

Web Document Classification

Ingrid Russell¹, Zdravko Markov, Todd Neller
June 3, 2005

1. Introduction

Along with search engines, topic directories are the most popular sites on the Web. Topic directories organize web pages in a hierarchical structure (taxonomy, ontology) according to their content. The purpose of this structuring is twofold: firstly, it helps web searches focus on the relevant collection of Web documents. The ultimate goal here is to organize the entire web into a directory, where each web page has its place in the hierarchy and thus can be easily identified and accessed. The Open Directory Project (dmoz.org) and About.com are some of the best-known projects in this area. Secondly, the topic directories can be used to classify web pages or associate them with known topics. This process is called tagging and can be used to extend the directories themselves. In fact, some well-known search portals as Google return with their responses the topic path of the response, if the response URL has been associated with some topic found in the Open Directory. Also, when one performs a search, in addition to a list of URLs one gets a ‘similar pages’ link. These similar pages could be generated from a system such as dmoz using the pages that the search returned and the dmoz structure. Dmoz could return, as similar pages to the documents returned, those in a subdirectory under a certain topic or subtopic.

Some of these topic directories are not very well developed yet. As the Open Directory is created manually, it cannot capture all URLs, therefore just a fraction of all responses are tagged. It would be good to be able to automatically classify pages and have the system identify where in the dmoz directory structure a page belongs or to be able to expand and create a new subdirectory/topic tree.

¹ Corresponding author: irussell@hartford.edu, Department of Computer Science, University of Hartford, West Hartford CT 06117.

2. Project overview

The aim of the project is to investigate the process of tagging web pages using the topic directory structures and apply Machine Learning techniques for automatic tagging. This would help in filtering out the responses of a search engine or ranking them according to their relevance to a topic specified by the user.

For example, a keyword search for “Machine Learning” in Google may return² along with the list of pages found (about 7,760,000) a topic directory path:

Category: Computers > Artificial Intelligence > Machine Learning

The first page found (David W. Aha: Machine Learning Page) belongs to this category. A search into the Google directories (the Open Directory) shows 210 pages belonging to the category Machine Learning. David Aha’s page is among them.

Another search however, “Tom Mitchell textbook”, returns the web page of the most popular ML book by Tom Mitchell, but does not return the topics path simply because this page is not listed in the Open Directory under Machine Learning.

Assuming that we know the general topic of the web page in question, say Artificial Intelligence, and this is a topic in the Open Directory, we can try to find the closest subtopic to the web page found. This is where Machine Learning comes into play. Using some text document classification techniques we can classify the new web page to one of the existing topics. By using the collection of pages available under each topic as examples, we can create category descriptions (e.g. classification rules, or conditional probabilities). Then using these descriptions we can classify new web pages. Another approach would be the nearest neighbor approach, where using some metric over text documents we find the closest document and assign its category to the new web page.

In this project, we will be working on a very small section of dmoz. Specifically, we plan to select 5 topics from dmoz and create a system that will automate adding web pages to that branch of dmoz. The plan is to use machine learning to implement a system that automates taking a web page and identifying which subtree of dmoz it belongs to. We will select about 100 web documents that we know where they belong in dmoz, train/teach the system to recognize how to classify these web documents, and then use it

² Note that this may not be what you see when you try this query. The web content is constantly changing as well as Google and other search engines’ approaches to search the web. This usually results in getting different results from the same search query at different times.

to categorize new web documents, i.e., identify which subdirectory of dmoz the new document should be added to.

3. Project description

The project is split into three major parts: data collection, feature extraction, and machine learning. These parts are also phases in the overall process of knowledge extraction from the web and classification of web documents (tagging). As this process is interactive and iterative in nature, the phases may be included in a loop structure that would allow each stage to be revisited so that some feedback from later stages can be used. The parts are well defined and can be developed separately (e.g. by different teams) and then put together as components in a semi-automated system or executed manually. Hereafter we describe the project phases in detail along with the deliverables that the students need to submit on completion of each stage.

Phase 1 consists of collecting a set of 100 web documents grouped by topic. These documents will serve as our training set. Phase 2 involves feature extraction and data preparation. During this phase the web documents will be represented by feature vectors, which in turn are used to form a training data set for the Machine Learning stage. Phase 3 is the machine learning phase. Machine learning algorithms are used to create models of the data sets. These models are used for two purposes. The accuracy of the initial topic structure is evaluated and secondly, new web documents are classified into existing topics.

