Automatic Text Classification Algorithm

Implementation and Evaluation

Final Report

By Hao Wu

Abstract

The project implements automatic text classification using different machine learning algorithms, including K-Nearest Neighbor and Naïve Bayesian, and then evaluates their precision and performance.

Keywords

Text classification, machine learning

1. Introduction

Automatic text classification is a supervised technique that uses labeled training data to learn the classification system and then automatically classifies the remaining text using the learned system. It has become one of the most important techniques in text mining and wildly used in many areas, such as filtering Email spam, and categorizing newspaper articles, etc. To implement its algorithms, it will give us a deep understanding on how it works.

Automatic text classification tasks can be divided into two sorts: knowledge engineering and supervised machine learning. For knowledge engineering, all the rules are defined manually. On the other hand, supervised machine learning builds an automatic text classifier by learning the characteristics of the categories from training sets, which is more accurate than knowledge engineering. There are several machine learning techniques. This project will mainly focus on KNN and Naïve Bayesian.

2. Implementation and Project Architecture

2.1 Data cleaning

In order to classify the document, the first step is to do data cleaning, including stemming and lemmatization, and removing stopwords. I select the stem package of python Natural Language Toolkit (NLTK) and NodeBox::Linguistics to do stemming and lemmatization.

2.2 Building internal representations for documents

Un-normalized TF-IDF has been chosen to calculate the vector for each document.

2.3 Algorithm implementation

KNN (K=5, using cosine similarity as the method to calculate distance) and Naïve Bayesian (using log to avoid underflow problem) have been implemented.

2.4 Term selection

Employ information gain as dimensionality reduction (term selection) measurement.

