CSC 5240 – Introduction to Artificial Intelligence Fall 2021

Project Assignment: Web User Profiling

DUE: Tuesday November 30th (by class time)
SUBMISSION: ALL materials through the iLearn class-website dropbox
PRESENTATION: Tuesday November 30th (by class time)

(Special thanks to Ingrid Russell, Zdravko Markov, and Todd Neller!)

Overview

The Web is the largest collection of electronically accessible documents, making it the richest source of information in the world. However, one of the greatest problems of 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, extracting knowledge from the raw data available on the Web by organizing web pages in well defined structures or by observing patterns of Web user activity. This project focuses on this challenge and explores the applicable ML techniques.

Web searches provide large amounts of information about 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 the “data gold rush”.

Objectives

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. You will be provided with decision tree learning tools and will collect data from web searches. You will then experiment with creating web user models and use these models in order to improve the efficiency of web searches performed by the same or new users. The learning objectives of the project are to:

- Learn the basics of information retrieval and machine learning
- Gain experience with recent software applications in these areas, and
- Gain better understanding of fundamental AI concepts such as knowledge representation and search

Project

The project is split into three major phases - data collection, feature extraction and machine learning. At the data collection and feature extraction phases, web documents (pages) are collected and organized by user preferences (e.g. like/dislike). Then they are represented as feature vectors and labeled accordingly (like/dislike). At the machine learning stage, decision tree learning algorithms are applied to the feature vectors in order to create a user model reflecting the user preferences. Then the model can be used to filter out web documents returned by searches so that the user can get more focused information from the search engines. Also, models of different users can be created and then new users classified accordingly.

Phase 1 consists of collecting a set of 100 web pages from a given topic or part of the web graph. The pages are then labeled by the user preferences. 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. In Phase 3 machine learning algorithms are used to create a model of the data set representing the user preferences. This model is used for two purposes: improving the efficiency of web searches performed by the same user or identifying the category of new users.

Phase 1: Web document collection

The purpose of this stage is to collect a set of web documents labeled with user preferences. Web documents can be collected by web searches, browsing the web, or by using a web crawler. Once a document is retrieved, the user assigns a label representing whether or not the document is interesting to the user. We recommend using a web crawler as it provides additional insights into the web structure and organization.

A good example of a web crawler is WebSPHINX. It can be downloaded from https://www.cs.cmu.edu/~rcm/websphinx/. The jar archive (websphinx.jar) can be directly run (provided that the Java run time environment is available) or the Java sources may be compiled and run (see the explanations in the web page). Experiment with varying the crawler parameters <Crawl the subtree> / <the server> / <the Web> and <Depth first> / <Breadth first>, and using different limits (number of threads, page size, timeout). This will change the dynamics of crawling, which will show up in the visualization of the web page graph.

By running WebPSHINX with proper parameter settings, you will have to collect 100 web documents. Then determine which of the 100 web documents the user likes and which the user does not like. It's important to note that mainly the text content of the documents has to be taken into account when determining the user preference regarding each document as like or dislike. This is because the user model created at the machine learning phase will be based only on the document text content.

As a result of this phase, you will need to provide the following items:

- The description of the process used to collect the web documents and to identify the user preferences for each web page (likes or dislikes).
- A list of all 100 web documents labeled according to the user preferences.

In addition, you need to answer the following questions:

- Explain the Web crawler algorithm in terms of search.
- The Web is (1) a tree, (2) a directed acyclic graph, (3) a directed graph, or (4) a graph.
- The nodes are represented by ...
- The edges are represented by ...
- Which search algorithms are used by Web crawlers and why?
- Can a crawler go into 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?

Phase 2: Feature Extraction and Data Preparation

This phase is very similar to the one used in the Web document classification project. By using the Weka filters, Boolean or numeric values are calculated for each web document and the corresponding feature vector is created. Then vectors are included in the ARFF file for input to the Weka system. Finally the vectors are extended with class labels (e.g., like/dislike, interesting/non-interesting, or +/-) according to the user preferences. Hereafter we outline the basic steps that have to be taken at this phase.