3.1 Phase 1 -- Collecting sets of web documents grouped by topic

The purpose of this stage is to collect sets of web documents belonging to different topics (subject area). The basic idea is to use a topic directory structure. Such structures are available from dmoz.org (the Open Directory project), the yahoo directory (dir.yahoo.com), about.com and many other web sites that provide access to web pages grouped by topic or subject. These topic structures have to be examined in order to find several topics (e.g. 5), each of which is well represented by a set of documents (at least 10). For this project you are asked to use dmoz and to identify 20 web documents for each of the 5 topics. This will result in approximately 100 web documents. These

documents will be used as our training and learning set. It is not necessary that your topics are computer science related topics. As you select your topics, you may want to select topics that are of interest to you and ones that you are familiar with, such as mountain biking, or any area of interest to you. As you decide on topics, note that dmoz is still weak in some areas but mature and rich in others. You may want to avoid topics that dmoz is weak at.

Alternative approaches could be extracting web documents manually from the list of hits returned by a search engine using a general keyword search or collecting web pages from the web page structure of a large organization (e.g. university).

The outcome of this stage is a collection of several sets of web documents (actual files stored locally, not just URLs) representing different topics or subjects, where the following restrictions apply:

a) As these topics will be used for learning and classification experiments at later stages. they have to form a specific structure (part of the topic hierarchy). It is good to have topics at different levels of the topic hierarchy and with different distances between them (a distance between two topics can be defined as the number of predecessors to the first common parent in the hierarchy). An example of such structure is:

topic1 > topic2 > topic3
topic1 > topic2 > topic4
topic1 > topic5 > topic6
topic1 > topic7 > topic8
topic1 > topic9

The set of topics here is {topic3, topic4, topic6, topic8, topic9}. Also, it would be interesting to find topics, which are subtopics of two different topics. An example of this is:

Top > ... > topic2 > topic4
Top > ... > topic5 > topic4

As you select your topics, it is important that this structure be used. An example of a topic structure from dmoz would be:

Topic 1 *Topic 2* *Topic 3*
Computers: Artificial Intelligence: **Machine Learning**

Topic 1 *Topic 2* *Topic 4*
Computers: Artificial Intelligence: **Agents**

Topic 1 *Topic 5* *Topic 6*
Computers: Algorithms: **Sorting and Searching**

Topic 1 *Topic 7* *Topic 8*
Computers: Multimedia: **MPEG**

Topic 1 *Topic 9*
Computers: **History**

Use the format of the sample list of documents for the above topics available at:
<http://uhaweb.hartford.edu/compsci/ccli/FinalVersion/DocClassification/Resources/Documents.doc>

Note how each link has a reference number in the left column.

- b) There must be at least 5 different topics with at least 20 documents in each.
- c) Each document should contain a certain minimum amount of text. This may be measured with the number of words (excluding articles and punctuation marks). For example, this minimum could be 200 words.
- d) Each document should be in HTML format and contain HTML tags as title, headings or font modifiers.
- e) Avoid documents that are mostly links to other documents.
- f) Each page must have enough keywords that represent the topic.

In the later phase, we will be extracting keywords from each web document that identifies the topic. These keywords will need to represent the web document well. The selection of these documents is very important as we will be training our system with them. If we do not use ‘good’ training data, our system will not work as well. It is thus important that the documents you select now represent the topic very well. As you compile these pages, you need to keep in mind the following:

- The documents you select should have at least 5 keywords that identify the topic.
- As mentioned above, each document should have around 200 words. However, it is acceptable to select a document with fewer words if the document has a high concentration of keywords.
- You have to stop at the first level web document, i.e., you cannot navigate more.
- You will run into web documents that have a significant amount of graphics and not enough text to classify them. You need to avoid such documents. However, you will see that some of these documents have an ‘introduction’ type link that fully describes the topic. In this case, you may select the web document pointed to by such a link to represent that topic.
- Each two pages under the same topic must share 5-10 keywords.

The topics at the non-leaf nodes of the structure (1, 2, 5 and 7) stand for topics represented by all pages falling into their successor nodes. There is no need to collect pages for them because all actual pages fall into the leaves. The higher level nodes are composite topics that conceptually include lower level ones. In fact these composite topics appear at the learning stage when the pages from their subtopics are supplied as examples.