2.5 Project architecture and main modules

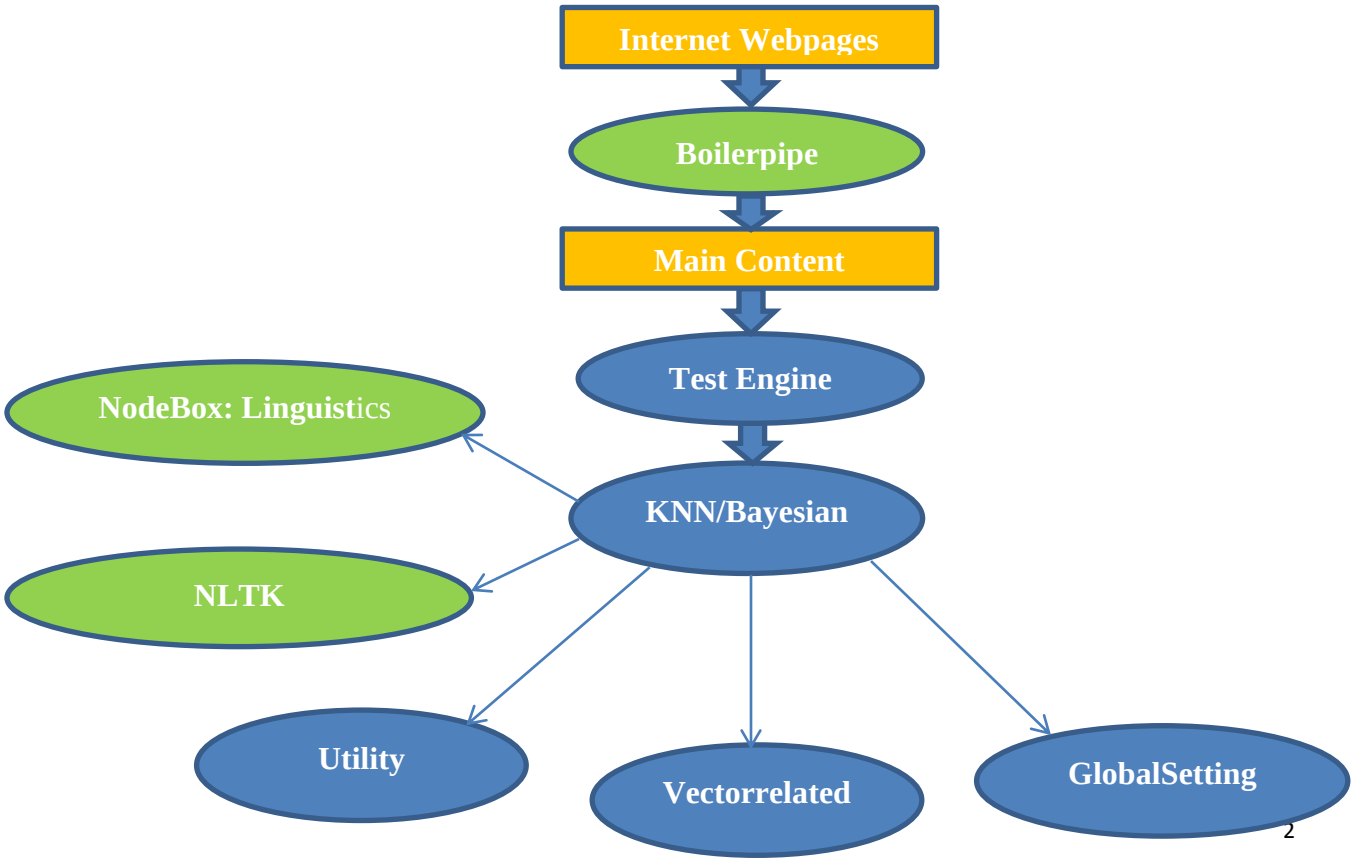


Figure 1: Project Architecture

Figure 1 shows the architecture of this project. Green ellipse means the third party module; yellow rectangle stands for the document which will be classified; blue ellipse stands for the modules implemented in this project.

Main Files (Modules) Description:

globalvariable.py	This file is used to store global environment settings, such as training set location, information gain ratio, a list of words which cannot be stemmed or lemmatized by NLTK and NodeBox library.
utility.py	This file contains some public functions used in KNN and Bayesian algorithm implementation.
vectorrelated.py	This file is a repository which contains all the parsing results of training set documents and test set documents, such as vector, vocabulary and document list, etc. It also has functions to calculate TF/IDF and Pearson distance.
knn.py	This file contains main implementation of KNN algorithm.
bayesian.py	This file contains main implementation of Bayesian algorithm.
mywget.py	This file is used to extract main content from webpage.
mytest.py	Main test engine module to evaluate accuracy of each algorithm
Getwebcontent.java	Java code is used to call boilerpipe
Test.log	Test set evaluation result
stoplist1,stoplist2	Stop list

