



E-content accessibility for dyslexics

by

Zhang Xiaorui

Supervisor: Annika Nietzio and Morten Goodwin Olsen

Project report for ICT407, Autumn 2006

Agder University College
Faculty of Engineering and Science

Grimstad, 27 November 2006
Status: Final

Keywords: Dyslexic, Accessibility, Naive Bayes classifier, Flesch reading ease test

Abstract:

This project is to develop a WAM (Web Accessibility Metric) that can be used within EIAO (European Internet Accessibility Observatory) to evaluate accessibility for people with reading difficulties. The new metric can be deduced from the score produced by Flesch reading ease test that describes the special requirements for content presentation

Licence: This work is licensed under the [Creative Commons Attribution-ShareAlike License](http://creativecommons.org/licenses/by-sa/2.5/) (<http://creativecommons.org/licenses/by-sa/2.5/>) or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Version Control

Version¹	Status²	Date³	Change⁴	Author⁵
0.1	Draft	2006-11-18	Original creation of the report	Zhang Xiaorui
0.2	Draft	2006-11-21	Chapter 3, 4 edited	Zhang Xiaorui
0.3	Draft	2006-11-22	Chapter 5, 6 added	Zhang Xiaorui
0.4	Draft	2006-11-23	Chapter 5 edited, chapter7, A1, A2 added	Zhang Xiaorui
0.5	Review	2006-11-24	Table of contents updated, spelling check	Zhang Xiaorui
0.6	Final	2006-11-24	Read through	Zhang Xiaorui

¹ **Version** indicates the version number starting at 0.1 for the first draft and 1.0 for the first review version.

² **Status** is DRAFT, REVIEW or FINAL

³ **Date** is given in ISO format: yyyy-mm-dd

⁴ **Change** describes the changes carried out since the previous version

⁵ **Author** is the one who did the change

Table of Contents

1	Introduction	5
1.1	Acknowledgement.....	5
1.2	Proposal outline	6
2	Problem description	7
2.1	Assignment.....	7
2.2	Approach	7
3	Background.....	8
3.1	Document classification	8
3.2	Bayes Theorem	8
3.3	Flesch reading ease test.....	9
3.4	What is dyslexic.....	9
3.5	Recent studies in e-content accessibility for dyslexic people.....	9
4	Solution	11
5	Verification and Testing.....	20
6	Further development.....	26
7	Conclusion.....	27
Appendices		
A1	Glossary & Abbreviations.....	28
A2	References.....	28
A3	Code work in this project.....	30
A4	Progress plan for the main project.....	33

1 Introduction

“You can’t just throw information up there, and clutter up cyberspace. Anybody who makes a web site should make the effort to organise the information.” (Morkes and Nielsen, 1997)[1]

“It is estimated that between 1.2% and 1.5% of students in Higher Education in the UK are dyslexic” (Singleton, 1999) [1].

Today, web sites can follow all W3C recommendations to increase the accessibility of them. But, besides the technical aspects of accessibility, the readability of content is commonly overlooked when working with a web site [2]. And the situation gets even worse when it comes to the accessibility of e-content for people with dyslexia, for the group has its specific requirements to meet. To support dyslexic people, there is need for the awareness of increasing the accessibility of e-content for them [1].

This project when done should be used for helping estimate whether the web page is accessible for dyslexic people or not accessible for them. With the help of that, people with dyslexia can evaluate web pages in advance, which might help them with their choices. Furthermore, the project might also give assistance to establishers of web sites, when they are trying to present more accessible e-content for dyslexic people in their own sites.

“The readability issue involving in this project is not an exact science, instead they are only statistical tests -at best they are helpful assistance, and at worst they are totally unreliable. And evaluation for web pages is often a tough task even for humans, so human supervision is essential when using the metric in this project.” [3]

1.1 Acknowledgement

“Awareness and support for dyslexic students has increased dramatically in the past decade” (Gilroy & Miles, 1996) [1]. People are paying more attention to increasing the e-content accessibility for dyslexic people. In that field, many related metrics have been implemented in Perl, Python and other implementation languages. And most of them are open source, so some can be referred in this project. There are also related essays focusing on this issue which might help clarify important concepts when working with this project.

This project refers much to the one done by Group 6 last year [4]. In contrast with the previous work, this project takes Naive Bayes algorithm as the basis of classifier as well, but uses different metrics. In the previous work, two metrics are adopted to evaluate the chosen web pages-one is for measuring the length of word and the other is for measuring the length of sentence, while this project would like to take Flesch reading ease test [5] as metric for readability evaluation. In conclusion, this project inherits the framework of the previous one, but tries to make improvements with the new metric. To make comparison of the accuracy of these two projects, previous data will be adopted this year.

1.2 Proposal outline

The following chapter gives a problem description focusing on problem definition, related research questions and the strategy of our solution to the problem.

In chapter 3, it lists the literature survey and findings to the problem and any existing standards which can help solve it. Besides, a good review of the current status in the field of the problem is also given in the chapter.

The chapter 4 presents the requirements for the solution under development. Detailed design and implementation of this project are both listed in this chapter.

The chapter 5 includes training data and test data which are used for verifying and testing the metric. Evaluation of the results is also stated.

The chapter 6 gives a broad view of further development of this project in the future.

The following chapter comes to a conclusion of the whole task in this project.

In the Appendices, glossary & abbreviations, references, code in this project and progress plan are all included.

2 Problem description

2.1 Assignment

The purpose of this project is to develop a WAM (Web Accessibility Metric) that can be used within EIAO (European Internet Accessibility Observatory) to produce special scores about accessibility for people with reading difficulties. The new metric can be deduced from the literature that describes the special requirements for content and layout / presentation[6].

2.2 Approach

For there are various WAM (web accessibility metrics) focusing on different fields, we put emphasis on making metrics which particularly evaluate textual aspect of the e-content. The goal of this project is to make a text classifier that classifies inputs into the following defined categories [4] :

- Accessible for people with dyslexia
- Not accessible for people with dyslexia

The text classifier is based on Naïve Bayes algorithm which is one of the most widely-acknowledged successful algorithms which applies to text classification. Classifier model can be induced from training set of data before test set of data used for testing the accuracy of the model.

In the project, Flesch reading ease test is adopted to give readability scores to the web pages which will be later used as observations in the classifier. It is noted that Flesch-Reading Ease test is designed to indicate how difficult a reading passage is to understand, and has become a U.S governmental standard [5].