It is important to note, however, that to make learning such composite topics possible, there must be some similarity between their constituent topics. This must be taken into account when collecting pages. For example, topic3 and topic 4 must be similar (i.e. share keywords), but they both must be different from other topics such as 6 and 8.

3.1.1 Phase 1 Deliverable

A document listing all web documents by topic in the same format as the sample example at:

<http://uhaweb.hartford.edu/compsci/ccli/FinalVersion/DocClassification/Resources/documents.doc>

3.2 Phase 2: Feature Extraction and Data Preparation

You should now have around 20 web documents for each of the 5 topics for a total of approximately 100 documents. During this phase the web documents will be represented by feature vectors, which in turn are used to form a training data set for the Machine Learning stage. The basic steps to achieve this follow. An alternative to this approach, a Prolog based approach to feature extraction and data preparation, is included in Appendix A.

3.2.1 Step 1: Keyword Selection

Select a number of terms (words) whose presence or absence in each document can be used to describe the document topic. This can be done manually by using some domain expertise for each topic or automatically by using a statistical text processing system which is what we will be using.

Use a text corpus analysis package that filters and extracts keywords with their frequency counts. An example of such a system is TextSTAT, freeware software available from <http://www.niederlandistik.fu-berlin.de/textstat/software-en.html>. Other such systems are also available as freeware from <http://www.textanalysis.info/>. The process involves entering each document into TextSTAT and sorting in ascending order all words appearing in each document by their frequency. The goal is to collect 100 keywords that represent the documents.

While TextSTAT can take URLs as input, TextSTAT is not able to differentiate between html formatting and other text and for some pages would return unusable site related text such as navigation bar text. As a result, each web document should be copied and pasted into a txt file. The names of the files being associated with the web document

reference numbers. Files corresponding to a topic should be included in a folder whose name is associated with that topic. You should have 5 folders with 20 documents each. For each of the 20 documents, use TextSTAT to generate a file with word frequency list. You will have 20 documents for each topic, for a total of 100 files. All word frequency files should be exported from TextSTAT as CSV files. Again, use the same file and directory naming format as above. We will refer to this as Data Set I.

The next step involves generating the keyword list and the ARFF file. While you may do this process manually or write your own program to automate the process, the steps below describe the process using a program that has already been created for you to automate the process. Import all 20 word frequency CSV files in each of the 5 folders into excel as one CSV file with topics divided by tabs. A template *ByPageTemplate.xls* with integrated VBA applications is provided to automate the process at <http://uhaweb.hartford.edu/compsci/ccli/FinalVersion/DocClassification/Resources/ByPageTemplate.xls>.

You may simply copy and paste from each of the 20 CSV files the words and their frequencies and put them in the corresponding columns of *ByPageTemplate.xls* under the appropriate web document number.

A toolbar provides several menu options that will be useful as you work on the next steps of generating the keywords including a script to gray out all instances of these commonly used words in the frequency list. This filtering eliminates noise and words that do not represent the topics. A demo of this process is available at <http://uhaweb.hartford.edu/compsci/ccli/FinalVersion/DocClassification/Resources/Demo.xls>

An option to generate the ARFF file is also included in the toolbar. This will be needed in a later step.

A list of the 1000 most common words is already imported into *ByPageTemplate.xls*. Use the options on the toolbar menu to help you generate the list of keywords. Your next step is to look through *ByPageTemplate.xls* and select a total of approximately 100 keywords that represent all 5 topics. You will need to select approximately 20 keywords per topic. Keywords need to be representing the documents well. It is important that each two documents under the same topic have 5-10 common

keywords. This process will be done manually and using various strategies including your knowledge about the domain. Once you finalize your list of keywords, include the list in the designated column in *ByPageTemplate.xls*. The list of keywords will help you in the next step as you generate the ARFF file which will serve as input in the machine learning phase.

3.2.1.1 Step 1 Deliverable

1. A document listing all 20 keywords for each of the 5 topics.
2. A combined list of all 100 keywords.
3. A description of the strategies used for the selection of these keywords.

3.2.2 Step 2: Feature Extraction

During this phase, you will be creating the vector space model, a 100x100 matrix, which serves as a representation of the 100 web documents. These documents will serve as our training and learning set in the machine learning phase.