Table 1: Main Modules

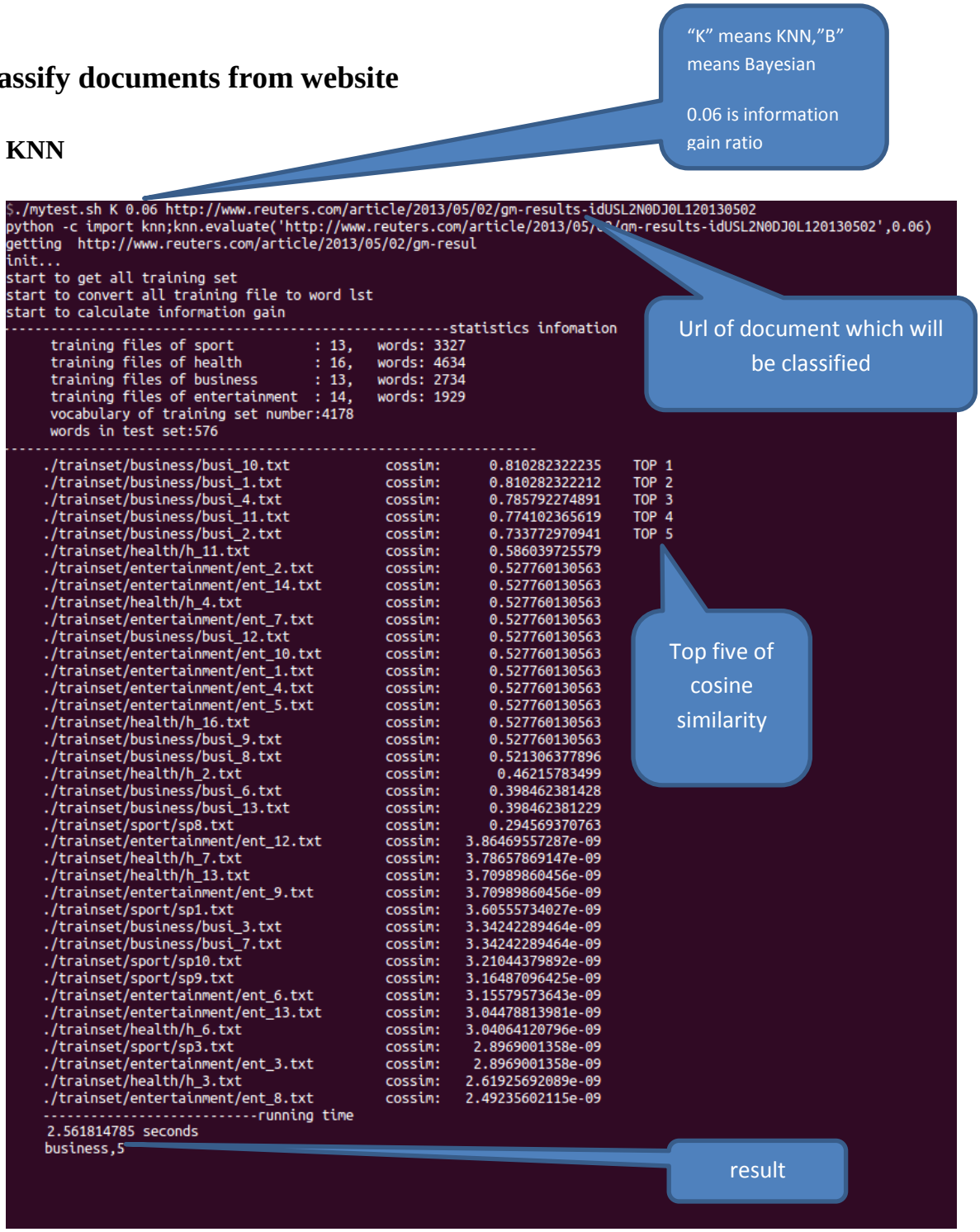
In order to improve the flexibility of the program, directory name has been used as the category. Each category training set will be stored in one directory, so it is easy to add other categories in future.

All codes have been uploaded to:

https://github.com/todatamining/textclassifiction.git

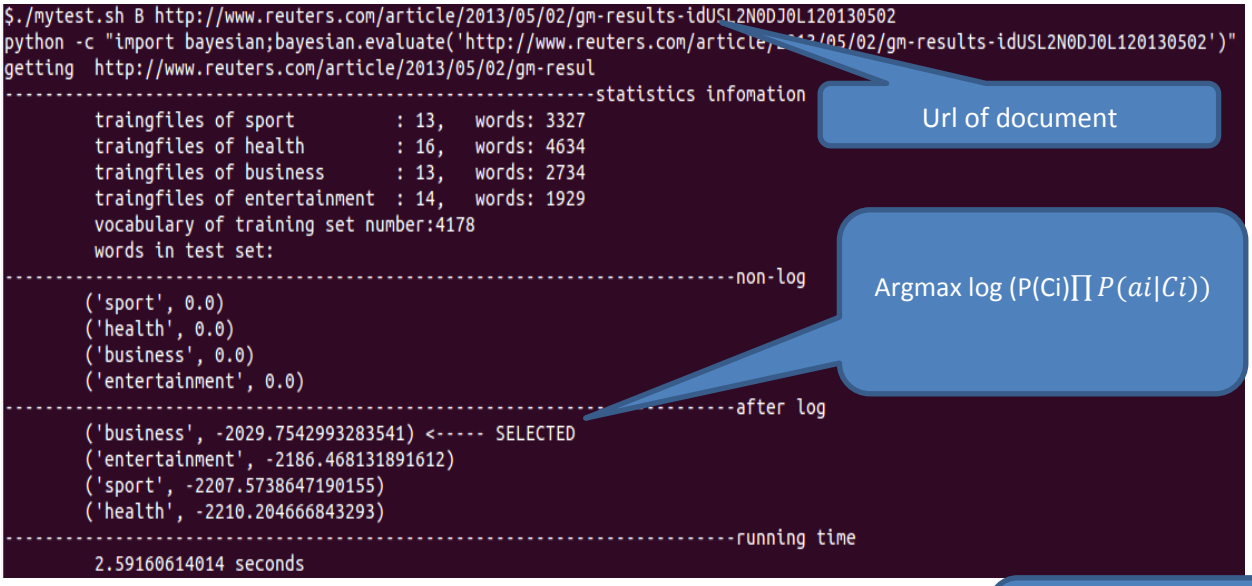
3. Classify documents from website

3.1 KNN

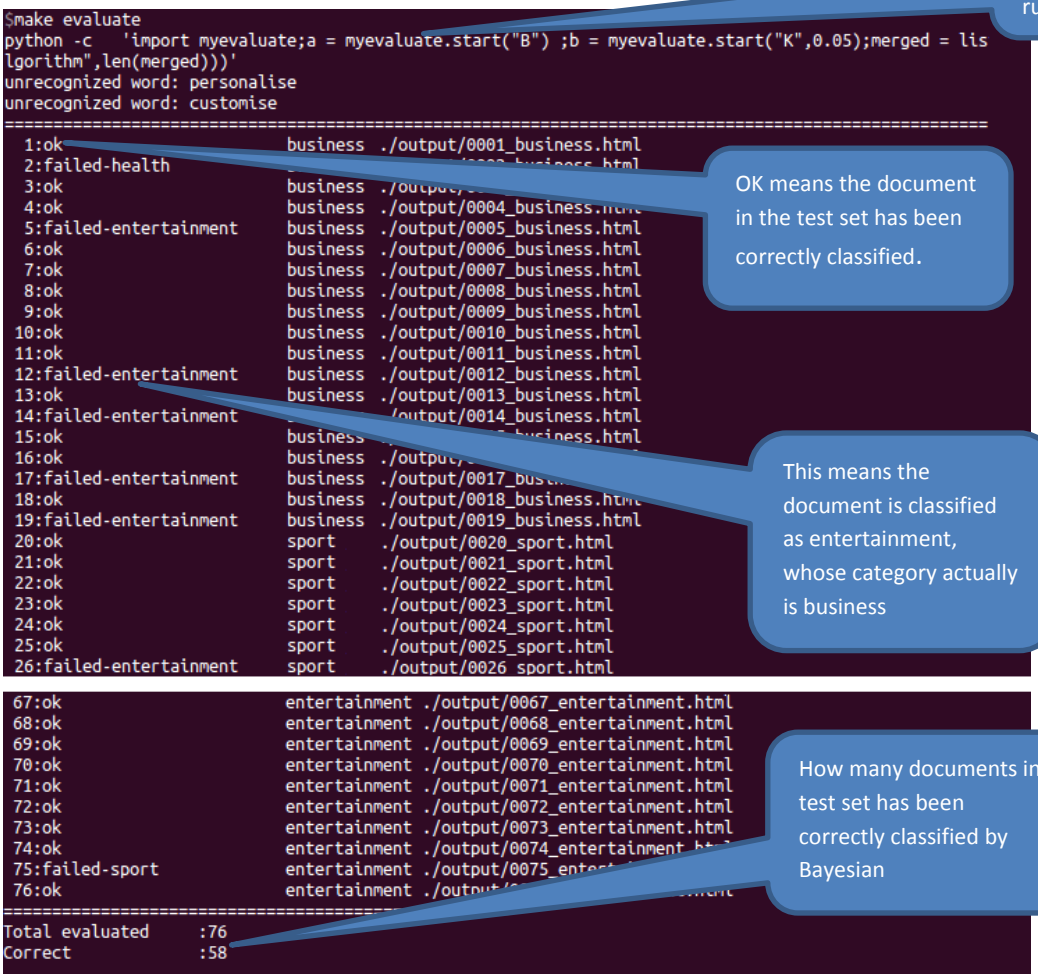



```
$./mytest.sh K 0.06 http://www.reuters.com/article/2013/05/02/gm-results-idUSL2N0DJ0L120130502
python -c import knn;knn.evaluate('http://www.reuters.com/article/2013/05/02/gm-results-idUSL2N0DJ0L120130502',0.06)
getting  http://www.reuters.com/article/2013/05/02/gm-resul
init...
start to get all training set
start to convert all training file to word lst
start to calculate information gain
------------------------------------------------------------statistics infomation
     training files of sport          : 13,   words: 3327
     training files of health         : 16,   words: 4634
     training files of business       : 13,   words: 2734
     training files of entertainment  : 14,   words: 1929
     vocabulary of training set number:4178
     words in test set:576
------------------------------------------------------------------
    ./trainset/business/busi_10.txt           cossim:      0.810282322235    TOP 1
    ./trainset/business/busi_1.txt            cossim:      0.810282322212    TOP 2
    ./trainset/business/busi_4.txt            cossim:      0.785792274891    TOP 3
    ./trainset/business/busi_11.txt           cossim:      0.774102365619    TOP 4
    ./trainset/business/busi_2.txt            cossim:      0.733772970941    TOP 5
    ./trainset/health/h_11.txt                cossim:      0.586039725579
    ./trainset/entertainment/ent_2.txt        cossim:      0.527760130563
    ./trainset/entertainment/ent_14.txt       cossim:      0.527760130563
    ./trainset/health/h_4.txt                 cossim:      0.527760130563
    ./trainset/entertainment/ent_7.txt        cossim:      0.527760130563
    ./trainset/business/busi_12.txt           cossim:      0.527760130563
    ./trainset/entertainment/ent_10.txt       cossim:      0.527760130563