3 Background

3.1 Document classification

“Document classification/categorization is a problem in information science. The task is to assign an electronic document to one or more categories, based on its contents. Document classification tasks can be divided into two sorts: supervised document classification where some external mechanism (such as human feedback) provides information on the correct classification for documents, and unsupervised document classification, where the classification must be done entirely without reference to external information.” [7]

Document classification techniques include [7]:

- Naive Bayes classifier
- Tf-idf
- Latent semantic indexing
- Support vector machines
- Artificial neural network
- KNN
- Concept Mining

3.2 Bayes Theorem

Bayes Theorem [8]:

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

- Given a set of data $X = \{x_1, \dots, x_n\}$
- Suppose that either hypothesis h_1 or h_2 may occur, but not both
- $P(h_1 | x_i)$ 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 the data value x_i
- $P(x_i | h_1)$ is the conditional probability

For Bayes Theorem allows us to assume that each value in $X = \{x_1, \dots, x_n\}$ is conditionally independent, it has high efficiency when using in assigning a sample into one of the predefined categories. Besides, it is fairly straightforward to code. The two reasons make it served as one of the most well-known and successful algorithms in classification.

3.3 Flesch reading ease test

“In the Flesch Reading Ease test, higher scores indicate material that is easier to read; lower numbers mark harder-to-read passages. The formula for the Flesch Reading Ease Score (FRES) test is:

$$206.835 - 1.015\left(\frac{\text{TotalWords}}{\text{TotalSentences}}\right) - 84.6\left(\frac{\text{TotalSyllables}}{\text{TotalWords}}\right)$$

“As a rule of thumb, scores of 90–100 are considered easily understandable by an average 5th grader. 8th and 9th grade students could easily understand passages with a score of 60–70, and passages with results of 0–30 are best understood by college graduates. Reader’s Digest magazine has a readability index of about 65, Time magazine scores about 52, and the Harvard Law Review has a general readability score in the low 30s.

This test has become a U.S. governmental standard. Many government agencies require documents or forms to meet specific readability levels. The U.S. Department of Defense uses the Reading Ease test as the standard test of readability for its documents and forms.” [5]

3.4 What is dyslexic

People with dyslexia often have problems with [1] :

- Visual processing (inc. scotopic sensitivity),
- Phonological decoding, analysis and processing,
- Reading and comprehension,
- Auditory processing,
- Memory recall,
- Structure and sequencing,
- Planning and organization.

These problems have an impact on the user’s ability to read, write, navigate, comprehend and recall relevant information from electronic materials e.g. web sites. “In addition, within the dyslexic population as a whole a number of underlying processing difficulties present themselves as various sub-types of dyslexia” (Siegel, 1994; Rack, 1996) [1].

3.5 Recent studies in e-content accessibility for dyslexic people

Current web accessibility studies for people with dyslexia mainly focus on the research affecting people with visual impairments and the Blind. Thus, many metrics concerning colour contrast and font size are available, either open source or for commercial purpose. However, less metrics have been implemented that specifically deals with the issues surrounding e-content accessibility from a perspective of readability [1]. But it is also good to have sufficient related essays as guidelines when working in that field.

There is a similar project done by Group 6 last year with the same project assignment. In the previous work, it also takes Naive Bayes algorithm as the basis of classifier, but uses

different metrics-one for measuring the length of word and the other for measuring the length of sentence. That is the way of readability evaluation in that project. It does work by combining the two metrics, then train and test the output into a Naive Bayes classifier, which gains a satisfied accuracy as 90% [4].

4 Solution

4.1 Requirements Specification

4.1.1 Functional requirements

Requirement 1: Preprocessing of web pages,

Priority: A1

Description: The chosen web page can be removed from html makeup firstly, and then split into list of words and list of sentences.

Testing: Web pages provided to test the module independently, later check whether the html makeup are removed completely

Requirement 2: Count the total words in the text,

Priority: A2

Description: Count how many words are there in the list of words.

Testing: Word list provided to test the module independently, later evaluate the accuracy of it.

Requirement 3: Count the total sentences in the text,

Priority: A3

Description: Count how many sentences are there in the list of sentences.

Testing: Sentence list provided to test the module independently, later evaluate the accuracy of it.

Requirement 4: Count the total syllables in the text,

Priority: A3

Description: Count how many syllables are there in the list of words.

Testing: Word list provided to test the module independently, later evaluate the accuracy of it.

Requirement 5: Evaluate readability of the text,

Priority: A4

Description: Grade the text according to Flesch reading ease test .

Testing: All the above provided to test the module independently, later evaluate the accuracy of it.

Requirement 5: Document classification,

Priority: A5

Description: Building a classifier which manages to assign the text into the most probable category.

Testing: Test the module independently with some simple and direct data input.

Requirement 6: control of work flow,

Priority: A6

Description: Building a function which manages to combine all the above and presents itself as an executable program.

Testing: Execute the main function as you suppose users do.

The requirements are given in a hierarchical way and it is understood that the real requirements are always on the last level. For each requirement it is started which priority it has A and its state.

4.1.2 Non-functional requirements

In this project, Python is chosen as implementation tool.

“Python is a remarkably powerful dynamic programming language that is used in a wide variety of application domains. Python is often compared to Tcl, Perl, Ruby, Scheme or Java. Some of its key distinguishing features include:

- *very clear, readable syntax*
- *strong introspection capabilities*
- *intuitive object orientation*
- *natural expression of procedural code*
- *full modularity, supporting hierarchical packages*
- *exception-based error handling*
- *very high level dynamic data types*
- *extensive standard libraries and third party modules for virtually every task*
- *extensions and modules easily written in C, C++(or Java for Jython, or .Net languages for IronPython)*
- *embeddable within applications as a scripting interface” [9]*

4.2 Design

4.2.1 Flesch reading ease test

Formula of Flesch reading ease test:

$$206.835 - 1.015\left(\frac{\text{TotalWords}}{\text{TotalSentences}}\right) - 84.6\left(\frac{\text{TotalSyllables}}{\text{TotalWords}}\right)$$

Each web page used for either training or testing has to be graded by the following formula. The score produced indicates how difficult the text in the page is to understand. And the score will be later used as observation of particular sample in the further process of classification by Naïve Bayes classifier.

To begin, the html file is firstly parsed and removed html notation as ‘<>’, and then converted into string so as to lay a basis for further operation. After obtaining the plain form of the web page, statistic of “total words”, “total sentences” and “total syllables” in it can be gained as follows:

- Split the file into a list of words, get the “total words” by calculating the length of the list
- Split the file into a list of sentences, get the “total sentences” by calculating the length of the list
- Split the file into a list of words, use an specific algorithm to estimate the “total syllables” (the algorithm makes the estimation by counting the vowel groups and etc.)

With the statistics above, the particular web page can be graded with the score of the test, which is produced by calculating the result of the formula. And the score will be later used as the observation of the sample in the process of classification.