Using the selected 100 keywords as features (attributes), you will be creating a feature vector (tuple) for each document with Boolean values corresponding to each attribute (1 if the keyword is in the document, 0 – if it's not). We have 100 keywords which are attributes used to represent the documents. You will end up with 100 feature vectors, with 100 elements each. This is the vector space model of the documents. It is a representation of the 100 documents. There are other ways to get better representation of these documents, but for now we will use this Boolean representation.

A more sophisticated approach for determining the attributes values can be used too. It is based on using the term frequencies scaled in some way to normalize the document length. Further, the HTML tags may be used to modify the attribute values of the terms appearing with the scope of some tags (for example. increase the values for titles, headings and emphasized terms).

Ideally, one would want to prepare several files by using different approaches to feature extraction. For example, one with Boolean attributes, one with numeric based on text only, and one with numeric using the html information. Versions of the data sets with a different number of attributes can also be prepared. A rule of thumb here is that the

number of attributes should be less than the number of examples. The idea of preparing all those data sets is twofold. By experimenting with different data sets and different machine learning algorithms the best classification model can be found. By evaluating all those models, students will understand the importance of various parameters of the input data for the quality of learning and classification.

3.2.2.1 Step 2 Deliverable

1. A paragraph describing what a feature vector is and another paragraph describing the vector space model.
2. For each topic, select one web document and create the corresponding feature vector.
3. A copy of all 100 keywords, the selected web document, and the resulting 5 feature vectors.

3.2.3 Step 3: Data Preparation

Next, you will need to create a data set in the ARFF format to be used by the Weka 3 Data Mining System which we will be using in the next phase. The input data to Weka should be in Attribute-Relation File Format (ARFF) format. An ARFF file is a text file, which defines the attribute types and lists all document feature vectors along with their class value (the document topic).

In the next phase and once we load the ARFF formatted files into Weka, we will be using several learning algorithms implemented in Weka to create models of our data and to test these models and decide which is the best model to use.

Weka 3 Data Mining System is a free Machine Learning software package in Java available from <http://www.cs.waikato.ac.nz/~ml/weka/index.html>. Install the Weka package using the information provided in the Weka software page and familiarize yourself with its functionality. A readme file for installing and using Weka 3 is available at <http://uhaweb.hartford.edu/compsci/ccli/FinalVersion/DocClassification/Resources/Weka/Readme.txt>

Weka 3 tips and tricks are available at:

http://www.cs.waikato.ac.nz/~ml/weka/tips_and_tricks.html

This is one of the most popular ML systems used for educational purposes. It is the companion software package of the book titled Machine Learning and Data Mining [Witten and Frank, 2000]. Chapter 8 of Witten's book describes the command-line-based version of Weka. Chapter 8 is available at

<http://uhaweb.hartford.edu/compsci/ccli/FinalVersion/DocClassification/Resources/Chapter8.pdf>.

Read Section 1 of chapter 8. For the GUI version, read Weka's user guide at <http://uhaweb.hartford.edu/compsci/ccli/FinalVersion/DocClassification/Resources/ExplorerGuide.pdf>

An introduction to Weka is also available at: <http://www.oefai.at/~alexsee/WEKA/>

Once you have installed Weka and read section 8.1, run some experiments using the data sets provided with the package (e.g. the weather data).

The links below provide additional information on the ARFF format:

<http://www.cs.waikato.ac.nz/~ml/weka/arff.html> and

<http://www.cs.waikato.ac.nz/~ml/old/workbench/arff.html>

Steps (1) and (2) above are part of the so-called vector space model, which is well known in the area of Information Retrieval (IR). For more details, see [Chakrabarti, 2002], Chapter 3 or any text on IR.

The next step is to generate the ARFF file for all 100 documents. You may write your own program to generate the ARFF files or may generate the files manually.

Alternatively, you may use a program in ByPageTemplate that helps you automate the process by selecting an option on the toolbar that allows you to generate the ARFF file.

This ARFF file will serve as input to Weka in the machine learning phase. A demo of this process is available at

<http://uhaweb.hartford.edu/compsci/ccli/FinalVersion/DocClassification/Resources/Demo.xls>

3.2.3.1 Step 3 Deliverable

1. The ARFF data file containing the feature vectors for all web documents collected during Phase I.
2. A description of the ARFF data file including:
 - An explanation of the correspondence between the 100 keywords and the attribute declaration part of the ARFF file (the lines beginning with @attribute).
 - An explanation of the data rows (the portion after @data). For example, pick a tuple and explain what the 0's and 1's mean for the document that this tuple represents.

