

Document Classification

IKT 407

Web-mining and data analysis

November, 2005



HØGSKOLEN I AGDER

Agder University College

Faculty of Engineering and Science

<p>Author(s):</p> <p>Kun Yang</p> <p>Fei Yao</p>	<p>Supervisor(s):</p> <p>Morten Goodwin Olsen</p>
<p>Version: 1.0</p> <p>Status: FINAL</p>	<p>Pages: 39 (including this page)</p> <p>Modified date: 2005-11-21</p>
<p>Keywords:</p> <p>Document classification Naïve-Bayes Algorithm Dyslexic Python</p>	
<p>Abstract:</p> <p>This report describes the development of a web page classifier which could identify whether a web page is readable for the dyslexic.</p> <p>The problem of classifying a web page has been approached in the way which utilizes the use of the Naïve Bayes algorithm.</p> <p>The project has resulted in a functional prototype where the implementations bases itself on combining different metrics which may use discrete and continuous tests. The prototype was able to classify up to 90% correct.</p> <p>Python was used for the implementation.</p> <p>In this report, we first introduce some basic theory of web mining and some detail information of our project. Secondly, we describe how we design and implement our prototype in detail, then we evaluate the results and point out how to make further improvement. Finally, we list out the references we used and give the source codes in Appendix A.</p>	

Version Control

<i>Version</i>	<i>Status</i>	<i>Date</i>	<i>Change</i>	<i>Author</i>
1	Presentation 1	2005-09-27		Kun & Fei
2	Presentation 2	2005-10-26		Kun & Fei
3	Draft	2005-11-09		Kun & Fei
4	Review	2005-11-19		Kun & Fei
5	Final	2005-11-21	Final report	Kun & Fei

Abstract

Our report describes the development of a web page classifier which could identify whether a web page is readable for the dyslexic.

The problem of classifying a web page has been approached in the way which utilizes the use of the Naïve Bayes algorithm.

The project has resulted in a functional prototype where the implementations bases itself on combining different metrics which may use discrete and continuous tests. The prototype was able to classify up to 90% correct.

Python was used for the implementation.

In our report, we first introduce some basic theory of web mining and some detail information of our project. Secondly, we describe how we design and implement our prototype in detail, then we evaluate the results and point out how to make further improvement. Finally, we list out the references we used and give the source codes in Appendix A.

Table of content

Abstract	3
1 Introduction	6
1.1 Project background.....	6
1.2 Project description.....	7
1.3 Approach.....	7
1.4 Limitation.....	7
2 Theories	9
2.1 Web mining.....	9
2.2 Web content mining.....	10
2.3 Text classification.....	10
3 Requirements specification for Classifier	12
3.1 Naïve-Bayes Algorithm.....	12
3.1.1 <i>Algorithm explained</i>	12
3.1.2 <i>Mathematical notation</i>	13
3.2 Readable web pages for dyslexic.....	14
3.2.1 <i>dyslexia</i>	14
3.2.2 <i>the content of dyslexia</i>	14
3.2.3 <i>Metrics</i>	15
3.3 Programming language.....	16
4 Web Document Classification System	17
4.1 Data Pre-Processing.....	17
4.2 Feature Extraction.....	17
4.3 Learning and Classification.....	17
5 Design and implement	19
5.1 Design.....	19
5.1.1 <i>Naïve Byes algorithm in our project</i>	19
5.1.2 <i>main stages</i>	20
5.1.3 <i>Executive flow</i>	21
5.2 Implement.....	22
5.2.1 <i>Structure of program</i>	22
5.2.2 <i>Main program</i>	22
5.2.3 <i>Wordlist.py and sentencelist.py</i>	24
5.2.4 <i>tp.py and sp.py</i>	24
5.2.4.1 <i>tp.py</i>	24
5.2.4.2 <i>sp.py</i>	25
5.2.5 <i>tpm.py and spm.py</i>	25
5.2.6 <i>wt.py and st.py</i>	26
5.2.7 <i>gp.py and hp.py</i>	26
6 Results	27
7 Evaluation of result	28
8 Further developments	31
9 Conclusions	32

10 References	33
11 Appendix	34
11.1 wordlist.py and sentencelist.py.....	34
11.2 tp.py and sp.py.....	34
11.2.1 <i>tp.py</i>	34
11.2.2 <i>sp.py</i>	35
11.3 tpm.py and spm.py.....	35
11.4 wt.py and st.py.....	36
11.5 gp.py and hp.py	37
11.6 mainnew.py.....	38

1 Introduction

1.1 Project background

Nowadays, Internet has become more and more important and popular in the world. Accessibility of the content of Internet is mainly decided in the form of the content (html-code, colors, pictures, animations, etc.). One important domain for machine learning is document classification, in which each instance represents a document and the instance's class is the document's topic. [2] Documents might be news items and the classes might be domestic news, overseas news, financial news, and sport. Not only the contents of the documents are different, but also the forms of the texts are different. In fact, the form of the content of Internet can determine to what extent people with disabilities can utilize the information.

On the other side, dyslexia is a specific type of learning difficulty where a person of normal intelligence has persistent and significant problems with reading, writing, spelling and, sometimes, mathematics and musical notation. The person may not have difficulties in other areas: many dyslexic people are extremely creative, think laterally and have excellent problem-solving skills. It may be helpful to think of dyslexia as an information processing difficulty.

Earlier work has focused on performing some text classifications with success using Naïve Bayes, but has not focused on the classification of text based on readability. [7][8]

In order to help the people with dyslexia to read the content of web pages on the Internet, we have to analyze the web pages, and get some special characters of this kind of web pages, and at last test whether a web page is easy enough for them to read.

Assessing understanding extent of the Internet content is therefore important, and is the main purpose of our project.