4.2.2 Naive Bayes algorithm in this project

Naive Bayes algorithm is specified in this project as follows [4]:

$$P(cls_i|obs) = \frac{P(obs|cls_i) * P(cls_i)}{P(obs)}$$

- Given a observation as *obs* of particular sample. In this project, it is specified as the score of a chosen web page according to Flesch reading ease test.
- Suppose that either hypothesis cls_1 or cls_2 may occur, but not both. In the classifier in this project:
 - cls_1 represents the class “Accessible for people with dyslexia”
 - cls_2 represents the class “Not accessible for people with dyslexia”
- $P(cls_i)$ is the prior probability associated with hypothesis cls_i . It is used for training process of the classifier. In this project, $P(cls_1)$ indicates the percentage of web pages in class “Accessible for people with dyslexia” while $P(cls_2)$ indicates the percentage of web pages in the other class
- $P(obs)$ is the probability of the occurrence of the observation *obs*. It can be ignored in this project when it comes to the actual process of classification.
- $P(obs|cls_i)$ is the conditional probability. It is the probability of occurrence of given observation from particular class. It can be calculated by making statistic of frequency of a web page with particular score and that of the given class.
- $P(cls_i|obs)$ is the posterior probability, indicating the probability of occurrence of particular class by giving observation in advance.

From the above formula, further classification is operated as follows:

- If $P(cls_1|obs) > P(cls_2|obs)$, then the web page is accessible for people with dyslexia
- If $P(cls_1|obs) < P(cls_2|obs)$, then the web page is not accessible for people with dyslexia

4.2.3 Main stages

The main stages designed in this project represents as follows [4]:

- Choosing metrics

To begin, it is essential to choose metrics, which can distinguish good and bad web pages by evaluating one of the web page’s characteristic. Flesch reading ease test is adopted in this project. It measures the readability by computing correlation between words, sentences and syllables in the text, which is also considered when trying to distinguish accessible web pages for dyslexic people. The metric includes several sub-metrics:

 - segmentation
 - word split & count
 - sentence split & count
 - syllable count.

- formula calculation

- Collecting samples

Sufficient samples as web pages are critical in both training and testing stages in this project. The samples contain either accessible or inaccessible web pages for dyslexic people. And which class they belong to is known in advance. The samples will be divided into training samples and testing samples. At first, training samples are used to train the classifier based on Naive Bayes algorithm. And when it comes to the testing stage, testing samples are adopted to evaluate the accuracy of the classifier, which might help make adjustment in the implementation.

For this stage will affect the accuracy of the classifier significantly, the samples must be chosen correctly and appropriately. The best way of doing that is to make an investigation with dyslexic participants, who can choose accessible and inaccessible web pages of their choices. But with the time and condition available, it is difficult to do so in this project. Instead, we collect web pages especially designed for dyslexic people or some of their own web pages. In order to compare the result with previous work on this field, previous samples are used in this project.

- Training the classifier

With inputting the training samples chosen at last stage, train the classifier to get $P(cls_i|obs)$ for each of the training set. It is the core of the classifier that will be later used for document classification.

- Verification and testing

After training, apply the classifier to the testing set and then compare the results with their predefined class. From comparison, we get the accuracy of the classifier.

4.3 Implementation

4.3.1 Architecture

Python is used as implementation tool in this project. According to the above design, here comes the architecture of all the python modules in the code work:

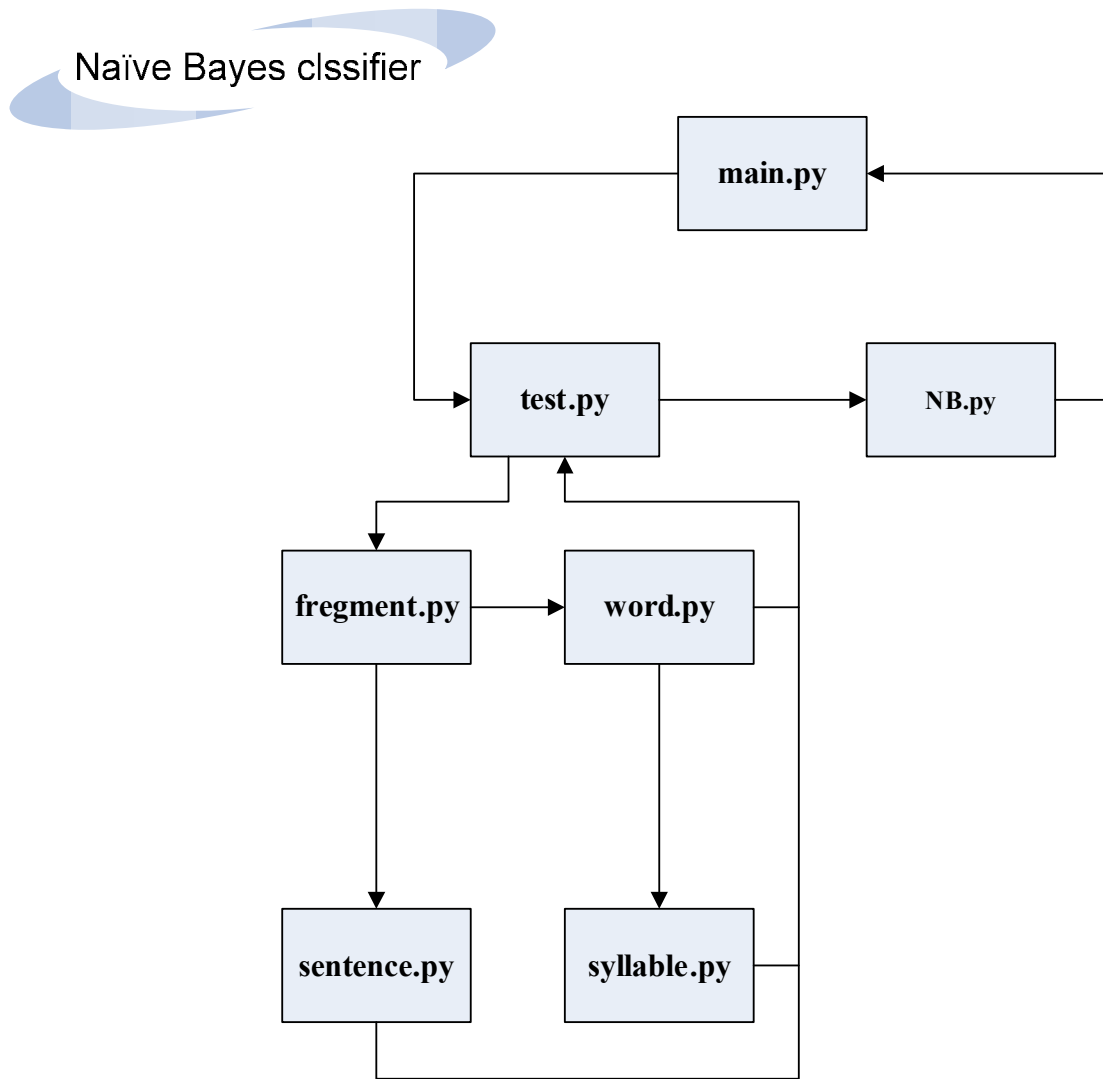


Figure 1 Architecture of modules

4.3.2 Fragment.py

The module is used first to convert the html file into the form of string. With the plain form of html file, split the text into list of words and list of sentences for further use.