3.3 Phase 3: Machine Learning Phase

At this stage, Machine Learning algorithms are used to create models of the data sets. These models are then used for two purposes. The accuracy of the initial topic structure is evaluated and secondly, new web documents are classified into existing topics. For both purposes we use the Weka 3 Data Mining System. The steps involved are:

1. Preprocessing of the web document data: Load the ARFF files created at project stage 2, verify their consistency and get some statistics by using the preprocess panel. Screenshots from Weka are available at http://www.cs.waikato.ac.nz/~ml/weka/gui_explorer.html.
A sample Weka output with descriptions of various terms is available at <http://uhaweb.hartford.edu/compsci/ccli/FinalVersion/DocClassification/Resources/J48Output.doc>
2. Using the Weka's decision tree algorithm (J48) examine the decision tree generated from the data set. Which are the most important terms (the terms appearing on the top of the tree)? Check also the classification accuracy and the confusion matrix obtained with 10-fold cross validation and find out which topic is best represented by the decision tree.
3. Repeat the above steps using the Naïve Bayes algorithm and compare its classification accuracy and confusion matrices obtained with 10-fold cross

validation with the ones produced by the decision tree. Which ones are better?
Why?

4. New web document classifications: Collect web documents from the same subject areas (topics), but not belonging to the original set of documents prepared in project stage 1. Collect also documents from different topics.

Use a web crawler to collect these new documents. An example of a web crawler is available at [WebSPHINX: A Personal, Customizable Web Crawler](http://www-2.cs.cmu.edu/~rcm/websphinx/websphinx.jar). Download it and try it (simply click on the jar file link <http://www-2.cs.cmu.edu/~rcm/websphinx/websphinx.jar> or see the explanations in the web page). Experiment with varying the following parameters: Crawl the subtree/the server/the Web, Depth/Breadth first, use different limits (number of threads, page size, timeout). See how the dynamics of crawling changes by inspecting the web page graph. Once you collected the new documents, apply feature extraction and create an ARFF test file with one data row for each document. Then using the Weka test set option classify the new documents. Compare their original topic with the one predicted by Weka. For the classification experiments use the guidelines provided in <http://uhaweb.hartford.edu/compsci/ccli/FinalVersion/DocClassification/Resources/DMEEx.doc>

3.3.1 Phase 3 Deliverable

1. Explain the web crawler algorithm in terms of search by answering the following questions:
 - The Web is (1) tree, (2) directed acyclic graph, (3) directed graph, (4) graph, where:
 - the nodes are represented by ...
 - the edges are represented by ...
 - Which search algorithms are used by Web Crawlers and why?
 - Can a crawler go in a loop?

- How does the choice of the part of the web to be crawled (subtree/the server/the Web) affect the search algorithm?
 - How is multi-threading used to improve the efficiency of crawling algorithms?
 - What happens when page size or timeout limits are reached?
2. Explain the decision tree learning algorithm (Weka's J48) in terms of state space search by answering the following questions:
- What is the initial state (decision tree)?
 - How are the state transitions implemented?
 - What is the final state?
 - Which search algorithm (uninformed or informed, depth/breadth/best-first etc.) is used?
 - What is the evaluation function?
 - What does tree pruning mean with respect to the search?
3. This stage of the project requires writing a report on the experiments performed. The report should include detailed description of the experiments (input data, Weka outputs), and answers to the questions above. Weka does not classify web documents. Instead, Weka prints classification accuracy for the test set (a new web document), which is simply a number (percentage). This number must be used to explain how this new document is classified. The report should also include such interpretation and analysis of the results with respect to the original problem stated in the project.
4. Looking back at the process, describe what changes in the process you think could improve on the classification.

4. Extra Credit Possibilities

1. Repeat Step 3 of Phase 2 using the Nearest Neighbor (IBk) algorithm. Compare their classification accuracy and confusion matrices obtained with 10-fold cross

validation with the ones produced by the decision tree. Which of the three models explored here are better? Why?

