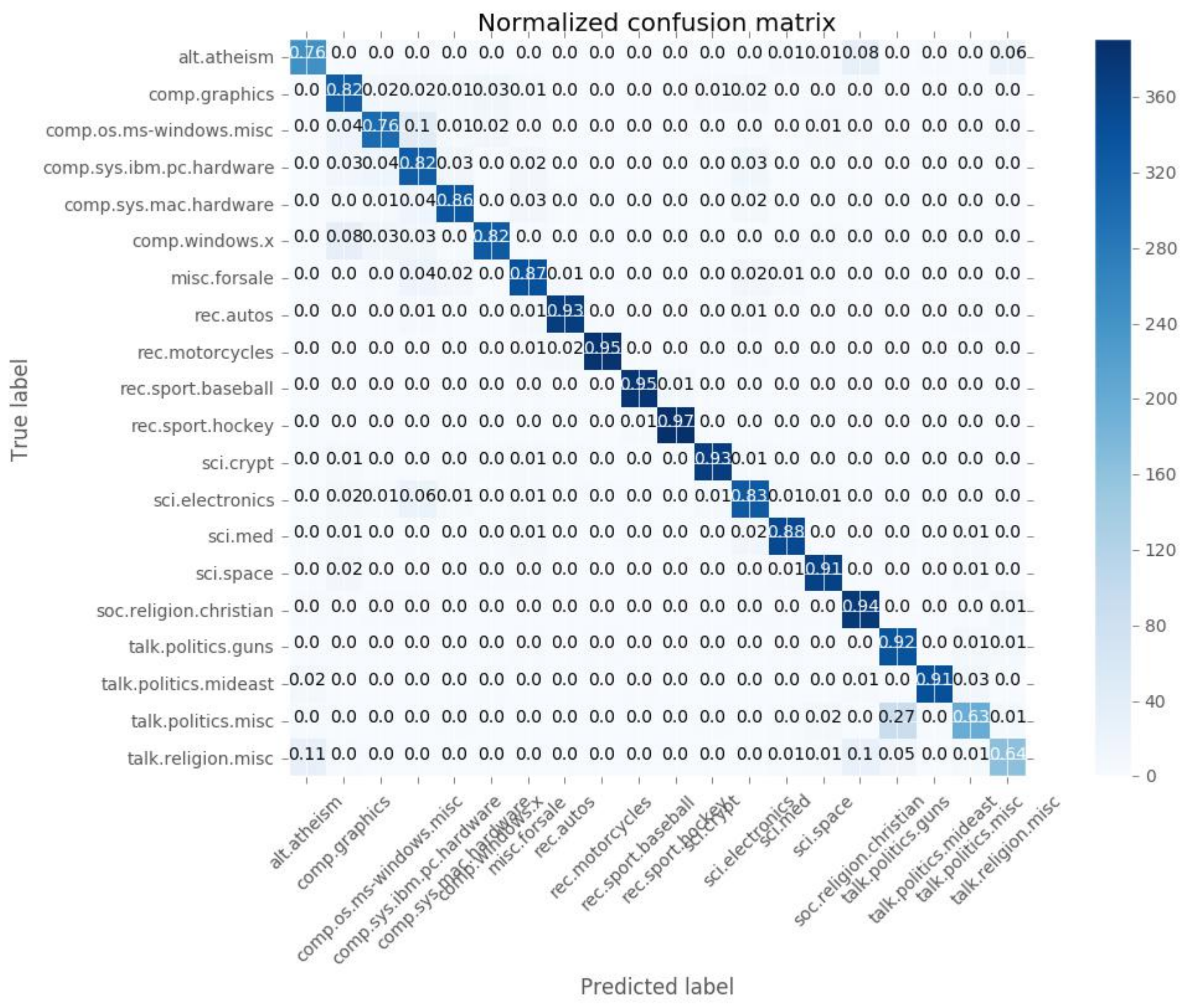
Result visualization

Here is the most important 10 terms for classification account for their *chi2 feature* value



We can see that all the terms shown in the charts are meaningful and expressive nouns.For example,the word ‘windows’ may imply computer technology topic and the word ‘gun’ may imply political and criminal topic.

Here is the confusion matrix of the final classification result.



We can see that the *talk.politics.misc and talk religion.misc* are more prone to be misclassified. I think this is because documents in these two classes have many similar terms to other topics such as *soc.religion.christian.*

Reflection

There are two main challenges I meet in this project,representing documents as matrix and fit them into classification algorithms. I tried the word embedding model to represent documents according to words semantic but the performance failed to defeat the traditional TF-IDF model. So I mainly focused on improve the accuracy and effectiveness of classification algorithms using TF-IDF model. After series of optimization and exploration,I finally reached accuracy approximate to 0.87,which is closed to the best performance of Stanford NLP Groups. I think the main barrier for accuracy to improve further is limitations of TF-IDF model itself instead of the classification algorithms.Some features cannot be extracted by TF-IDF model such as the similarity of words and the structure of paragraph.

Improvement

Transformation of nouns and verbs in English was not taken into account in this project.For example,' *computer'* and '*computers'* ,'*research'* and *researched'* have the same meaning,but they were considered as different features in TF-IDF model. I think the performance can be improved further if I reduce those redundant.

Reference

[1].Natural Language Processing: en.wikipedia.org/Natural Language Processing.

[2].Word2Vec: en.wikipedia.org/Word2vec

[3].scikit-learn package:www.scikit-learn.org

[4].gensim package:www.gensim.org

[5].NLP Stanford:nlp.stanford.edu/wiki/Software/classifier/20Newsgroup

[6].Classification of texts documents using sparse features: www.sciit-learn.org