In this module, we generally reuse wordlist.py and sentencelist.py [4] in the previous work with tiny changes.

The pseudocode for this module presents as follows:

```

f = open (file)
data = ' f '
text = substitute '<content>' with ' ' in data
wordlist = text. lower().split()
sentencelist = text. lower(). split('.')
  
```

In the real code, the substitute process will be implemented by the use of regular expressions. And particular string method will be used in the split process.

4.3.3 word.py and sentence.py

These two modules use the same method to count how many words or sentences there are in the word list or sentence list.

The pseudocode for this module presents as follows:

```
total_words = len(wordlist)
total_sentences = len(sentencelist)
```

For using different metrics, such modules are not contained in the previous work.

4.3.4 syllable.py

This module estimates the number of syllables in the word passed to it. Note that when working with this module we reuse this metric [10]:

<http://viewvc.red-bean.com/gnoetics/src/syllables.py?revision=1&view=markup>

The following presents the pseudocode:

```
sub_syllable = ["cial", "tia", "cius", "cious", "gui", "ion", "iou",
               "sia$", ".ely$"]
add_syllable = ["ia", "riet", "dien", "iu", "io", "ii",
               "[aeiou]bl$", "mbl$",
               "[aeiou]{3}",
               "^mc", "ism$",
               "(.)(?!\\1)([aeiou])\\2!$",
               "[^]llien",
               "^coad.", "^coag.", "^coal.", "^coax.",
               "(.)(?!\\1)[gq]ua(.)(?!\\2)[aeiou]",
               "dnt$"]
```

```
for word in wordlist:
    # Remove the silent 'e'
    if ends with 'e':
        remove the 'e'

    # Add & subtract syllables
    for syllable in add_syllable:
        if syllable in word:
            add syllables
    for syllable in sub_syllable:
        if syllable in word:
            subtract syllables

    # Count vowel groups
    for letter in word:
        if is_vowel and not prev_was_vowel:
            add syllables
```

The metric we reuse is implemented in Python, but the metric itself is based on the algorithm used in `Lingua::EN::Syllable::syllable()` that implemented in Perl :

<http://search.cpan.org/~gregfast/Lingua-EN-Syllable-0.251/Syllable.pm>

Note that it cannot gain an accuracy of 100%, instead it fails (by one syllable) for about 10-15% in the word list. The only way to get entire accurate count is to look up in a dictionary, so this is only a small and fast alternative [11].

4.3.5 test.py

This module grades text according to Flesch reading ease test.

The pseudocode presents as follows:

```
# Based on Flesch reading ease test
score = 206.835 - 1.015*( total_words/total_sentences)-84.6*(total_syllables/total_words)
score = int(score)

if score < 0:
    score = 0
if score > 206:
    score = 206
```

According to practical situation, we set appropriate upper limit and lower limit to scores graded by the test.

4.3.6 NB.py

A Naive Bayes classifier is built up in this module. The classifier has two components, one is training part, and the other is testing part. With the result of training part, the testing part manages to assign the test sample into its probable class. Note that it may fail to classify due to insufficient training samples.

The pseudocode presents as follows:

```
class NaiveBayes:
    def __init__(self):
        self.prior = {} # Frequency of each class
        self.total = {} # Frequency of each (class,value)-tuple
        self.count = 0 # Number of observations

    def add(self, cls, obs):
        'Adds an observation to the classifier'
        self.prior[cls] = self.prior.get(cls, 0) + 1
        key = cls, obs
        self.total[key] = self.total.get(key, 0) + 1
        self.count += 1

    def discr(self,cls, obs):
        'Bayesian discriminant. Proportional to posterior probability'
        res = self.prior[cls]/self.count
        freq = self.total.get((cls,obs), 0)
        result =res* freq/self.prior[cls]
        self.total[(cls,obs)] = result
        return result

    def classify(self, obs):
        f self.total.has_key(('good',obs)):
            gp = self.total.get(('good',obs))
        else:
            gp=0
        if self.total.has_key(('bad',obs)):
```

```
        bp = self.total.get(('bad',obs))
else:
    bp=0
if gp>bp:
    print 'Accessible for dyslexic people'
elif gp<bp:
    print 'Not accessible for dyslexic people'
elif gp==bp:
    print 'fail to classify according to the training dictionary'
```

Compared with previous work, this module improves the previous one by taking advantage of class type to become more compact and readable.

4.3.7 Main module

The main module is the one controlling the execute flow of the total program. It is executed by users of this program. After obtaining the web page for classification, the module invokes methods defined in other modules to get the final result.

The pseudocode represents as follows:

```
import NB, test
nb = NaiveBayes()
for each file in trainingset:
if file is good:
nb.add(file,'good')
if file is bad:
nb.add(file,'bad')
for test file
score = test.test(test file)
assigned_ class = nb.classify(score)
print assigned_ class
```

From the above code, it is clearly to see at first the module invokes the test() method in test.py to get score for the web page. Then add the observation as score into the training part of the classifier. After the training work, the module invokes the classify() method in NB.py to assign the chosen web page into the most probable class according to the result of training.

In contrast with the main module made last year, these two both follows the “first training, then testing” logic. But due to differences in some other modules, they are not quite the same in detailed implementation. For instance, for loop is used to add files of training set iteratively into the training part of the classifier.