2. Write your own web crawler to fetch a web document to be classified by the system. An algorithm for this is available in “Mining the Web” book listed below. You may restrict your collection to URL's only and to page titles to be able to do some page content analysis. You should introduce parameters to control the search. For example, depth-first, breadth-first with some parameters to bound the search as depth or breath limits, number of pages to retrieve, time-out time for each page or for the whole run, size limits for the pages etc.
3. Customize or add new and significant features to WebSPHINX. You should discuss with me the new features before you start working on this.

References and Readings

[Chakrabarti, 2002] Soumen Chakrabarti, Mining the Web - Discovering Knowledge from Hypertext Data, Morgan Kaufmann Publishers, 2002.

[Mitchell, 97] Mitchell, T.M. Machine Learning, McGraw Hill, New York, 1997.

[Witten and Frank, 2000] Ian H. Witten and Eibe Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, 2000.

Web->KB project: <http://www-2.cs.cmu.edu/~webkb/>

Appendix A

Prolog Based Approach to Feature Extraction and Data Preparation (Section 3.2)

3.2 Phase 2: Feature Extraction and Data Preparation

During this phase the web documents will be represented by feature vectors, which in turn are used to form a training data set for the Machine Learning stage. We provide a Prolog program that can do all the steps in this process and generate a data file to be used by the Weka ML system. The following components are needed for performing this:

- [SWI-Prolog](http://www.swi-prolog.org/). Use the stable versions and the self-installing executable for Windows 95/98/ME/NT/2000/XP. Available at <http://www.swi-prolog.org/>
- [Quick Introduction to Prolog](http://www.cs.ccsu.edu/~markov/ccsu_courses/prolog.txt) available at http://www.cs.ccsu.edu/~markov/ccsu_courses/prolog.txt
- Other Prolog Tutorials (optional)
 - [A Prolog Tutorial by J.R. Fisher](http://www.csupomona.edu/~jrfisher/www/prolog_tutorial/contents.html)
(http://www.csupomona.edu/~jrfisher/www/prolog_tutorial/contents.html)
 - More tutorials: <http://www.swi-prolog.org/www.html>
- Prolog program [textmine.pl](http://www.cs.ccsu.edu/~markov/ccsu_courses/mlprograms/textmine.pl) available at http://www.cs.ccsu.edu/~markov/ccsu_courses/mlprograms/textmine.pl
- A data set [webdata.zip](http://www.cs.ccsu.edu/~markov/ccsu_courses/mlprograms/webdata.zip) used to illustrate the use of [textmine.pl](http://www.cs.ccsu.edu/~markov/ccsu_courses/mlprograms/textmine.pl) available at http://www.cs.ccsu.edu/~markov/ccsu_courses/mlprograms/webdata.zip

The basic steps to achieve this follow. We use the data sample provided in the [webdata.zip](http://www.cs.ccsu.edu/~markov/ccsu_courses/mlprograms/webdata.zip) file as an illustrative example of this process.

3.2.1 Step 1: Keyword Selection

The [webdata.zip](http://www.cs.ccsu.edu/~markov/ccsu_courses/mlprograms/webdata.zip) archive contains 20 text files generated from 20 web pages collected from the web site of the CCSU school of Art and Sciences. For convenience we put the file names in a list, and the list in a file called `files.pl` (also available from the archive). The contents of `files.pl` is the following:

```
files([ 'Anthropology.txt',
        'Art.txt',
        'Biology.txt',
        'Chemistry.txt',
        'Communication.txt',
        'Computer.txt',
        'Justice.txt',
        'Economics.txt',
        'English.txt',
        'Geography.txt',
        'History.txt',
        'Math.txt',
        'Languages.txt',
```

```

'Music.txt',
'Philosophy.txt',
'Physics.txt',
'Political.txt',
'Psychology.txt',
'Sociology.txt',
'Theatre.txt' ]]).

label( [
  art - [ 'Art.txt',
          'Justice.txt',
          'English.txt',
          'History.txt',
          'Languages.txt',
          'Music.txt',
          'Philosophy.txt',
          'Political.txt',
          'Theatre.txt' ],

  sci - ['Anthropology.txt',
         'Biology.txt',
         'Chemistry.txt',
         'Communication.txt',
         'Computer.txt',
         'Math.txt',
         'Physics.txt',
         'Geography.txt',
         'Economics.txt',
         'Psychology.txt',
         'Sociology.txt' ]
]).

```

The first list (files) is a catalog of all file names and the second one (label) groups the files (documents) in two classes (two sublists) – art and sci.