    ./trainset/entertainment/ent_1.txt        cossim:      0.527760130563
    ./trainset/entertainment/ent_4.txt        cossim:      0.527760130563
    ./trainset/entertainment/ent_5.txt        cossim:      0.527760130563
    ./trainset/health/h_16.txt                cossim:      0.527760130563
    ./trainset/business/busi_9.txt            cossim:      0.527760130563
    ./trainset/business/busi_8.txt            cossim:      0.521306377896
    ./trainset/health/h_2.txt                 cossim:       0.46215783499
    ./trainset/business/busi_6.txt            cossim:      0.398462381428
    ./trainset/business/busi_13.txt           cossim:      0.398462381229
    ./trainset/sport/sp8.txt                  cossim:      0.294569370763
    ./trainset/entertainment/ent_12.txt       cossim:   3.86469557287e-09
    ./trainset/health/h_7.txt                 cossim:   3.78657869147e-09
    ./trainset/health/h_13.txt                cossim:   3.70989860456e-09
    ./trainset/entertainment/ent_9.txt        cossim:   3.70989860456e-09
    ./trainset/sport/sp1.txt                  cossim:   3.60555734027e-09
    ./trainset/business/busi_3.txt            cossim:   3.34242289464e-09
    ./trainset/business/busi_7.txt            cossim:   3.34242289464e-09
    ./trainset/sport/sp10.txt                 cossim:   3.21044379892e-09
    ./trainset/sport/sp9.txt                  cossim:   3.16487096425e-09
    ./trainset/entertainment/ent_6.txt        cossim:   3.15579573643e-09
    ./trainset/entertainment/ent_13.txt       cossim:   3.04478813981e-09
    ./trainset/health/h_6.txt                 cossim:   3.04064120796e-09
    ./trainset/sport/sp3.txt                  cossim:    2.8969001358e-09
    ./trainset/entertainment/ent_3.txt        cossim:    2.8969001358e-09
    ./trainset/health/h_3.txt                 cossim:   2.61925692089e-09
    ./trainset/entertainment/ent_8.txt        cossim:   2.49235602115e-09
    --------------------------running time
    2.561814785 seconds
    business,5
```