The screenshot when executing the main.py is as follows:

```

Python 2.4.2 (#67, Sep 28 2005, 12:41:11) [MSC v.1310 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.

*****
Personal firewall software may warn about the connection IDLE
makes to its subprocess using this computer's internal loopback
interface. This connection is not visible on any external
interface and no data is sent to or received from the Internet.
*****

IDLE 1.1.2      ==== No Subprocess ====
>>>
Enter the file you want to classify:E:\python\Testdic\yy6.htm
({'bad', 54): 0.016666666666666666, ('bad', 50): 0.050000000000000003, ('good',
54): 0.016666666666666666, ('bad', 34): 0.016666666666666666, ('good', 33): 0.05
0000000000000003, ('bad', 26): 0.00125, ('bad', 39): 0.012500000000000001, ('bad
', 38): 0.012500000000000001, ('good', 14): 0.012500000000000001, ('good', 39):
0.025000000000000001, ('bad', 40): 0.00125, ('bad', 14): 0.012500000000000001, (
'good', 26): 0.01, ('bad', 33): 0.050000000000000003)
48 fail to classify

```

Figure 2 Screenshot of running the main.py

From the screenshot, we can see that at first the program asks for the file name for evaluation. Then the second part of the screenshot displays the dictionary $\{(cls, obs): P(cls|obs)\}$ after training process. In the last line, test file's score graded by Flesch reading ease test and its most probable class are both shown. And it is clearly to see that the classifier fails to assign the test sample to any classes due to insufficient training samples. The accuracy will be improved by training with more samples.

5 Verification and Testing

5.1 Results

In order to contrast with previous work, training data and test data are retrieved from last year's group. In training set and test set, accessible web pages were all retrieved last year from the following web site [12]:

"The British Dyslexia Association – Indications for dyslexia" www.bdaweb.co.uk.

Besides, inaccessible web pages in the two sets were all cases and codes in law that were retrieved from the following web site last year [13]:

"Find Law – for Legal Professionals" <http://www.findlaw.com/casecode/>

- **Test with training data**

Before testing the accuracy with test data, at first we can test it with training data which have been used in the training process.

The following table presents the testing result with training pages:

<i>Test pages</i> <i>Result</i>	<i>Accessible for dyslexic people</i>	<i>Not accessible for dyslexic people</i>	<i>Accuracy</i>
Accessible for dyslexic people	2	0	40%
Not accessible for dyslexic people	0	5	100%
Fail to classify according to the training dictionary	3	0	

Table 1 result of testing with training pages

- **Test with test data**

The best way to verify the classifier is to test it with test pages but not with the same training pages as shown above. From that way, we can estimate the accuracy of it when the user tries to classify a new web page.

The following table presents the result with test pages:

<i>Test pages</i> <i>Result</i>	<i>Accessible for dyslexic people</i>	<i>Not accessible for dyslexic people</i>	<i>Accuracy</i>
Accessible for dyslexic people	0	0	0%

Not accessible for dyslexic people	1	1	50%
Fail to classify according to the training dictionary	4	4	

Table 2 result of testing with test pages

5.2 Evaluation of results

From the above results, we can see that when testing with test data, in the 5 good samples for testing, 4 of them are failed to be classified and the rest is assigned to the wrong class. The unsatisfied situation turns better when it comes to the bad samples for testing, 4 of them are also failed but one is classified correctly.

In the following we will set assumptions to figure out reasons for this pretty low accuracy of the classifier:

- Assumption 1 Bugs in the fragment.py

For verifying this assumption, we add the following command lines into the module to test it independently:

```
print word_split('E:\python\Testdic\yy6.htm')
print sentence_split('E:\python\Testdic\yy6.htm')
```

Then it comes the result:

```
word list = ['plain', 'english', 'campaign:', 'crystal', 'mark', 'and', 'editing:', 'questions', 'about', 'the', 'crystal', 'mark', 'you', 'are', 'in', 'the', 'crystal', 'mark', 'and', 'editing', 'section.', 'how', 'the', 'crystal', 'mark', 'works', ':', 'in', '1990', 'we', 'introduced', 'the', 'public's', 'own', 'seal', 'of', 'approval', '-', 'the', 'crystal', 'mark', '-', 'to', 'encourage', 'organisations', 'to', 'communicate', 'clearly', 'with', 'the', 'public.', 'the', 'crystal', 'mark', 'has', 'now', 'become', 'firmly', 'established', 'in', 'the', 'uk', 'as', 'the', 'standard', 'that', 'all', 'organisations', 'aim', 'for', 'when', 'they', 'produce', 'public', 'information.', 'the', 'mark', 'also', 'appears', 'on', 'documents', 'in', 'other', 'countries', 'such', 'as', 'the', 'usa,', 'australia', 'and', 'south', 'africa.', 'more', 'than', '1000', 'organisations', 'know', 'that', 'only', 'plain', 'english', 'campaign's', 'crystal', 'mark', 'will', 'be', 'accepted', 'by', 'the', 'public', 'as', 'a', 'guarantee', 'of', 'a', 'document's', 'clarity.', 'this', 'is', 'because', 'we', 'will', 'not', 'give', 'the', 'crystal', 'mark', 'to', 'any', 'document', 'unless', 'our', 'testing', 'shows', 'it', 'can', 'be', 'read,', 'understood', 'and', 'acted', 'upon', 'by', 'the', 'intended', 'audience.', 'what', 'we', 'look', 'for', 'things', 'we', 'look', 'for', 'include:', 'a', 'good', 'average', 'sentence', 'length', '(about', '15', 'to', '20', 'words);', 'plenty', 'of', 'active"', 'verbs', '(instead', 'of', 'passive"', 'ones);', 'everyday', 'english;', 'words', 'like', 'we"', 'and', 'you"', 'instead', 'of', 'the"', 'insured',"', 'the"', 'applicant',"', 'the"', 'society"', 'and', 'so', 'on;', 'conciseness;', 'clear,', 'helpful', 'headings', 'with', 'consistent', 'and', 'suitable', 'ways', 'of', 'making', 'them', 'stand', 'out', 'from', 'the', 'text;', 'a', 'good', 'typesize', 'and', 'clear', 'typeface;', 'a', 'reasonably', 'short', 'average', 'line', 'length;', 'and', 'plenty', 'of', 'answer', 'space', 'and', 'a', 'logical', 'flow', '(on', 'forms).', 'how', 'to', 'apply', 'for', 'the', 'crystal', 'mark', 'simply', 'send', 'us', 'a', 'copy', 'of', 'the', 'document.', 'you', 'can', 'either', 'post', 'it', 'to', 'us', '(po', 'box', '3,', 'new', 'mills,', 'high', 'peak,', 'sk22', '4qp).', 'send', 'it', 'by', 'fax', '(01663', '747038)', 'or', 'send', 'it', 'by', 'e-mail', '(info@plainenglish.co.uk).', 'please', 'let', 'us', 'know', 'before', 'sending', 'very', 'large', 'files', 'by', 'e-mail.', 'we', 'will', 'then', 'test', 'a', 'sample', 'of',
```