1.2 Project description

Our project is to make a classifier to decide whether a document should be assigned to a particular category or not. In our case, there are two categories:

- Well designed for dyslexic readers
- Not well designed for dyslexic readers

It is a kind of text classification. It was therefore liable to assume that methods seen applied for text-classification also could be used for our specific purpose. So we would need to base our classification on textual content. At first we should get some proper metrics, and then we need to train the necessary probabilities according to the Naïve-Bayes Algorithm which is One of the more successful and known algorithms for learning to classify text and is rather easy to implement as a good basis for the prototype of our classifier.

1.3 Approach

The project consists of several tasks. The first step is to find out the metrics of the text that is suitable for the dyslexia. This step is so important that if we can not find the correct metrics, it would influence our result.

Using Naïve Bayes algorithms, we will then analyze these data to make the classifier.

1.4 Limitation

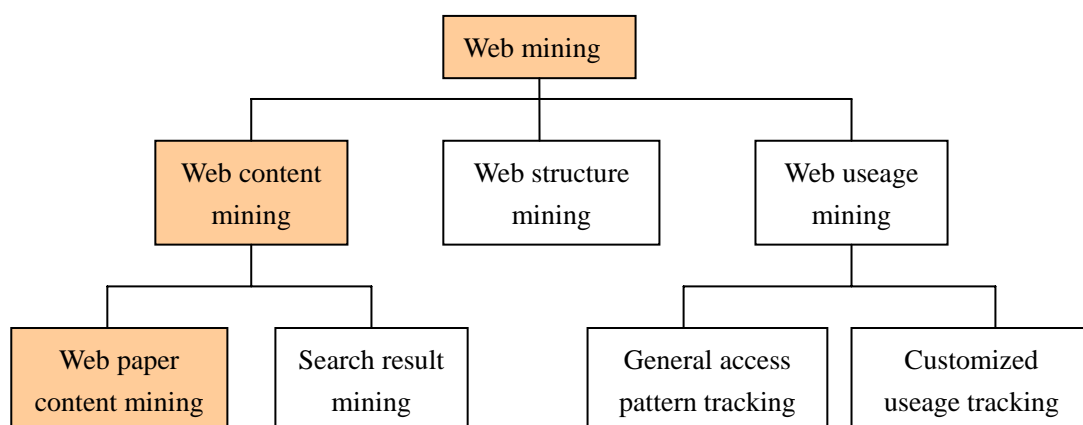
As we all know, web pages are not only in English, and of course dyslexic are at all parts of the world, most of them cannot read the web pages in English. But in our project, we only consider web pages written in English for reasons of simplicity and it is only a prototype

The classifier is designed based on the metrics of the texts' shape. We didn't refer to the meaning of the texts. So, it dose not consider readability of pages. A lot of dyslexics have the Web pages read to them from an application. However, this is not easy to test without such an application.

2 Theories

2.1 Web mining

Web mining is mining of data related to the World Wide Web. This may be the data actually present in Web pages or data related to Web activity. And the Web can be viewed as the largest database available and presents a challenging task for effective design and access. This field consists of using data mining techniques in order to develop approaches and tools that make it possible to extract relevant information about the Web (documents, traces of interactions, structure of pages and so on). Organizations and companies have invested millions of dollars in internet and intranet technologies. They need to be able to track and analyze user access patterns to further enhance and improve this technology. [8]



Web mining structure

Web mining tasks can be divided into several classes, web structure mining, web content mining and web usage mining. Web structure mining focuses on analyzing the physical link structure of websites. Web content mining focuses on analyzing the contents and resources on a website, such as images, text and code. Web usage mining focuses on the navigation and behavior of the website users and is the approach taken by this project. [8]

2.2 Web content mining

From the above picture, the red parts of it are related to our project theory. Web content mining can be thought of as extending the work performed by basic search engines. There are many different techniques that can be used to search the Internet.

Web Content Mining uses the ideas and principles of data mining and knowledge discovery to screen more specific data. The use of the Web as a provider of information is unfortunately more complex than working with static databases. Web content mining aims to discover, extract, integrate and analyze useful information from Web page contents.

2.3 Text classification

Texts are characterized by the words that appear in them, and one way to apply machine learning to text classification is to treat the presence or absence of each word as a Boolean attribute. And the categories are already known beforehand and given in advance for each training documents. The unsupervised version of the problem is called document clustering.

However, this does not take into account the number of occurrences of each word, which is potentially useful information when determining the category of a document. Instead, a document can be viewed as a bag of words — a set that contains all the words in the document, with multiple occurrences of a word appearing multiple times (technically, a set includes each of its members just once, whereas a bag can have repeated elements). [8]

There are three main approaches to solve this task:

- Naïve Bayes
- Support Vector Machine
- Nearest Neighbour

In our project, we use Naïve Bayes algorithm to solve our problem, because it is very fast and quite accurate.

3 Requirements specification for Classifier

In this chapter, we refer some specification for our project. First of all, we should know how to use the algorithm which is the basic knowledge of the classifier. Then, training is the most important part. Thus, we should know what kind of web pages we choose for training. The programming part cannot be overlooked which is to execute the whole project.

3.1 Naïve-Bayes Algorithm

3.1.1 Algorithm explained

We have mentioned that one of the classification algorithms is Naïve Bayes (NB) theorem from probability theory. And in this section we briefly introduce the key concepts underlying the Naïve Bayes algorithm.

NB algorithm has been widely used for document classification, and shown to produce very good performance. The basic idea is to use the joint probabilities of words and categories to estimate the probabilities of categories given a document. [7]

NB algorithm computes the posterior probability that the document belongs to different classes and assigns it to the class with the highest posterior probability. The posterior probability of class is computed using Bayes rule and the testing sample is assigned to the class with the highest posterior probability.

The naive part of NB algorithm is the assumption of word independence that the conditional probability of a word given a category is assumed to be independent from the conditional probabilities of other words given that category.

