Web Document Classification

Ingrid Russell[1], Zdravko Markov, Todd Neller

July, 2010

1. Introduction

Along with search engines, topic directories (a.k.a. web directories) are the most popular sites on the Web as they are usually provided to narrow searches. Topic directories organize web pages in a hierarchical structure (taxonomy, ontology) according to their content. The purpose of this structuring is twofold. First, 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) is one of the best-known projects in this area. Second, the topic directories can be used to classify web pages or associate them with known topics. This "tagging" process can be used to extend the directories themselves. In fact, well-known search engines such as Yahoo and Google may return with their responses the topic path, if the response URL has been associated with some topic found in a topic directory. As these topic directories are usually created manually they cannot capture all URL's, therefore just a fraction of all responses are tagged.

2. Project overview

The aim of the project is to investigate the process of tagging web pages using topic directory structures and apply ML techniques for automatic tagging or classifying web pages into topic categories. This would help filter search engine responses or rank them according to their relevance to a user-specified topic. For example, a Yahoo keyword search for "Machine Learning" may return topic directory paths along with the pages found:

Category: Artificial Intelligence > Machine Learning
Category: Artificial Intelligence > Web Directories
Category: Maryland > Baltimore > Johns Hopkins University > Courses

However, most of the pages returned are not tagged with directory topics. Assuming that we know the general topic of such an untagged web page, e.g., Artificial Intelligence, and this is a topic in a 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). Using these

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

descriptions, we can then classify new web pages. Another approach would be the similarity search approach, where we find the closest text document according to some metric, and assign its category to the new web page.

3. Project objectives

While working on this project students will learn the basics of information retrieval, data mining and machine learning, gain experience in using recent software applications, and most importantly have a better understanding of the role that fundamental AI concepts as knowledge representation and search play in these areas.

While reinforcing traditional AI core topics within a single, unified task of web document classification, the project allows the discussion of various issues related to machine learning, including:

- Basic concepts and techniques of machine learning
- Learning system implementation issues
- The role of learning in improved performance and in allowing a system to adapt based on previous experiences
- The important role data preparation and feature extraction play in machine learning
- The vector space model for representing web documents
- Feature extraction techniques with associated pros and cons for identifying and classifying documents
- The importance of model evaluation in machine learning and, in particular, the training and testing framework used to choose the best model for web page classification.

4. 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. 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. In Phase 3 machine learning algorithms are used to create models of the data sets. These models are used for two purposes. Firstly, the accuracy of the initial topic structure is evaluated and secondly, new web documents are classified into existing topics.

4.1. Phase 1: Collecting web documents grouped by topic

The purpose of this phase is to collect sets of web documents belonging to different topics (subject areas). The students begin by examining a topic directory structure. Such structures are available from dmoz.org (the Open Directory project), Google directory (directory.google.com), Yahoo! directory (dir.yahoo.com) and other web directories[2]. They first find several topics (e.g. 5), each of which is well represented by a sufficient number of documents (e.g. 20). Alternative approaches would be to extract web documents manually from a list of search engine hits, or collect web pages by using a Web Crawler (a good example is WebSPHINX available at http://www-2.cs.cmu.edu/~rcm/websphinx/) from the web page structure of a large organization (e.g. a university). The outcome of this phase is a collection of several sets of web documents representing different topics or subjects. Below are some guidelines that should be followed during this phase of the project:

a) As the topics will be used for learning and classification experiments at later phases they have to form a specific structure (part of the topic hierarchy). It's good to have topics at different levels of the topic hierarchy and with different distances between them (i.e. different depths in the hierarchy tree). An example of such structure is the following:

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

   The set of topics here is formed by the leaves of the tree which are topic3, topic4, topic6, topic8, and topic9. Also, it would be interesting to consider a more general graph structure where topics may have more than one parent, as this would make the classification task more difficult.

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, e.g. 200 words excluding stopwords and punctuation marks.

Our experiences with the project show that students easily understand this phase and prepare the initial data well without much guidance. This phase also provides an opportunity to connect to other CS and AI topics such as trees, graphs, semantic networks and ontologies. For example, an interesting question that students may be asked to

[2] E.g. http://en.wikipedia.org/wiki/List_of_web_directories

investigate is the possibility of cycles in the topic hierarchy. This question is related to the generality/specificity of topics and may be also considered in the context of set-subset relationships.

4.2. Phase 2: Feature extraction and data preparation

In this phase, web documents are represented by feature vectors, which in turn are used to form a training data set for the machine learning phase. This conversion actually illustrates the basic concepts of the vector space document model that plays an important role in information retrieval and web search. Various approaches and tools may be used to achieve the goals of this phase in the project. However, we recommend using the Weka machine learning system [4] in this and later phases. Although it is a large software suite, it is easy to install and use with an intuitive graphical interface. Learning to use the whole functionality of the system might overwhelm students in a one-semester course. However, for the purposes of this project only a small subset of functions is needed. Students may easily learn these functions on their own using online material, but in this phase we would especially recommend faculty assistance. Our experience shows that this may be done in a one or two hour lab that may also be used as a demo session for the ML part of the AI class.