'your', 'document.', 'if', 'this', 'sample', 'meets', 'our', 'standard', 'straight', 'away,', 'we', 'will', 'then', 'test', 'the', 'whole', 'document.', 'we', 'can', 'sign', 'any', 'confidentiality', 'agreements', 'needed', '-', 'we', 'often', 'deal', 'with', 'documents', 'that', 'are', 'not', 'yet', 'available', 'to', 'the', 'public.', 'if', 'you', 'do', 'not', 'hear', 'back', 'from', 'us', 'within', '24', 'hours', '(or', 'three', 'days', 'if', 'you', 'have', 'sent', 'the', 'document', 'by', 'post),', 'please', 'let', 'us', 'know', 'straight', 'away.', 'if', 'either', 'the', 'sample', 'or', 'the', 'entire', 'document', 'does', 'not', 'meet', 'crystal', 'mark', 'standard,', 'we', 'will', 'give', 'you', 'an', 'estimate', 'for', 'editing', 'the', 'document.', 'it', 'is', 'impossible', 'to', 'give', 'an', 'estimate', 'before', 'we', 'see', 'the', 'document,', 'but', 'the', 'price', 'is', 'based', 'on', 'the', 'size', 'and', 'complexity', 'of', 'the', 'document.', 'the', 'costs', 'are', 'higher', 'if', 'you', 'want', 'us', 'to', 'use', 'our', 'accountant', 'or', 'barrister,', 'but', "don't", 'forget', 'the', 'price', 'is', 'reduced', 'for', 'corporate', 'members.', 'if', 'you', 'are', 'sending', 'a', 'microsoft', 'word', 'document,', 'we', 'may', 'be', 'able', 'to', 'send', 'you', 'the', 'file', 'back', 'in', 'electronic', 'form', 'with', 'our', 'changes', 'highlighted.', 'there', 'is', 'a', 'small', 'fee', 'for', 'this.', 'please', 'let', 'us', 'know', 'if', 'you', 'may', 'be', 'interested', 'in', 'this', 'option', 'so', 'that', 'we', 'can', 'include', 'this', 'in', 'your', 'estimate.', 'the', 'initial', 'testing', 'and', 'estimate', 'are', 'both', 'free.', 'you', 'do', 'not', 'have', 'to', 'use', 'our', 'editing', 'service', 'to', 'earn', 'the', 'crystal', 'mark,', 'but', 'the', 'final', 'document', 'must', 'pass', 'our', 'testing.', 'once', 'your', 'document', 'reaches', 'the', 'standard', '(whether', 'we', 'have', 'done', 'any', 'editing,', 'or', 'you', 'do', 'it', 'yourself),', 'you', 'can', 'use', 'the', 'crystal', 'mark', 'for', 'a', 'one-off', 'fee', 'of', '\xa3500.', 'this', 'includes:', 'the', 'costs', 'of', 'the', 'full', 'testing;', 'an', 'individually-numbered', 'crystal', 'mark', 'logo', 'as', 'camera-ready', 'artwork', 'or', 'on', 'disk;', 'listing', 'on', 'our', 'crystal', 'mark', 'holders', 'page', 'and', 'a', 'link', 'to', 'your', 'site;', 'our', 'full', 'assistance', 'if', 'a', 'customer', 'questions', 'the', 'clarity', 'of', 'a', 'crystal', 'mark', 'document;', 'our', 'ability', 'to', 'act', 'as', 'an', 'expert', 'witness', 'if', 'your', "document's", 'clarity', 'is', 'ever', 'questioned', 'in', 'court;', 'the', 'public', 'recognition', 'that', 'you', 'have', 'achieved', 'the', 'toughest', 'standard', 'of', 'clarity', 'in', 'the', 'world!', 'the', '\xa3500', 'fee', 'does', 'not', 'apply', 'to', 'corporate', 'members.', 'they', 'can', 'have', 'a', 'free', 'crystal', 'mark', 'for', 'every', 'document', 'that', 'reaches', 'the', 'standard.', 'what', 'if', 'i', 'have', 'something', 'that', 'isn't", 'a', 'standard', 'document?', 'we', 'have', 'a', "'recommended", "by'", 'logo', 'for', 'products', 'such', 'as', 'magazines,', 'long', 'books', 'and', 'computer', 'software.', 'can', 'i', 'get', 'a', 'crystal', 'mark', 'for', 'an', 'entire', 'website?', 'we', 'have', 'a', 'special', 'scheme', 'for', 'this.', 'with', 'the', "'internet", 'crystal', "mark'", 'logo.', 'more', 'details', 'if', 'you', 'want', 'further', 'details', 'on', 'the', 'crystal', 'mark,', 'please', 'phone', 'us', 'on', '01663', '744409', 'or', 'e-mail', 'us.', 'please', 'include', 'a', 'mailing', 'address', 'and', 'ask', 'for', 'our', 'crystal', 'mark', 'brochure.', 'we', 'have', 'a', 'full', 'list', 'of', 'crystal', 'mark', 'holders.', 'you', 'may', 'also', 'be', 'interested', 'in', 'our', 'honesty', 'mark', 'scheme.', 'we', 'celebrated', 'all', 'crystal', 'mark', 'documents', 'on', '30', 'june', '2000', 'at', 'our', 'crystal', 'clear', 'day', 'ceremony.', ':', 'home', 'page', "what's", 'new', 'about', 'this', 'site', 'e-mail', 'us', 'sections', 'about', 'the', 'campaign', 'annual', 'awards', 'books', 'and', 'magazines', 'corporate', 'membership', 'crystal', 'mark', 'and', 'editing', 'examples', 'free', 'guides', 'issues', 'and', 'subjects', 'press', 'office', 'training', 'courses', 'contact', 'plain', 'english', 'campaign', 'po', 'box', '3', 'new', 'mills', 'high', 'peak', 'sk22', '4qp', 'phone', '01663', '744409', 'fax', '01663', '747038', 'info@plainenglish.co.uk', 'type', 'your', 'e-mail', 'address', 'and', 'click', 'the', "'join'", 'button', 'to', 'join', 'our', 'mailing', 'list:', 'powered', 'by:', 'messagebot']

sentence list = [\n\nplain english campaign: crystal mark and editing: questions about the crystal mark\n\n\n\n\n\n\n\n \n \n you are in the crystal mark and \n editing section', ' \n \n \n \n \n how the crystal mark works\n \n \n ', "\n \n \n \n \n \n \n \n in 1990 we \n