There are two versions of NB algorithm. One is the multi-variate Bernoulli event model that only takes into account the presence or absence of a particular term, so it doesn't

capture the number of occurrence of each word. The other model is the multinomial model that captures the word frequency information in documents.

There is one thing should be implemented despite its reasonably good results. Using the naive Bayes requires some training data, but its accuracy stalls somewhat if the training data gets too big. In other words, finding quite enough training data is important with Naive Bayes, but using too much training data gives little benefit.

3.1.2 Mathematical notation

- A. Given a set of data $X = \{x_1, \dots, x_n\}$;
- B. Suppose that either hypothesis h_1 or hypothesis h_2 must occur, but not both;
- C. Also suppose that x_i is an observable event.

Bayes Theorem is:

$$P(h_1 | x_i) = \frac{P(x_i | h_1)P(h_1)}{P(x_i | h_1)P(h_1) + P(x_i | h_2)P(h_2)} \quad (1)$$

$P(h_1 | x_i)$ is called the **posterior probability**;

$P(h_1)$ is the **prior probability** associated with hypothesis h_1 ;

$P(x_i)$ is the probability of the occurrence of data value x_i ;

$P(x_i | h_1)$ is the conditional probability.

Where there are m different hypotheses we have:

$$P(x_i) = \sum_{j=1}^m P(x_i | h_j)P(h_j) \quad (2)$$

Thus, we have:

$$P(h_1 | x_i) = \frac{P(x_i | h_1)P(h_1)}{P(x_i)}$$

Bayes rule allows us to assign probabilities of hypotheses given a data value, $P(h_j | x_i)$. Here we discuss tuples when in actuality each x_i may be an attribute

value or other data label. Each h_i may be an attribute value, set of attribute values (such as a range), or even a combination of attribute values. [8]

3.2 Readable web pages for dyslexic

3.2.1 Dyslexia

First of all, we should get some conception about what is dyslexia. Dyslexia is mainly perceived to be a problem with literacy skills, i.e. reading, writing and spelling; although it is now widely accepted that dyslexia can affect a number of areas including memory, organization and concentration. Dyslexia can occur at any level of intellectual ability. It is not the result of poor motivation, emotional disturbance, sensory impairment or lack of opportunities, but it may occur alongside any of these. [1]

3.2.2 The content of dyslexia

Then, what kind of web page is suitable to dyslexic? Basically speaking, the web page should be easy to read, that is to say the design and layout of the pages should be clear, and the text in it cannot be complicated.

In fact, we tried to connect with some people who are real dyslexia, but we failed to do that. So, we collected some information just from the prior researches. Having read some materials about it, we concluded several points shown to distinguish between the levels of readability in [1]:

- Keep sentences short
- Keep the words short
- Prefer active verbs
- Use 'you' and 'we'
- Choose words appropriate for the reader
- Don't be afraid to give instructions

- Use positive language
- Avoid Italic
- Font size
- Readable fonts

There are two examples about the styles of the same text, and we can feel it clearly.

The one which is easy to read for a person with dyslexia:

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The one which is hard to read for a person with dyslexia:

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

3.2.3 Metrics

In our project, we choose the length of word and the length of sentence as the metrics. These two metrics are most easy to implement and could be used to prove the concept of our prototype.

The two metrics are:

- The length of word

The length of word is very important factor for dyslexia. Generally speaking, the word which contains above 7 letters is more complicated than the one which contains just 1 – 6 letters. That is to say, the dyslexia usually uses the words which are shorter.

- The length of sentence

Line length can affect the ease and speed of your reading. Very long and very short lines force you to read more slowly. It is helpful to think of line length in terms of the

number of characters in the line (including spaces). A line of body text should normally contain 60 to 72 characters, or about 10 to 12 words. [1]

3.3 Programming language

We use python to program in our project. Python is an interpreted, interactive, object-oriented programming language. It is often compared to Tcl, Perl, Scheme or Java. [3]

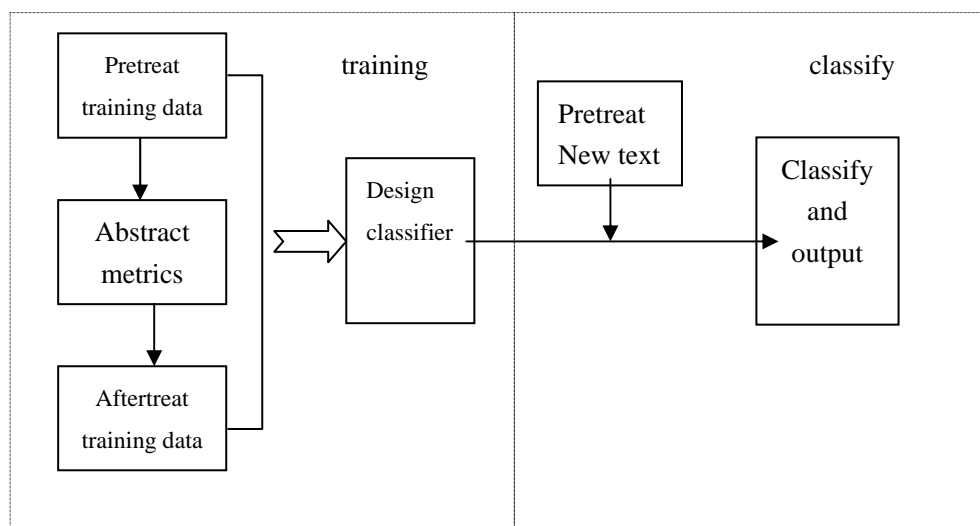
Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems (X11, Motif, Tk, Mac, MFC, wxWidgets). New built-in modules are easily written in C or C++. Python is also usable as an extension language for applications that need a programmable interface.