After installing and running SWI-Prolog we have to load textmine.pl and files.pl in the Prolog database with the following queries:

?- [files].

?- [textmine].

Then the following query generates a list of the 20 most frequent terms that appear in the corpus of all 20 documents. Note that the actual text files (listed in files) should be stored in the same folder where textmine.pl and files.pl are located.

?- files(F),tf(F,20,T),write(T).

[department, study, students, ba, website, location, programs, 832, phone, chair, program, science, hall, faculty, offers, music, courses, research, studies, sociology]

```
F = ['Anthropology.txt', 'Art.txt', 'Biology.txt', 'Chemistry.txt', 'Communication.txt',
'Computer.txt', 'Justice.txt', 'Economics.txt', 'English.txt'|...]
T = [department, study, students, ba, website, location, programs, 832, phone|...]
```

Note that we use write(T) to print the whole list, because Prolog prints just the first 9 elements in its standard answer.

Then we may extend the query to generate the inverted document frequency list (IDF). First we have to generate a list of terms and then we pass them to the procedure that generates the IDF list. For example:

```
?- files(F),tf(F,50,T),idf(F,T,20,IDF),write(IDF).
[3.04452-music, 3.04452-sociology, 3.04452-anthropology, 3.04452-theatre, 3.04452-
criminal, 3.04452-justice, 3.04452-communication, 3.04452-chemistry, 2.35138-physics,
2.35138-political, 1.94591-history, 1.94591-sciences, 1.65823-american, 1.65823-social,
1.65823-international, 1.65823-public, 1.43508-computer, 1.43508-offered, 1.25276-ma,
1.25276-work]
```

```
F = ['Anthropology.txt', 'Art.txt', 'Biology.txt', 'Chemistry.txt', 'Communication.txt',
'Computer.txt', 'Justice.txt', 'Economics.txt', 'English.txt'|...]
T = [department, study, students, ba, website, location, programs, 832, phone|...]
IDF = [3.04452-music, 3.04452-sociology, 3.04452-anthropology, 3.04452-theatre,
3.04452-criminal, 3.04452-justice, 3.04452-communication, 3.04452-chemistry, ... -...|...]
```

Note that the IDF list is ordered by decreasing values of IDF (shown before each term). As the IDF value is usually big for rare terms, in the IDF list we have the least frequent 20 terms out of the 50 terms generated by tf(F,50,T).

3.2.2 Step 2: Feature Extraction

At this step we add the document labels and generate document vectors with the following query:

```
?- files(F),tf(F,50,T),idf(F,T,20,IDF),label(L),class(F,L,FL),vectors(FL,IDF,V),ppl(V).
```

ppl(V) will print here the vectors with numeric values (the output is skipped for brevity). We may also generate binary vectors by replacing vectors with binvectors with the following query:

```
?-
files(F),tf(F,50,T),idf(F,T,20,IDF),label(L),class(F,L,FL),binvectors(FL,IDF,V),ppl(V).
```

The above two queries just show the vectors and can be used to visually inspect the results of feature extraction. The idea is that the two parameters – the sizes of the TF and IDF lists (50 and 20) have to be adjusted so that the vectors do not have columns or rows with all the same value or all 0's.

3.2.3 Step 3: Data Preparation

After we get a good set of vectors from the previous step we may generate the ARFF data files for Weka just by adding the arff procedure at the end of the query (or replacing ppl with it, if we don't want to see the output):

```
?- files(F),tf(F,50,T),idf(F,T,20,IDF),label(L),class(F,L,FL),binvectors(FL,IDF,V),  
arff(IDF,V,'wekadata.arff').
```

This query generates binary vectors. By using vectors instead of binvectors we get numeric vectors (using the IDF values).

The file 'wekadata.arff' is in the proper format to be loaded in Weka and used for classification.

More information about using the textmine.pl program for feature extraction, classification and clustering is available in the following documents:

- http://www.cs.ccsu.edu/~markov/ccsu_courses/mlprograms/WebMiningLab1.txt
- http://www.cs.ccsu.edu/~markov/ccsu_courses/mlprograms/WebMiningLab2.txt
- http://www.cs.ccsu.edu/~markov/ccsu_courses/mlprograms/WebMiningLab3.txt
- http://www.cs.ccsu.edu/~markov/ccsu_courses/mlprograms/WebMiningLab4.txt