introduced the public's own seal of approval - the crystal mark - to encourage organisations to communicate clearly with the public", the crystal mark has now become firmly established in the uk as the standard that all organisations aim for when they produce public information', the mark also appears on documents in other countries such as the usa, australia and south africa', " more than 1000 organisations know that only plain english campaign's crystal mark will be accepted by the public as a guarantee of a document's clarity", ' this is because we will not give the crystal mark to any document unless our testing shows it can be read, understood and acted upon by the intended audience', " what we look for things we look for include: a good average sentence length (about 15 to 20 words); plenty of 'active' verbs (instead of 'passive' ones); everyday english; words like 'we' and 'you' instead of 'the insured', 'the applicant', 'the society' and so on; conciseness; clear, helpful headings with consistent and suitable ways of making them stand out from the text; a good typesize and clear typeface; a reasonably short average line length; and plenty of answer space and a logical flow (on forms)", " how to apply for the crystal mark simply send us a copy of the document', ' you can either post it to us (po box 3, new mills, high peak, sk22 4qp), send it by fax (01663 747038) or send it by e-mail (info@plainenglish', 'co', 'uk)', ' please let us know before sending very large files by e-mail', " we will then test a sample of your document", ' if this sample meets our standard straight away, we will then test the whole document', ' we can sign any confidentiality agreements needed - we often deal with documents that are not yet available to the public', " if you do not hear back from us within 24 hours (or three days if you have sent the document by post), please let us know straight away', " if either the sample or the entire document does not meet crystal mark standard, we will give you an estimate for editing the document", ' it is impossible to give an estimate before we see the document, but the price is based on the size and complexity of the document', " the costs are higher if you want us to use our accountant or barrister, but don't forget the price is reduced for corporate members", " if you are sending a microsoft word document, we may be able to send you the file back in electronic form with our changes highlighted", ' there is a small fee for this', ' please let us know if you may be interested in this option so that we can include this in your estimate', " the initial testing and estimate are both free', ' you do not have to use our editing service to earn the crystal mark, but the final document must pass our testing', " once your document reaches the standard (whether we have done any editing, or you do it yourself), you can use the crystal mark for a one-off fee of £3500', " this includes: the costs of the full testing; an individually-numbered crystal mark logo as camera-ready artwork or on disk; listing on our crystal mark holders page and a link to your site; our full assistance if a customer questions the clarity of a crystal mark document; our ability to act as an expert witness if your document's clarity is ever questioned in court; the public recognition that you have achieved the toughest standard of clarity in the world! the £3500 fee does not apply to corporate members", ' they can have a free crystal mark for every document that reaches the standard', " what if i have something that isn't a standard document? we have a 'recommended by' logo for products such as magazines, long books and

Further more, if we can make an investigation with dyslexic participants to help us chose the accessible web pages, it is also unavoidable to occur bias. For dyslexia contains different kinds of impairment, if we don't specify participants before the investigation, it is probable to get different choices according to their own impairment. Therefore, the accessible pages in this project should be chosen specifically by dyslexic people with study difficulties who are more care about the readability of texts.

- Assumption 4 Insufficient training samples

When the classifier tries to classify a given test sample, first it grades the sample with test.py to obtain an observation of that sample, and then do search in the training dictionary to see the probability of good class and the probability of bad class for the particular observation. The rest of the work is to assign the sample to the class with higher probability. If not found in the training dictionary, the two probabilities will be assigned 0, and of course it will be fail to classify. To improve this situation, numerous valid training samples are needed in order to get a broad range of observations in the training dictionary. But due to the lack of accessible web pages for dyslexic people, it will be probable that we cannot find enough good training samples for the classifier. Therefore, much related work should be improved if we are willing to achieve higher accuracy in this project.

6 Further development

To obtain a satisfied accuracy, necessary improvements present as follows:

- Use some third party extension modules to make fragment.py behave better when converting html file into its plain form.
- Adopt more accurate algorithm when counting syllables in the chosen web page.
- Make an investigation with dyslexic participants, which might help distinguish web pages that are accessible for them or not. Try to get sufficient and more accurate training samples from such investigation for further use.

Further extensions to the project can be developed as follows:

- For world wide use
People with dyslexia are all over the world and speak different languages, however, the Flesch reading ease test used in this project is specifically designed for evaluating the readability of English text. Thus, classifier in this project can only process English web pages. Note that there are also such formulas for other languages, for Spanish we have Fernandez Huerta, for French we have Kandel & Moles and for Swedish & Danish we have LIX [2]. The language involving in this project is limited by the language capability of the designer. Further extensions on other languages can be developed based on this work.
- For more friendly interface
Develop Graphical User Interface (GUI) for this project, which might make use of wxpython [14] or other python GUI tools.

7 Conclusion

This project is to develop a classifier that can distinguish whether the chosen web page is accessible or inaccessible for people with dyslexia. The classifier is based on Naïve Bayes algorithm which is one of the most known and straightforward algorithms which applies to text classification. Classifier model can be induced from training set of data before test set of data used for testing the accuracy of the model.

In the project, Flesch reading ease test is adopted to give readability scores to the web pages which will be later used as observations in the classifier.

The accuracy of the classifier when testing with test data is unsatisfied. For there might be multi-reasons, we set assumptions to evaluate the results. The results might be much better if all the problems as we assumed can be improved in further development.

Appendices

A1 Glossary & Abbreviations

Short definitions of terms used and references to further relevant glossary sources

Term	Explanation	URL
cls	Class	
obs	Observation	
EIAO	European Internet Accessibility Observatory	http://www.eiao.net/
W3C	World Wide Web Consortium	http://w3.org/
<u>GUI</u>	Graphical User Interface	http://en.wikipedia.org/wiki/GUI

A2 References

- [1] A Dyslexic Perspective on e-content Accessibility
<http://www.techdis.ac.uk/seven/papers/>
Author: Peter Rainger (p.f.rainger@sussex.ac.uk)
Date of Publication: 25/03/03
- [2] Methods for measuring text readability:
<http://standards-schmandards.com/2005/measuring-text-readability>
- [3] A Perl module on categorization:
<http://aspn.activestate.com/ASPN/CodeDoc/AI-Categorize/Categorize.html>
- [4] Final report of previous work:
<http://www.eiao.net/webmining/previousprojects/reportgroup6.pdf> -
Yao Fei and Yang Kun, Document classification
- [5] Introduction on Flesch-Kincaid readability test:
<http://en.wikipedia.org/wiki/Flesch-Kincaid>
- [6] Assignment for this project:
<http://eiao.net/webmining/projects>
- [7] Introduction on document classification:
http://en.wikipedia.org/wiki/Document_classification
- [8] <data mining>, Margaret H.Dunham
- [9] Introduction on Python:
<http://www.python.org/doc/>
- [10] a Python module for counting syllables:
<http://viewvc.red-bean.com/gnoetics/src/syllables.py?revision=1&view=markup>
- [11] a Perl module for counting syllables:
<http://search.cpan.org/~gregfast/Lingua-EN-Syllable-0.251/Syllable.pm>