The Python implementation is portable: it runs on many brands of UNIX, on Windows, OS/2, Mac, Amiga, and many other platforms. If your favorite system isn't listed here, it may still be supported, if there's a C compiler for it.

4 Web Document Classification System

We now introduce the system and accompanying process that have been developed.

This is the picture of system structure



4.1 Data Pre-Processing

Data preparation is a key step in the data mining process. For us this step involved feature extraction and modeling and document classification. We describe how these were achieved next.

4.2 Feature Extraction

The process of feature extraction is that of determining which keywords will be useful for expressing each document for classification learning. Document modelling is the process of expressing the existence or non-existence, frequency and weight of each document feature based on a fixed feature word.

4.3 Learning and Classification

In order to do supervised learning and evaluate the accuracy of Web document classification based on Bayes we must provide classified documents as input.

Our system uses the naïve Bayesian learning method as it is a representative algorithm for supervised learning. The Naïve Bayesian classification learning method classifies each Web document with the highest probability class. Where the conditional probability of a given document is low or there is a conflict the system asks the user to choose the most appropriate classification. In situations where either the difference between the two or more highest conditional probabilities is small or the highest conditional probability is low (for example, the highest conditional probability is 0.2 ~ 0.3 and less) we ask the user to intervene. Since precision and trust are closely related, we don't want the system to give an incorrect classification, resulting in the users loss of faith in the system. Hence, when the system can not clearly assign a class, the system assigns the document to 'Others' for the user to deal with.

5 Design and implement

The design of classifier and implementation of the program will be covered in this section. We will describe how we use the naïve Bayes algorithm in our project and what's the execution flow according to NB, In the mean time we will explain what the various components of the program do and how they do it. We will not descend into coding details, as the source code is available in *Appendix A* and is well commented.

5.1 Design

5.1.1 Naïve Bayes algorithm in our project

The classifier should be like this:

$$P(Y|X) = (P(X|Y) * P(Y))/P(X)$$

Where

- P(X) attributes that should be classified.
- P(Y) is the prior probability
- P(X|Y) is class conditional probability
- P(Y|X) is what we want to find out

Suppose the web page we want to test is a good web page, then

$$P(G|X) = (P(X|G) * P(G))/P(X)$$

Suppose the web page we want to test is a bad web page, then

$$P(B|X) = (P(X|B) * P(B))/P(X)$$

- P(X|G) is trained from good web pages
- P(X|B) is trained from bad web pages.

If $P(G|X) > P(B|X)$, then the document is good for dyslexics.

If $P(G|X) < P(B|X)$, then the document is bad for dyslexics.

In order to get $P(Y|X)$, first we should get $P(X|Y)$ by training samples. And we know that different metrics can lead to different $P(X|Y)$. So first we should conclude the

metrics which are suitable for our classification. Another question we should mention is we should balance the weight of different metrics, when we combine them.

5.1.2 main stages

Based on naïve Byes algorithm, the main stages in our project are as follows:

1. Choose the metrics

We know that there are many characteristics in the web pages. So we have to find the metrics which reflect these characteristics. Some metrics can not be used to distinguish the good pages and the web pages. So we will drop some of them by training and combine many metrics to make the classifier more effectively.

Metrics described in [1]:

- a. the length of words (4-6 letters)
- b. the length of sentences (15-20 words)
- c. use more “you” & “me”
- d. the font size (12-14 pt)
- e. avoid using block capital letters for emphasize
- f. avoid blocks of text in italics

2. Find the samples

In order to get the accuracy $P(X|G)$ and $P(X|B)$, we should search the good samples which can be easily distinguished by the dyslexics that whether they are readable or not. So we get the good web pages from the web site which are designed for the dyslexics or by some of them. On the other hand, we choose some specific web pages such as laws, mathematics, economics and so on which are difficult of the dyslexics. Our sample resources are the following [5].

- ✧ www.dyslexic.com
- ✧ www.ala.org
- ✧ www.jce.divched.org
- ✧ www.4ulr.com

This stage is very important. Because the result will not be accuracy , if the samples are not right. And we think if some dyslexics can help us to search some

web pages, the result will be better. The sample would be even better with the help of a person with dyslexia.

3. Train samples and get the $P(X|G)$ and $P(X|B)$

In this stage, we can drop some unsuitable metrics according to the training probability.