3.2 Na ïve Baysian



4. Performance Evaluation



4.1 Datasets and performance test

All the documents in the training set are from news.google.com. Table 2, Table 3 and Table4 show the summary of the training set and test sets:

Type	Documents Number	Total Word Count
Sport	13	3327
Health	16	4634
Business	13	2734
Entertainment	14	1929
The vocabulary number is 4178		

Table 2: Training Set Summary

1 business doc and 1 health doc from BBC.com
2 business docs and 1 sport doc from NYT.com
1 entertainment doc and 1 health doc from CNN.com
1 entertainment doc and 1 health doc from MSN.com

Table 3: Test Set 1

11 business news from news.google.com
18 sports news from news.google.com
19 health news from news.google.com
20 entertainment news from news.google.com
Total 68 documents

Table 4: Test Set 2

The average document processing time using Bayesian is 0.13s; while the average processing time using KNN (without term selection) is 43.4s.

4.2 Performance improvement

In order to improve the speed of processing document for KNN, the most importance measure is to employ term selection functions. Here we use information gain to reduce dimension. To make furthermore improvement, I have also employed python object as

cache to store mediate calculate result and using object serialization to cache information gain information which is already calculated.

After using term-selection, set information gain ratio to 0.05, 26 words has been selected: 'played' , 'tax' , 'month' , 'player' , 'march' , 'spending' , 'people' , 'start' , 'health' , 'team' , 'economy' , 'week' , 'star' , 'business' , 'season' , 'time' , 'wednesday' , 'game' , 'gain' , 'report' , 'february' , 'quarter' , 'study' , 'sale' , 'night' , 'retail'.

The average processing time of KNN now is 0.57 seconds. We can see that after term selection, the processing speed of KNN is highly improved.

Summary:

Algorithm	Processing time(second) for each document
Bayesian	0.13s
KNN(without term selection)	43.4s
KNN(with term selection , set information gain ratio to 0.05)	11s
KNN(with term selection, set information gain ratio to 0.05 and cache information gain result)	0.57s

Table 4: Average Processing Time for Each Algorithm

5. Precision Evaluation

Testing using the test set Table 3, Bayesian algorithm can classify test set correctly into corresponding category by one hundred percent, while KNN algorithm can correctly classify most of test set except one doc (1 business doc from BBC).

Test using the test set in Table 4, the precision for Bayesian is 0.853(58/68), and for KNN we get the following table and charts:

Information Gain Ratio	KNN Precision	KNN & Bayesian Precision	Words Selected	Average Processing Time for Each Documents
0.008	0.8235	0.9412	405	6.79
0.009	0.7941	0.9412	303	5.86
0.01	0.7647	0.9265	254	4.32
0.02	0.8088	0.9412	131	2.27
0.03	0.7941	0.9412	72	1.29
0.04	0.8088	0.9265	40	0.78
0.05	0.8382	0.9559	26	0.56
0.06	0.7647	0.9559	21	0.49
0.07	0.7353	0.9559	18	0.44
0.08	0.8088	0.9559	15	0.39
0.09	0.7941	0.9265	12	0.36
0.1	0.5882	0.9118	7	0.28
0.2	0.2794	0.9559	1	0.19

