

Web User Profiling

Zdravko Markov¹, Ingrid Russell, Todd Neller
June 3, 2005

1. Introduction

The Web is the largest collection of electronically accessible documents, which make the richest source of information in the world. The problem with the Web is that this information is not well structured and organized so that it can be easily retrieved. Search engines help in accessing web documents by keywords, but this is still far from what we need in order to effectively use the knowledge available on the Web. Machine Learning and Data Mining approaches go further and try to extract knowledge from the raw data available on the Web by organizing web pages in well defined structures or by looking into patterns of activities of Web users. This project focuses on this challenge and explores the Machine Learning techniques suitable for this purpose.

Web searches provide large amounts of information about the web users. Data mining techniques can be used to analyze this information and create web user profiles. A key application of this approach is in marketing and offering personalized services, an area referred to as “data gold rush”.

2. Project Overview

The aim of this project is to develop a system that helps us develop an intelligent web browser. The project will focus on the use of Decision Tree learning to create models of web users. Students will be provided with Decision Tree learning tools and will collect data from web searches. They will then experiment with creating web user models and using these models for improving the efficiency of web searches performed by the same or new users. Students learn the basics of information retrieval and machine learning, gain experience in using recent software applications in these areas, and have a

¹ Corresponding author: markovZ@mail.ccsu.edu, Department of Computer Science, Central Connecticut State University, New Britain, CT 06050.

better understanding of fundamental AI concepts such as knowledge representation and search.

3. Project Description

This project is split into three major parts - data collection, feature extraction and machine learning (mining). At the data collection and feature extraction stages, web documents (pages) are collected and organized by users. Then they are represented as feature vectors and mapped into user categories. At the machine learning stage, decision tree learning algorithms are applied to the feature vectors in order to create models of the users these vectors (documents) are mapped onto. Then the models can be used to filter out web documents returned by searches so that the users can get more focused information from the search engines. In this way users can also be identified by their preferences and new users classified accordingly.

Phase 1 consists of identifying 5 users and collecting a set of 100 web pages (documents) from a given topic. The pages are then labeled by user preferences. These documents will serve as our training set. Phase 2 involves feature extraction and data preparation. During this phase the web pages 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 representing the respective users. These models are used for two purposes: improving the efficiency of web searches performed by the same user or identifying the category of new users.

3.1 Phase 1: Web Document Collection Grouped by User Preferences

The purpose of this stage is to collect a set of web documents labeled with user preferences. This can be done in the following way: The user performs web searches with simple keyword search or just browses web documents. To each web document, the user assigns a label representing whether or not the document is interesting to the user. For this project, you are asked to use a web crawler to collect the web pages.

An example of a web crawler is available at [WebSPHINX: A Personal, Customizable Web Crawler](#). 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.

Select a topic and identify five users. Using WebPSHINX, collect 100 pages related to this topic. For each user, you will need to identify which of the 100 web documents the user likes and which the user does not like. Text content as well as format of the page should be taken into account when users are identifying their views of a page as positive or negative.

3.1.1 Phase 1 Deliverable

1. The description of the process used to select the web pages and identify user likes and dislikes.
2. The list of all 100 web documents labeled by user likes and dislikes.
3. Explain the Web Crawler algorithm in terms of search. Answer 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?

3.2 Phase 2: Feature Extraction and Data Preparation

You should have 100 documents and should have identified user likes and dislikes for each of the 5 users. Having the set of labeled web documents, during this phase each document is used to create a feature vector (data tuple) labeled with the user preferences. These feature vectors form the training data set for the Machine Learning phase. ARFF files are then generated which form the input to the data mining system. The basic steps to achieve this follow.

3.2.1 Step 1: Keyword Selection

Select a number of terms (keywords) whose presence or absence in each document identifies the document. This can be done manually 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, certain formatting within some web pages may cause problems. As a result, use the 'view source' option and save the source file of each page into a text file. For each of the 100 documents, use TextSTAT to generate a file with word frequency list. All word frequency files should be exported from TextSTAT as CSV files.