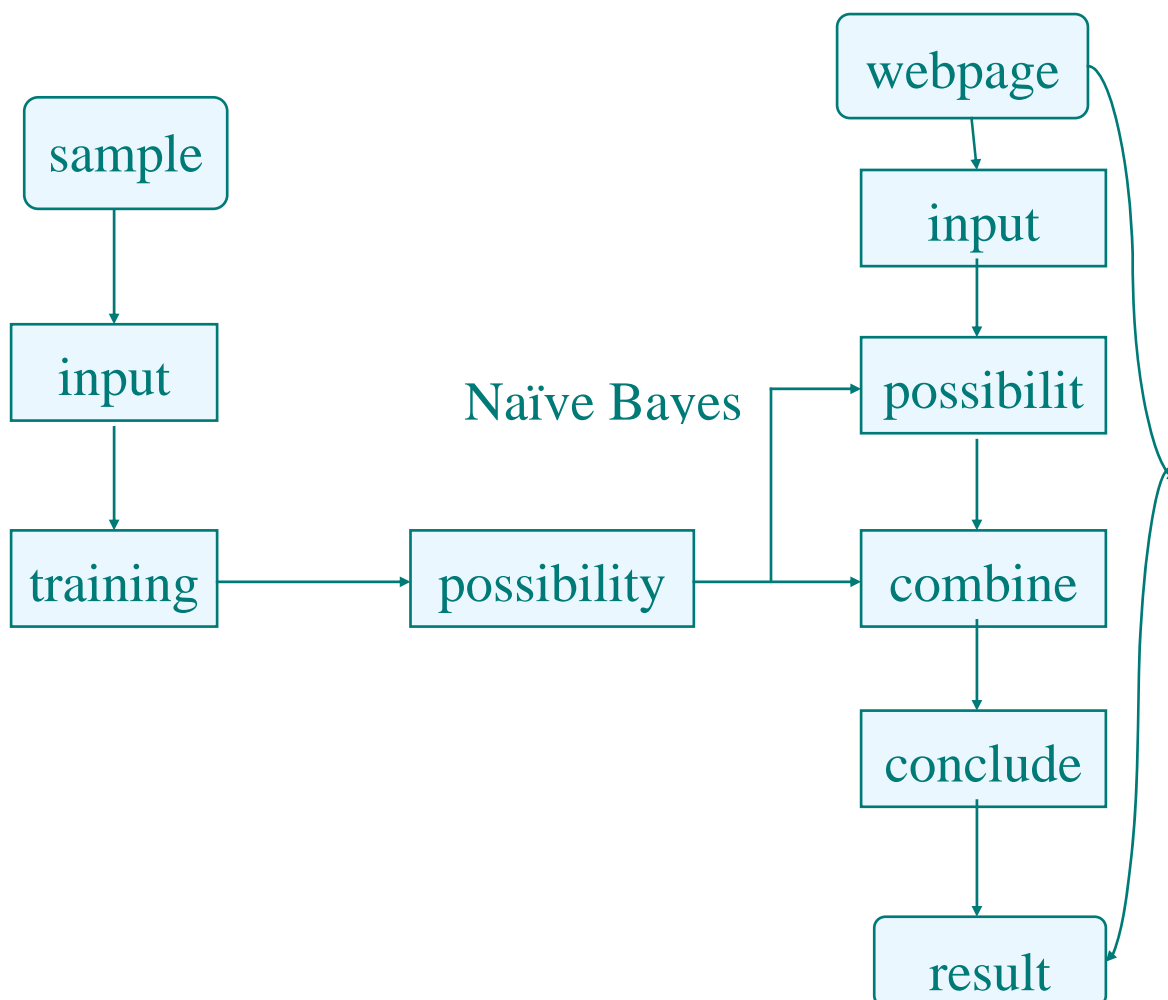
4. Use the NB to combine the metrics and compare the $P(G|X)$ and $P(B|X)$

We should balance the weight of different metrics to make every metric available.

If one of the metrics is unavailable in the classification, then the others can fix it.

5.1.3 Executive flow

The execution flow of the program is quite straight forward:

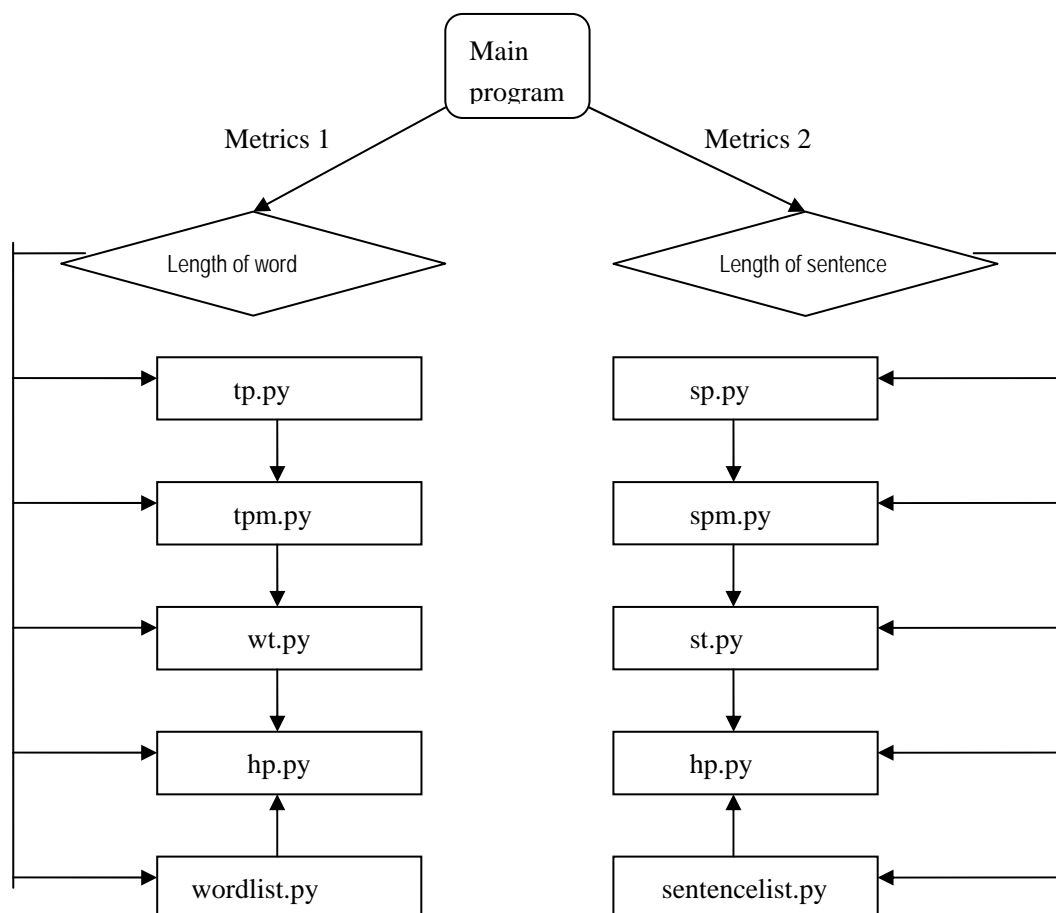


This was an overview of the top-level program design. Now we will take a closer look at the individual modules.