Downloading and installing Weka

The Weka system is available at http://www.cs.waikato.ac.nz/~ml/weka/. It comes with documentation and helpful examples to familiarize yourself with its use. A suggested exercise for this step is experimenting with the weather data set (weather.arff and weather.nominal.arff), which is also a classical example for the ML part in the AI course.

Creating a string data file in ARFF format

The Weka ARFF data format is described in the book [4] as well as in the document http://www.cs.waikato.ac.nz/~ml/weka/arff.html. To accomplish this step, first create a concatenation of all text documents (text corpus) obtained from the data collection step and save them in a single text file, where each document is represented on a separate line in plain text format. For example, this can be done by loading all text files in MS Word and then saving the file in plain text format without line breaks. Other editors may be used for this purpose too. (You want to write a program to automate this process.)

Once the file with the text corpus is created each line in it (an individual document content) must be enclosed in quotation marks ("), a document name or ID has to be added in the beginning of the line, and the user preference for the document (e.g. +/-), which plays the role of the class label - at the end. Also, a file header is needed in the beginning of the file followed by @data as shown below:

```
@relation text_corpus_1_string

@attribute document_name string
@attribute document_content string
@attribute document_class {+,-}

@data

Doc1, "example text document …", +
…
```

This representation uses three attributes – document_name, document_content, both of type string, and document_class – of type nominal. Each row in the data section (after @data) represents one of the initial text documents. Note that the number of attributes and the order in which they are listed in the header should correspond to the comma separated items in the data section.

Creating Term counts, Boolean, and TFIDF data sets

The string data file is now loaded in Weka using the "Open file" button in "Preprocess" mode. After successful loading the system shows some statistics about the number of attributes (3) their type (string) and the number of instances (rows in the data section or documents).

At this point we may start transforming our documents into feature vectors. First, we choose the **StringToNominal** filter and apply it to the first attribute, document_name. Then we choose the **StringToWordVector** filter and apply it with outputWordCounts=true. We may also change the setting of onlyAlphabeticTokens and useStoplist to see how the results change. As Weka moves the class attribute at the second place, we need to move it back as a last attribute by using the **Copy** filter and the **Remove** button. The result of all these steps is a Weka data set that uses a term count representation for the documents.

Now we have a document-term matrix loaded in Weka. By pressing the "Edit" button we can see it in a tabular format, where we can also change its content or copy it to other applications (e.g. MS Excel). Once created in Weka the table can be stored in an ARFF file through the "Save" option. Weka can also show some interesting statistics about the attributes. The class distribution over the values of each attribute (including the document name) is shown at the bottom of the "Selected attribute" area. With "Visualize All" we can see the class distribution in all attributes. A good exercise here is to examine the diagrams (the color indicates the document class, i.e. the user preference) and find the terms most specific for each user preference (+ and -).

Similarly we can create the Boolean and TFIDF representation of the document collection. To obtain the Boolean representation we need to apply the NumericToBinary filter to the term count representation. It's interesting to see how the visualization diagrams change after this transformation. For the TFIDF representation, we use the original string representation and apply the StringToWordVector filter with IDFTransform=true. Another good exercise here is to examine the document-term table and the bar diagrams and explain why some columns are filled with zeros only.

As a result of this phase, you are to provide the ARFF data file containing the feature vectors for all 100 web documents collected at Phase 1. It is recommended that you prepare all three types of data files – Boolean, term count, and TFIDF, and also provide some statistics (tables and diagrams) and analysis of the attributes and class distributions (see the suggested exercises above). Versions of the data sets with different numbers of attributes can be also prepared.

A suggested reading for this phase of the project is Chapter 1 of Data Mining the Web book [2]. This chapter discusses the basics of information retrieval and provides a set of exercises explaining the details of all steps needed to complete this phase of the project. Example datasets as well as Weka files for the string, Boolean, term count, and TFIDF representation formatted as ARFF files are available from the companion website of the book (http://www.dataminingconsultant.com/DMW.htm) and also from the authors' web page (http://www.cs.ccsu.edu/~markov/dmwdata.zip). Another good reading for this and later steps of the project is the excellent book by Witten and Frank [4], which discusses machine learning algorithms in general as well as their implementation and use through the Weka system.