The next step involves generating the keyword list and the ARFF files. 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 100 word frequency CSV files into excel as one CSV file. You should have 5 such CSV files, one per user. Each CSV file should have two tabs, one for the pages the user identified as likes and another tab for the pages the user does not like. A template *ByPageTemplate.xls* with integrated VBA applications is provided to automate the process at

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

Note that you will need to use 5 such Excel files with only 2 tabs each. You may ignore the remaining tabs in the file. For each user, copy and paste from each of the 100 CSV files the words and their frequencies and put them in the corresponding columns of *ByPageTemplate.xls* under the appropriate web document number. The frequency of the pages that the user likes should go under the user likes tab and the rest under the user dislikes.

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/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, for each of the 5 excel files, select a total of approximately 100 most common keywords, approximately 50 from the likes and 50 from the dislikes. This process will be done manually and using various strategies. 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

4. A document listing all 100 keywords compiled for each of the 5 excel files that correspond to the 5 users.
5. A description of the strategies used for the selection of these keywords.

3.2.2 Step 2: Feature Extraction

You will need to complete this section for each of the 5 users, i.e., repeat the steps for each of the 5 excel files, i.e., for each user. For each of the labeled web documents, create a feature vector (data tuple) labeled with the user preferences. These vectors form the training and learning set in the machine learning phase. You will be creating what is referred to as the vector space model, a 100x100 matrix. The basic steps to create this data set are presented below.

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. The label to each feature vector is a 'yes' or 'no' based on whether or not the user likes that page. This is the vector space model of the 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 attribute 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 user, select one web document that the user likes and one that the user does not like and describe the corresponding feature vectors.
3. A copy of all 100 keywords, the selected web documents, and the resulting 100 feature vectors.

3.2.3 Step 3: Data Preparation

In this phase you will need to create data sets in the ARFF format to be used by the Weka Machine Learning system. 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 (for the Boolean values they will be nominal, and for the frequency-based ones - numeric) and lists all document feature vectors along with their class value. According to the ARFF format, the class is the last attribute in the sequence whose value appears as a last element in each data row. This value represents the document label assigned by the user (for example, interesting/ non-interesting, yes/no or like/dislike).

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/DocClassification/Resources/WekaReadme.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/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/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 training data files containing the feature vectors labeled with user preferences. Using the 100 documents create one single ARFF file without labels. Use the user preference information and create 5 different ARFF files by adding different labels (like, dislike) as a last value in each row (the class attribute value).

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 files. These ARFF files 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/DocClassification/Resources/Demo.xls>

3.2.3.1 Step 3 Deliverable

1. The 5 ARFF training data files containing the feature vectors labeled with user preferences 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 and what the last value in the row means for the corresponding user.

3.3 Phase 3: Machine Learning Stage

At this stage decision tree machine learning algorithms are used to create models of the data sets/users. Using these models users can be identified by their preferences and web documents from new searches classified as interesting/non-interesting for each particular user. The learning step uses the Weka Data Mining System – a free Machine Learning software package in Java available at <http://www.cs.waikato.ac.nz/~ml/weka/index.html>. This is one of the most popular ML systems used for educational purposes. The steps involved during this phase 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/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) for each user? 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 **Nearest Neighbor** (IBk) 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. Web document filtering (focusing the search). Collect a number of web documents returned by a search. Apply feature extraction and create an ARFF test file with one data row for each document. Then using the training set for a particular user and the test set option classify the new documents. Each one will get a corresponding label (interesting/non-interesting). Then simply discard the non-interesting documents and present the interesting ones to the user. Further, this step can be incorporated into a web browser, so that it automatically labels all web pages as interesting/non-interesting according to the user preferences. For the classification experiments use the guidelines provided in <http://uhaweb.hartford.edu/compsci/ccli/DocClassification/Resources/DMEx.doc>

3.3.1 Phase 3 Deliverable

1. 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?
2. 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 for the particular user - like or dislike. The report should also include such interpretation and analysis of the results with respect to the original problem stated in the project.

3. Looking back at the process, propose changes in the process that could improve on the classification.

4. Extra Credit

1. 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 which is on reserve in the library. 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 which is on reserve in the library. 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.
2. 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.

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 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