5.2 Implement

5.2.1 Structure of program

Based on the above program stages we created the following python modules:



Picture 1

5.2.2 Main program

The purpose of the main program module is to control the execution flow of the program, i.e. run the other modules in turn, act as the program entry point and combine the metrics and so on. It can be considered the spine of the program, to which all the other bones (modules) are attached. This is the module that is executed by testers.

```

C:\Python24> python mainnew.py -h
0.0401228206194 0.12974235901 0.165349003837 0.146331583282 0.0931097712365 0.09
75905415298 0.0832591710358 0.0754718658652 0.0533068184589 0.0391856463421 0.02
33190270217 0.0196036382379 0.013812131382 0.00495136008498 0.0021939275536 0.00
194650246227 0.0011323588352 0.00147935907715 0.00118996751657 0.000727737245692
682 2203 2675 2357 1472 1538 1387 1323 880 652 396 323 248 86 41 33 20 20 18 11
16462
0.0447841718415 0.131135280892 0.143635034578 0.11883482735 0.0880511301901 0.09
47910603965 0.0913477219197 0.0840856895035 0.0600626738549 0.043722204743 0.028
5151348687 0.0204456592121 0.0159013925269 0.00477394647901 0.00258394904969 0.0
0284035854115 0.00391624336815 0.00290626228486 0.00151616132585 0.0013361279418
2 712 2112 2137 1600 1218 1160 1192 1070 781 516 326 247 165 73 34 32 35 29 25 2
2 13690
-7364.47673674 -6175.16209187 5493
0.0869905584263 0.0374714165585 0.0402557640776 0.0220874093975 0.0273780869867
0.0341389498938 0.0239977751761 0.0225461664935 0.0333825647258 0.0280772978877
0.0273558561074 0.0296944103367 0.0371757901993 0.0337764342811 0.0194223832497
0.0390801759023 0.0334847884813 0.0239632083759 0.0230948290584 0.0245908796203
144 68 69 38 29 32 32 32 31 39 32 33 39 34 30 40 28 30 25 29 1139
0.133139390974 0.0731250228961 0.109350034856 0.0419519319457 0.0582025205282 0.
0352997184777 0.0188841857693 0.0260109573132 0.021016470505 0.0318287489292 0.0
288853014024 0.0170699565234 0.0313472133198 0.0158991396513 0.00986871223238 0.
009309350261 0.00965085402426 0.0186061748755 0.0147748348569 0.0137918630609 23
3 83 166 71 94 51 32 27 24 29 18 25 26 14 12 11 10 18 12 16 1221
-355.20461619 -366.663223778 621
-7719.68135293 -6541.82531565
gr: -7719.68135293 br: -6541.82531565 wgp: -7364.47673674 sgp: -355.20461619 wbp
-6175.16209187 sbp: -366.663223778
Bad

```

Picture 2

The first part is the conditional probabilities which we get by training samples, and the last part is the result "bad". So the web page we test is classified as bad for dyslexics to read.

We use two metrics (length of word and length of sentence) in our project. So we call the subprogram of each metric and combine them with weights. Finally, we compare the 'gr'(suppose it is good) and 'br'(suppose it is bad). If 'gr'>'br', then print 'good'. If 'br'>'gr', then print 'bad'. The codes are as follows:

```

import gp, hp

if __name__ == "__main__":

    t = gp.doTraining

    s = hp.doTraining

    wgp,wbp,c = t(testfile,goodfiles,badfiles)

    sgp,sbp,d = s(testfile,goodfiles,badfiles)

```

```
gr=wgp+sgp
br=wbp+sbp
print gr,br
print 'gr:',gr,'br:',br,'wgp:',wgp,'sgp:',sgp,'wbp',wbp,'sbp:',sbp
if gr>br:
    print "Good"
else:
    print "Bad"
#raw_input("")
```

5.2.3 Wordlist.py and sentencelist.py

This part is to divide a web page text into word list and sentence list. We know that the source code of web page is composed by many components. But we just abstract the text part according to our metrics. As there are space between the words ,so we divide the text into word according to space, and the codes are as follows:

```
text=re.sub(r"\<([^\<\>])*>',' ',data)
wordlist=text.split()
```

As there are '.' between two sentences. So we divide the text into sentences according to full stop, and the codes are as follows:

```
text=re.sub(r"\<([^\<\>])*>',' ',data)
sentences=text.split('.')
```

5.2.4 tp.py and sp.py

5.2.4.1 tp.py

This part is to get how many n-letter words there are in the web page and how many percent of the text is n-letter words. The codes are as follows:

```
for word in wordlist:
    if len(word)==n:
```

```

        count=count+1
total= len(wordlist)
possibility= float(count)/float(total)
return possibility, count, total

```

5.2.4.2 *sp.py*

This part is to get how many n-word sentences there are in the web page and how many percent of the text is n-word sentences. The codes are as follows:

for sentence in sentences:

```

        if len(sentence.split())==n:
            count =count+1
total =len(sentences)
h= float(count)/float(total)
return h, count, total

```

5.2.5 *tpm.py and spm.py*

This part is to get the average value of the conditional probability by training many samples , only in this way can we avoid the mistake which is caused by a unsuitable sample.

The codes of word metric and the codes of sentence metric are the same , but they use different subprograms tp.py and sp.py. the codes are as follows:

for file in range(len(files)):

```

        onea, oneb, onec = t(files[file],i)
        alist.append(onea)
        blist.append(oneb)
        clist.append(onec)

a= float(sum(alist))/len(alist) #(a1+a2+a3+a4+a5+a6+a7+a8+a9+a10)/10
b= sum(blist) #b1+b2+b3+b4+b5+b6+b7+b8+b9+b10
c= sum(clist) #c1+c2+c3+c4+c5+c6+c7+c8+c9+c10

```

```
return a, b ,c
```

5.2.6 wt.py and st.py

wt.py is to get the sum number of words and $P(i=n|Y)$ ($0 < n < 21$). And st.py has the same function as wt.py.