- [12] “The British Dyslexia Association – Indications for dyslexia”:
www.bdaweb.co.uk
- [13] “Find Law – for Legal Professionals”:
<http://www.findlaw.com/casecode/>
- [14] Python GUI tool:
<http://www.wxpython.org/>

A3 Code work in this project

fragment.py

```
import sys,re
def word_split(file):# Split the text into wordlist
    f = open(file)
    data = ".join(f.readlines())
    f.close()
    text = re.sub(r'\<([\^\<\>]*\>)',",",data)
    wordlist = text.lower().split()
    return wordlist

def sentence_split(file): # split the text into sentencelist
    f = open(file)
    data = ".join(f.readlines())
    f.close()
    text = re.sub(r'\<([\^\<\>]*\>)',",",data)
    sentencelist = text.lower().split('.')
    return sentencelist
```

word.py

```
import sys,fragment
def word_count(file): # Count all the words
    total_words = len(fragment.word_split(file))
    return total_words
```

sentence.py

```
import sys,fragment
def sentence_count(file): # Count all the sentences
    total_sentences = len(fragment.sentence_split(file))
    return total_sentences
```

syllable.py

```
import string,re,fragment
# This is based on the algorithm in Greg Fast's Pearl module
# Lingua::EN::Syllable.

sub_syllable = ["cial", "tia", "cius", "cious", "gui", "ion", "iou",
                "sia$", ".ely$"]
add_syllable = ["ia", "riet", "dien", "iu", "io", "ii",
                "[aeiouy]bl$", "mbl$",
                "[aeiou]{3}",
                "^mc", "ism$",
                "(.)(?!\\1)([aeiouy])\\2l$",
                "[^l]llien",
                "^coad.", "^coag.", "^coal.", "^coax.",
                "(.)(?!\\1)[gq]ua.(?!\\2)[aeiou]",
                "dnt$"]

# Compile regular expression patterns
for i in range(len(sub_syllable)):
    sub_syllable[i] = re.compile(sub_syllable[i])
```

```

for i in range(len(add_syllable)):
    add_syllable[i] = re.compile(add_syllable[i])

def syllable_count(file):
    syllable_count = 0

    # Add & subtract syllables
    for word in fragment.word_split(file):

        # Remove final silent 'e'
        if word[-1]=="e":
            word = word[:-1]

        # Add & subtract syllables
        for syllable in add_syllable:
            if syllable.search(word):
                syllable_count +=1
        for syllable in sub_syllable:
            if syllable.search(word):
                syllable_count -=1

        # Count vowel groups
        prev_was_vowel = 0
        for c in word:
            is_vowel = c in ("a", "e", "i", "o", "u", "y")
            if is_vowel and not prev_was_vowel:
                syllable_count +=1
            prev_vowel = is_vowel
        total_syllables = syllable_count
    return total_syllables

#if __name__ == "__main__":
#    print syllable_count(".join(open('file1.html','r').readlines()))

```

test.py

```

import word,sentence,syllable,math
def test(file): # Based on Flesch-Kincaid reading ease test
    a = word.word_count(file)
    b = sentence.sentence_count(file)
    c = syllable.syllable_count(file)

    score = 206.835 - 1.015*( float(a)/float(b))-84.6*(float(c)/float(a))
    score = int(score)

    if score < 0:
        score = 0
    if score > 206:
        score = 206
    return score

```

NB.py

```

from __future__ import division

```

```

import math,cmath,copy
class NaiveBayes:
    def __init__(self):
        self.prior = {} # Frequency of each class
        self.total = {} # Frequency of each (class,value)-tuple
        self.count = 0 # Number of observations
    def add(self, cls, obs):
        'Adds an observation to the classifier'
        self.prior[cls] = self.prior.get(cls, 0) + 1
        key = cls, obs
        self.total[key] = self.total.get(key, 0) + 1
        self.count += 1
    def discr(self,cls, obs):
        'Bayesian discriminant. Proportional to posterior probability'
        res = self.prior[cls]/self.count
        freq = self.total.get((cls,obs), 0)
        result =res* freq/self.prior[cls]
        self.total[(cls,obs)] = result
        return result

    def classify(self, obs):
        if self.total.has_key(('good',obs)):
            gp = self.total.get(('good',obs))
        else:
            gp=0
        if self.total.has_key(('bad',obs)):
            bp = self.total.get(('bad',obs))
        else:
            bp=0
        if gp>bp:
            print 'Accessible for dyslexic people'
        elif gp<bp:
            print 'Not accessible for dyslexic people'
        elif gp==bp:
            print 'fail to classify according to the training dictionary'

```

main.py

```

import NB
import test
if __name__=="__main__":

    #Training

    nb = NB.NaiveBayes()
    obs1 = [None]*10
    obs2 = [None]*10
    for i in range(1,11):

        filen='E:\python\Gooddic\yy'+str(i)+'.htm'
        obs1[i-1] = test.test(filen)
        nb.add('good',obs1[i-1])

```

```

filen2='E:\python\Baddic\y'+str(i)+'.htm'
obs2[i-1] = test.test(filen2)
nb.add('bad',obs2[i-1])
nb.total[('bad',obs1[i-1])]=nb.discr('bad',obs2[i-1])
nb.total[('good',obs1[i-1])]=nb.discr('good',obs1[i-1])

for i in range(1,11):
    nb.total[('bad',obs1[i-1])]=nb.discr('bad',obs2[i-1])
    nb.total[('good',obs1[i-1])]=nb.discr('good',obs1[i-1])

#Classification
a = raw_input('Enter the file you want to classify:')
testobs = test.test(a)
print nb.total
print testobs,nb.classify(testobs)
    
```

A4 Progress plan for the main project

ID	Task	Start	Finish	2006年09月		2006年10月				2006年11月			
				9-17	9-24	10-1	10-8	10-15	10-22	10-29	11-5	11-12	
1	Define the problem, Read related essays	2006-9-15	2006-9-21	█									
2	Learning related algorithms	2006-9-22	2006-9-26	█									
3	Decide the algorithms used	2006-9-26	2006-10-6	█									
4	Beginning learning Python	2006-9-22	2006-10-19	█									
5	Basic code implementation	2006-10-20	2006-10-31	█									
6	Combine new and existing metrics	2006-11-1	2006-11-6	█									
7	Training the classifier	2006-11-6	2006-11-10	█									
8	Iteration of testing and improving	2006-11-10	2006-11-17	█									
9	Write the final report	2006-11-6	2006-11-17	█									