Phase 3: Machine Learning Stage

At this phase, the approaches and experiments are similar to those described in the Web document classification project with an important difference in the last step where the machine learning models are used. This step can be called "web document filtering" (i.e. focusing the search) and can be summarized as follows: A number of web documents are collected using one of the approaches suggested in Phase 1. Feature extraction is applied, and an ARFF test file is created with one data row for each document. Using the training set prepared in phase 2 and the Weka test set option, new documents are classified so that each one maps to a corresponding label (+/-). Finally, the non-interesting documents (labeled -) are discarded and the interesting ones (labeled +) are presented to the user.

For this phase of the project, you are to do the following:

1. Preprocessing web document data: In this step, the ARFF files created at project phase 2 have to be verified for consistency and analyzed by using the preprocess mode.

2. Using the Weka's decision tree algorithm (J48), you are to examine the decision tree generated from the data set, and answer the following questions:

- Which are the most important terms (the terms appearing on the top of the tree) for determining the user preference?
- Check the classification accuracy and the confusion matrix obtained with 10-fold cross validation and explain the meaning of the precision and recall parameters that Weka reports
- 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. Run 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 or by browsing the web. Then they apply feature extraction and create an ARFF test file with one data row for each document. Using the training set and the test set option the new documents are classified. Each one will get a corresponding label (+ or -). Then the non-interesting documents (labeled -) and discarded and the interesting ones (labeled +) are presented 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, you may use the guidelines and tips provided in the document http://www.cs.ccsu.edu/~markov/ccsu_courses/DataMining-Ex2.doc.

5. Identifying users by their preferences. This step requires more than one different web users (I suggest at least 5) and creating a training data set for each one reflecting that user preferences. I suggest using the same set of 100 web documents collected at Phase 1, however with different labeling of each document (+/-) for each one of the 5 users. Thus after the data preparation phase (Phase 2) you will have 5 different data files, which then (during Phase 3) will be used to create 5 different user models. Once these data sets are available, you can have a new web user provide his/her preferences (using the same initial set of 100 web pages). Then after creating the corresponding data file, you can use it as a test set and combine it with each of the 5 training data sets. Running J48 or IBk with each one of the these 5 training data sets (models of existing users) and the same test set (the new user model) and examining the obtained classification accuracy will provide you with insights into how close the new user preferences are to the preferences of each one of the existing 5 users. This information can be then used to classify users and put them in groups according to their preferences.

Project deliverables are worth 90 points, and your presentation is worth 20 points, for a possible 110 points (i.e., 10 additional points).

Prerequisites and Requirements

To accomplish this project, you should have basic knowledge of algebra, discrete mathematics and statistics. Another prerequisite is the data structures course. While not necessary, experience with programming in Java would be helpful as the project uses Java-based packages. These packages are open source and you may want to use specific parts of their code to implement stand-alone applications.

The software packages and data sets used in the project are freely available on the Web:

- Weka 3 – Free open source Machine Learning and Data Mining software in Java available from http://www.cs.waikato.ac.nz/~ml/weka/index.html.
- Data sets for document classification accompanying the book *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage* ([2]) available from http://www.cs.ccsu.edu/~markov/dmwdata.zip.

It is recommended that before starting the project, you should read Chapter 19 of Russell and Norvig's book ([1]), and Chapters 1, 3, and 5 of Markov and Larose's book ([2]), or Chapters 3, 6, and 8 of Mitchell's book ([3]). While working on the project, you can use Witten and Frank's book [4] and the online documentation for the Weka 3 system available with the installation or from the website.

References

1. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach* (Third Edition), Prentice Hall, 2010. Chapters 18 and 20.
2. Zdravko Markov and Daniel T. Larose. *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage*, Wiley, 2007. Chapter 1 is available for at http://media.wiley.com/product_data/excerpt/56/04716665/0471666556.pdf.
3. Tom Mitchell. *Machine Learning*, McGraw Hill, 1997, Chapter 6.
4. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques* (Third Edition), Morgan Kaufmann, 2011.