5.2.7 gp.py and hp.py

By using the subprogram wt.py in the section gp.py, we can get $P(i=n|G)$ ($0 < n < 21$) from good samples and $P(i=n|B)$ ($0 < n < 21$) from bad samples. According to NB algorithm, we calculate $P(G| i=n)$ ($0 < n < 21$) and $P(B| i=n)$ ($0 < n < 21$) which we use 'gprob' and 'bprob' instead.

We make the hp.py in the same way for metric sentence. And get $P(G| i=n)$ ($0 < n < 21$) and $P(B| i=n)$ ($0 < n < 21$) which we use 'sgprob' and 'sbprob' instead.

We found that there were too many words and sentences in the text. If we multiply all the $P(i=n|Y)$ ($0 < n < 21$), the result will be very small and python may recognize it as zero, So we use logarithm instead.

6 Results

We have tested the individual modules as well as the program as a whole using both python test code. This has ensured that the program is capable of performing all stages of the project satisfactorily. What we concern about is the accuracy of our classifier.

We have tested our classifier with 10 good web pages and 10 bad web pages from a subset for [3].In order to show the accuracy of our prototype, we present the over all results in the following table:

Result	Good	bad	percent
Test pages			
Good	8	2	80%
Bad	9	1	90%

Result of test pages(table 1)

7 Evaluation of result

So the result is not bad. But why are there still some disadvantages in our classifier. We have checked the conditional probabilities we have gotten by training. The result is showed in tables below (The first column, N, shows the length of the words / sentences):

N	$P(i = n G)$	$P(i = n B)$
1	0.0401228206194	0.0447841718415
2	0.12974235901	0.131135280892
3	0.165349003837	0.143635034578
4	0.146331583282	0.11883482735
5	0.0931097712365	0.0880511301901
6	0.0975905415298	0.0947910603965
7	0.0832591710358	0.0913477219197
8	0.0754718658652	0.0840856895035
9	0.0533068184589	0.0600626738549
10	0.0391856463421	0.043722204743
11	0.0233190270217	0.0285151348687
12	0.0196036382379	0.0204456592121
13	0.013812131382	0.0159013925269
14	0.00495136008498	0.00477394647901
15	0.0021939275536	0.00258394904969
16	0.00194650246227	0.00284035854115
17	0.0011323588352	0.00391624336815
18	0.00147935907715	0.00290626228486
19	0.00118996751657	0.00151616132585
20	0.000727737245692	0.00133612794182

Metric: length of word (table2)

So we can see that the differences between the good web pages and bad pages are not so obvious as $P(i=n | G)$, assuming that the classified page is good, and $P(i=n | B)$, assuming that the classified page is bad, is close to each other.. So It's not enough to classify a web page according to only one metric.

N	P (i = n G)	P (i = n B)
1	0.0869905584263	0.133139390974
2	0.0374714165585	0.0731250228961
3	0.0402557640776	0.109350034856
4	0.0220874093975	0.0419519319457
5	0.0273780869867	0.0582025205282
6	0.0341389498938	0.0352997184777
7	0.0239977751761	0.0188841857693
8	0.0225461664935	0.0260109573132
9	0.0333825647258	0.021016470505
10	0.0280772978877	0.0318287489292
11	0.0273558561074	0.0288853014024
12	0.0296944103367	0.0170699565234
13	0.0371757901993	0.0313472133198
14	0.0337764342811	0.0158991396513
15	0.0194223832497	0.00986871223238
16	0.0390801759023	0.009309350261
17	0.0334847884813	0.00965085402426
18	0.0239632083759	0.0186061748755
19	0.0230948290584	0.0147748348569
20	0.0245908796203	0.0137918630609

Metric: length of sentence (table 3)

It is showed that the length of sentence is better than the length of word as a metric. But it is also not enough for our classifier. And this is why our classifier is not so accuracy.

Another reason is that we can not get suitable enough samples to train. In fact, we are not dyslexics. But the good web page samples and the bad web page samples are chose in our opinion. So it is impossible to get the perfect samples. And it will result in unperfect conditional probabilities.

Last but not least reason is that the metrics we use are not enough or we should abstract some metrics which are better than we use now.

8 Further developments

Based on the reasons we have mentioned before, many things can be done to improve the program. Our suggestions are as follows:

Firstly, it is very necessary to contact with some dyslexic people. And it will be an effective way to collect samples from them. The more samples we get, the more accurate the result will be. In the mean time, it will be very wise to do a investigation with them and you may be find some good metrics.

Secondly, we'd better combine more metrics. Some metrics we have find are very good too. They can be abstract from the format of a web page such as font size, block capital letters, blocks of text in italics and so on which are reflected by css files. The problem is that each css files only contain 1-4 value of that kind of format. Take font size for example, each css file contain 1-4 font size, so we can not calculate the conditional probabilities as we did on length of word, first we should collect as many samples as possible, and calculate how percent of web pages is n font size web pages. And the rest job can be done in a similar way.

Finally, we could consider the web pages in different languages. If possible, we can also make the classifier which can have more subcategories.

9 Conclusions

The goal of our project is to make a classifier which can distinguish whether a web page is good for dyslexic people to read or not. Our program can training the samples to get conditional probabilities , take advantage of Naïve Byes algorithm to combine the different metrics and compare the values to decide the result.

We found that our classifier is more effective to distinguish the bad web pages than the good web pages (The correct results account 90% and 80% separately).

We believe that we can get more accurate result by using more suitable samples and combine more metrics on the format of a web page.