Table 5: Evaluation Result of Test Set 2

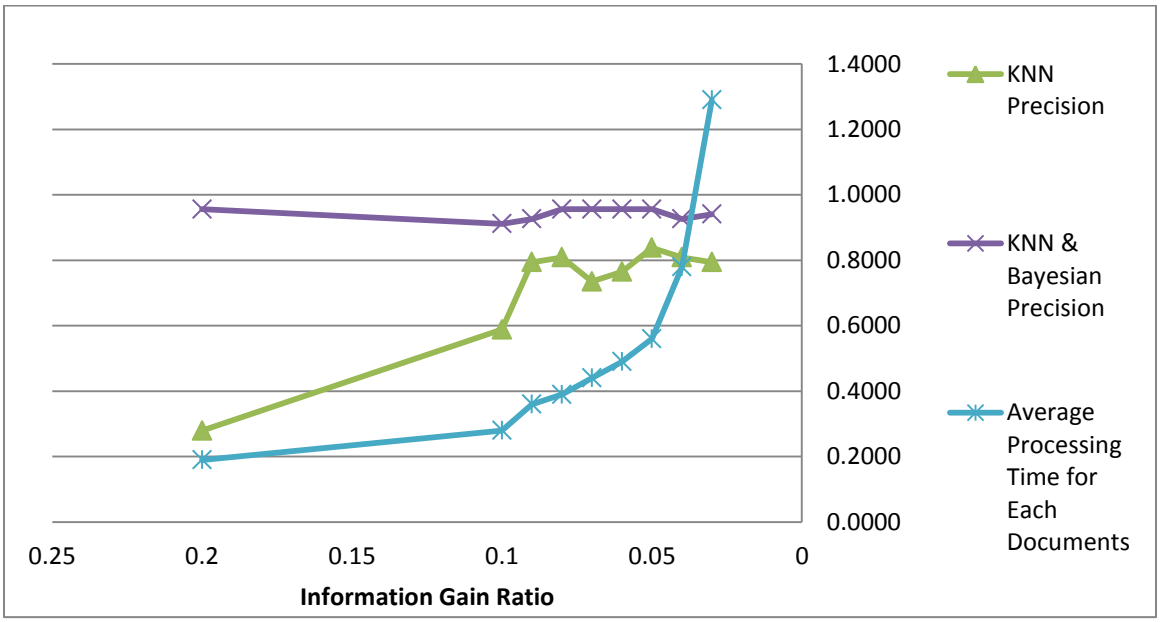


Figure 2: KNN Precision, KNN&Bayesian Precision and Average Document Processing Time VS Information Gain Ratio

Figure 2 shows the relations between the KNN precision, KNN&Bayesian precision and processing time of each documents with information gain ratio.

For the KNN&Bayesian precision, due to time limit I just simply merge their results. For example, suppose we have ten documents in our test set: doc1-doc10. The documents which correctly classified by KNN is doc1, doc2, doc3, the documents which correctly

classified by Bayesian is doc1, doc5, doc6. Then we say the result of KNN&Bayesian is doc1, doc2, doc3, doc5, doc6.

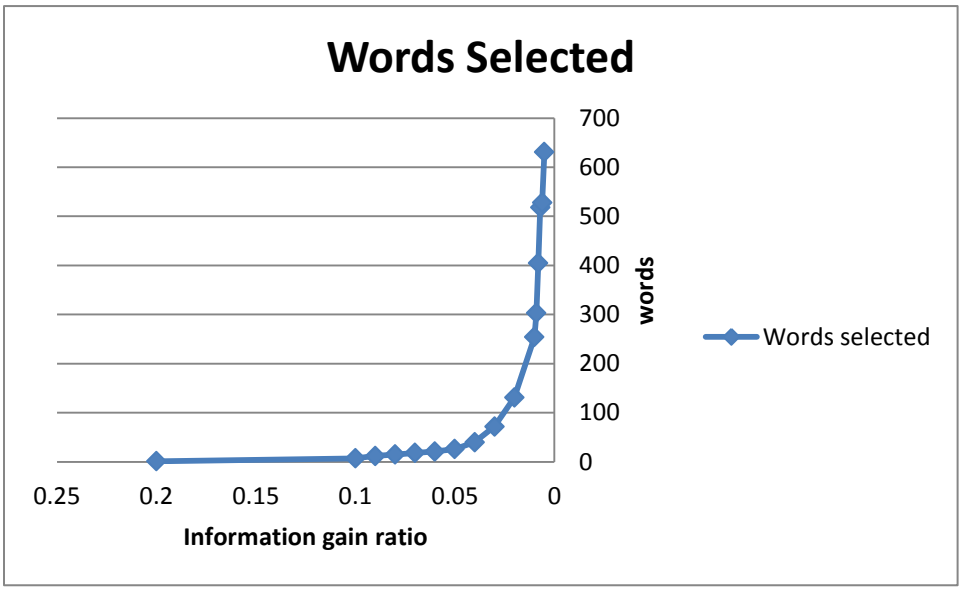


Figure 3: Words Selected VS Information Gain Ratio

Figure 3 shows the relations between words selected and information gain ratio.

We can draw the conclusion that generally the smaller the information gain ratio is, the better classification result we can get. On the other hand, it will need more time to process each document.

6. Problem Analysis

How to get the training set effectively

Currently all the documents in the training set are manually copied and pasted from news.google.com. If we want to add more documents into training set, it would be a tedious work.

However, if we simply use network tools or search engine to fetch the webpage, it will cause another problem. As shown in the figure 4 below, we can find out that there are "noise" contents in the webpage, such as menus, publicity, and banners, etc., which are highlighted by red rectangle. What we want is to analyze the main content of the page on

the left. Because the training set documents should be some random pages from random sites, there is no specific pattern to extract the contents from the website. Too many noise contents will greatly influence the classification result, which can be demonstrate from the table 6 below.