10 References

- [1] Dyslexic.com - <http://dyslexic.com>
- [2] Web Content Accessibility Guidelines -
<http://www.w3.org/TR/WAI-WEBCONTENT/>
- [3] Python Programming language – www.python.org
- [4] The former report - <http://eiao.net/webmining/previousprojects>
- [5] Several samples of our training part:
- i. www.dyslexic.com
 - ii. www.ala.org
 - iii. www.jce.divched.org
 - iv. www.4ulr.com
- [6] Master Thesis: <http://eiao.net/webmining/previousprojects>
- [7] <DATA MINING> Ian H.Witten & Eibe Frank
- [8] <DATA MINING> Margaret H.Dunham

11 Appendix

This appendix contains the Python source code developed during the project.

11.1 wordlist.py and sentencelist.py

```
import sys      #!/ sentencelist.py

def doTraining( file):

    f = open(file)

    data = ".join(f.readlines())

    f.close()

    import re

    text=re.sub(r'\<([\^\<\>])*>', '',data)

    wordlist=text.split()/sentences=text.split('.')

    return wordlist/sentences
```

11.2 tp.py and sp.py

11.2.1 tp.py

```
import sys

def doTraining( file,n):

    print file

    f = open(file)

    data = ".join(f.readlines())

    f.close()

    import re

    text=re.sub(r'\<([\^\<\>])*>', '',data)

    wordlist=text.split()

    count=0

    for word in wordlist:

        if len(word)==n:
```

```
        count=count+1
total=len(wordlist)
possibility=float(count)/float(total)
return possibility,count,total
```

11.2.2 *sp.py*

```
import sys
def doTraining( file,n):
    f = open(file)
    data = ".join(f.readlines())
    f.close()
    import re
    text=re.sub(r'\<([\^\<\>])*>', ' ',data)
    sentences=text.split('.')
    count=0
    for sentence in sentences:
        if len(sentence.split())==n:
            count=count+1
    total=len(sentences)
    h=float(count)/float(total)
    return h,count,total
```

11.3 *tpm.py* and *spm.py*

```
import sys
import tp/sp #for: tpm.py/spm.py
def training(files ,i):
    t=tp.doTraining/sp.doTraining
    alist = []
    blist = []
    clist = []
```

```
for file in range(len(files)):
    onea, oneb, onec = t(files[file],i)
    alist.append(onea)
    blist.append(oneb)
    clist.append(onec)

a=float(sum(alist))/len(alist)
b=sum(blist) #b1+b2+b3+b4+b5+b6+b7+b8+b9+b10
c=sum(clist) #c1+c2+c3+c4+c5+c6+c7+c8+c9+c10
return a,b,c
```

11. 4 wt.py and st.py

```
import tpm/spm #: for wt.py/ st.py
```

```
def operation( files):
```

```
    t=tpm.training/spm.training
    p1,l1,c1=t(files,1)
    p2,l2,c2=t(files,2)
    p3,l3,c3=t(files,3)
    p4,l4,c4=t(files,4)
    p5,l5,c5=t(files,5)
    p6,l6,c6=t(files,6)
    p7,l7,c7=t(files,7)
    p8,l8,c8=t(files,8)
    p9,l9,c9=t(files,9)
    p10,l10,c10=t(files,10)
    p11,l11,c11=t(files,11)
    p12,l12,c12=t(files,12)
    p13,l13,c13=t(files,13)
    p14,l14,c14=t(files,14)
    p15,l15,c15=t(files,15)
    p16,l16,c16=t(files,16)
```

```

p17,l17,c17=t(files,17)
p18,l18,c18=t(files,18)
p19,l19,c19=t(files,19)
p20,l20,c20=t(files,20)
c=(c1+c2+c3+c4+c5+c6+c7+c8+c9+c10+c11+c12+c13+c14+c15+c16+c17+c
18+c19+c20)/20
print
p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p17,p18,p19,p20,l1,l2,l3,l
4,l5,l6,l7,l8,l9,l10,l11,l12,l13,l14,l15,l16,l17,l18,l19,l20,c

result=[(p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p17,p18,p
19,p20),(l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12,l13,l14,l15,l16,l17,l18,l19,l20),c]
c = result[len(result)-1]
trainingdict = {}
for i in range(len(result[0])):
    trainingdict[i] = result[0][i]
    if trainingdict[i] == 0:
        trainingdict[i] = 0.00001
return c,trainingdict

```

11.5 gp.py and hp.py

```

import wt,wordlist,math

/import st,sentencelist, math

def doTraining( file,files,files1):

    #print files

    t = wt.operation/ t=st.operation

    c,gooddict = t(files)

    d,baddict = t(files1)

    pg=float(c)/float(c+d)

    pb=1-pg

```

```
f=wordlist.doTraining/sentencelist.doTraining
wlist = f(file)
lengths = map(len,wlist)
glist = [gooddict[i] for i in lengths if i in gooddict]
blist = [baddict[i] for i in lengths if i in baddict]
m = max(glist+blist)
gprob = 0
for i in glist:
    gprob += math.log(i)
gprob=pg*gprob
bprob = 0
for i in blist:
    bprob += math.log(i)
bprob=pb*bprob
print gprob,bprob,len(wlist)
return gprob,bprob,len(wlist)
```

11.6 mainnew.py

```
import gp,hp
if __name__ == "__main__":
    t = gp.doTraining
    s = hp.doTraining
    wgp,wbp,c = t(testfile, good samples, badsamples)
    sgp,sbp,d = s(testfile, good samples, badsamples)
    #gr=wgp*(sgp**(float(c)/float(d)))
    #br=wbp*(sbp**(float(c)/float(d)))
    gr=wgp+sgp
    br=wbp+sbp
    #Getting number for bad
    print gr,br
```

```
print 'gr:',gr,'br:',br,'wgp:',wgp,'sgp:',sgp,'wbp',wbp,'sbp:',sbp
if gr>br:
    print "Good"
else:
    print "Bad"
#raw_input("")
```