A business document is selected from WSJ.com and tested using KNN:

Copy and Paste from Webpage (Correct Result)	Get Whole Webpage (Wrong Result)
busi_2.txt cos: 0.160062008017 TOP 1	ent_4.txt cos: 0.0449222843799 TOP 1
busi_8.txt cos: 0.113130239133 TOP 2	ent_7.txt cos: 0.0448252988967 TOP 2
busi_11.txt cos: 0.109842991972 TOP 3	busi_6.txt cos: 0.0348505335107 TOP 3
busi_5.txt cos: 0.0971055944383 TOP 4	health_5.txt cos: 0.0313589380841 TOP 4
busi_6.txt cos: 0.0724766022027 TOP 5	ent_1.txt cos: 0.0289928223494 TOP 5
busi_4.txt cos: 0.0709718417705	health_15.txt cos: 0.0284979416636
ent_10.txt cos: 0.0679803047571	health_3.txt cos: 0.0264845651566
busi_7.txt cos: 0.0669870540235	busi_4.txt cos: 0.0225596785676
busi_13.txt cos: 0.0592794006664	health_10.txt cos: 0.0225101918615
busi_10.txt cos: 0.0529237486133	health_8.txt cos: 0.0221527916601

Table 6: Testing Results of Main Content VS Testing Results of Whole Webpage

We can see that because of the noise contents, KNN produces the wrong result. It is also tested using Bayesian, and the document is also recognized as entertainment.

Figure 4: Snapshot of Webpage from online.wsj.com

Solution (Not perfect):

In this project I use Boilerpipe library[2] to extract main content from webpage. But it seems that Boilerpipe still cannot handle all urls correctly.

7. Conclusions and Further Work

Conclusions. In this project, I have implemented KNN and Naïve Bayesian algorithm for text classification and use it to classify the document from internet, and then evaluate their performance and precision. From the evaluate result we can see that reducing the dimension using Information gain, the performance of KNN is highly improved. Using KNN together with Bayesian should get better result.

Further Work. 1) When evaluating the precision of this project, I found that Naïve Bayesian works well for business, health and entertainment news, while KNN works well for business, health and sports news. It need more research to find out the issue. 2) In order to make this project more practical, it would be better to integrate this project with other techniques, such as a search engine. 3) In this project I just directly merge the result of KNN and Bayesian for KNN&Bayesian. In fact we should give them different weight using least-squares linear regression.[1]

8. Reference

[1]A. Doan, P. Domingos, and A. Halevy. (2003). Learning to match the schemas of Databases: A Multistrategy Approach. Machine Learning.

[2]Christian Kohlschütter, Peter Fankhauser and Wolfgang Nejdl. (2010). Boilerplate Detection using Shallow Text Features WSDM 2010.

http://code.google.com/p/boilerpipe/

[3]http://answers.google.com/answers/main?cmd=threadview&id=225316

[4]Jythonhttp://www.jython.org/

[5]Package stem

Its Interfaces are used to remove morphological affixes from words, leaving only the word stem. Stemming algorithms aim to remove those affixes required for grammatical role, tense, derivational morphology leaving only the stem of the word. This is a difficult problem due to irregular words (eg. common verbs in English), complicated morphological rules, and part-of-speech and sense ambiguities (eg. ceil- is not the stem of ceiling).

http://nltk.googlecode.com/svn/trunk/doc/api/nltk.stem-module.html

[6]NodeBox:Linguistics

With the Nodebox English Linguistics library you can do grammar inflection and semantic operations on English content. You can use the library to conjugate verbs, pluralize nouns, write out numbers, find dictionary descriptions and synonyms for words, summarise texts and parse grammatical structure from sentences.

The library bundles WordNet (using Oliver Steele's PyWordNet), NLTK, Damian Conway's pluralisation rules, Bermi Ferrer's singularization rules, Jason Wiener's Brill tagger, several algorithms adopted from Michael Granger's Ruby Linguistics module, John Wiseman's implementation of the Regressive Imagery Dictionary, Charles K. Ogden's list of basic English words, and Peter Norvig's spelling corrector.

http://nodebox.net/code/index.php/Linguistics

[7]Noise content URL

http://online.wsj.com/article/SB10001424127887324323904578370170512486896.html?mod=WSJ_Business_LatestHeadlines

[8]Stop list:

http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

http://www.lextek.com/manuals/onix/stopwords1.html