Hereafter we outline the basic steps that have to be taken at this phase.

4.2.1. 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 students 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.

4.2.2. 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 students have first to 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. Students with programming experience may 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 document topic (class) - at the end, all separated by commas. 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 string

@data

Doc1, "example text document …", topic3
…
```

This representation uses three attributes – document_name, document_content, and document_class, all of type string. 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. An example of such string data file is "Departments-string.arff", available from the data repository at http://www.cs.ccsu.edu/~markov/dmwdata.zip, folder "Weka data".

4.2.3. 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 start transforming our documents into feature vectors. First, we choose the **StringToNominal** filter and apply it (one at a time) to the first attribute, document_name and then to the last attribute (index 3), document_class. 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 last 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. An example of this data set is the file "Departments-counts.arff", available from the data repository at http://www.cs.ccsu.edu/~markov/dmwdata.zip, folder "Weka data".

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. If we change the class to document_name we can see the distribution of terms over documents as bar diagrams.

A good exercise here is to examine the diagrams (the color indicates the document) and find the most specific terms for each document.

Similarly we can create the boolean and TFIDF representation of the document collection. Examples of these representations are provided in the files Departments-binary.arff and Departments-TFIDF.arff, available from the data repository at http://www.cs.ccsu.edu/~markov/dmwdata.zip, folder "Weka data". The comment in the beginning of each file explains the steps taken to create it.

To obtain the Boolean representation we need to apply the NumericToBinary filter to the term count representation. It's interesting to see how the diagrams change. 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, students have to provide ARFF data files containing the feature vectors for all web documents collected at Phase 1. We recommend that students prepare all three 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.

To complete this phase, in addition to Weka, other software tools for text processing are also needed, especially at step 2 in creating the string file with the text corpus. Students are usually provided with suggested approaches and tools for this purpose (for example, using MS Word to concatenate documents and convert them into plan text format). They are also given a choice to use any other software they may find suitable for the task. Our experience with this project shows that students at this level generally have the required background and skills to find and use the software they need. (Some even find this step interesting and challenging and write their own programs for text processing.) However, concerning Weka, we believe more faculty guidance is needed. Students find the Weka documentation insufficient and often ask for help. As we mentioned above, an hour or two hands-on lab usually suffices to fill this gap and to allow students to use Weka successfully in other phases of the project and for other Weka-based projects beyond.

4.3. Phase 3: Machine Learning

At this phase, machine learning algorithms are used to create models of the data sets. These models are then used for two purposes: (1) the accuracy of the initial topic structure is evaluated, and (2) new web documents are classified into existing topics. The Weka system is used for both purposes. The ML phase consists of the following steps:

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. Creating decision tree models with Weka's decision tree algorithm (J48): Good questions to ask students at this step are: Which are the most important terms for each data set (the terms appearing on the top of the tree)? How do they change with changing the data set? Also, students have to check the classification accuracy and the confusion matrix obtained with 10-fold cross-validation in order to find out which topic is best represented by the decision tree.
3. Applying the Naïve Bayes and Nearest Neighbor (IBk) algorithms: Students compare the classification accuracy and confusion matrices obtained with 10-fold cross-validation from these algorithms with the ones produced by the decision tree in order to evaluate which is superior and provide an explanation.
4. Experimenting with clustering: Three basic clustering algorithms, k-means, EM, and Cobweb, are applied to all data sets where the class attribute (document topic) is ignored. The resulting clusterings are then analyzed and compared with the original set of topics, or with the topic hierarchy when using Cobweb. Weka's classes to clusters evaluation method may be also used for this purpose. A data mining lab that provides useful information and tips about this step is available online.[3]
5. Classifying new web documents: Students find web documents from the same subject areas (topics), but not belonging to the original set of documents prepared in project phase 1. Documents from different topics are also needed. Then, feature extraction is applied and ARFF files are created for each document. Using the Weka test set option the new documents are classified and their original topic is compared with the one predicted by Weka. Additional online material provides guidelines for this step.[4]

For this final phase of the project, students may be asked to write a report on the experiments performed, including detailed descriptions of all experiments (input data, Weka outputs), answers to the questions, and interpretation and analysis of the results with respect to the original problem stated in the project, web document classification.

[3] http://www.cs.ccsu.edu/~markov/ccsu_courses/DataMining-Ex3.html
[4] http://www.cs.ccsu.edu/~markov/ccsu_courses/DataMining-Ex2.doc

5. Prerequisites and requirements

To accomplish this project students 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 students may want to use specific parts of their code to implement stand-alone applications.

The project is customizable and can accommodate different teaching approaches and different implementations depending on the choice of particular problems to be solved and tools to be used. The data collection step can be implemented manually or by using some software tools. The machine learning step uses implementations of Decision Trees, Naïve Bayes, and Nearest Neighbor algorithms available from free open source software packages. This allows the project to be extended to building stand-alone applications depending on the particular teaching goals and student experience in programming.

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 students read Chapters 18 and 20 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 students 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.

6. References

1. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach* (Second Edition), Prentice Hall, 2003. Chapters 13, 14.
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* (Second Edition), Morgan Kaufmann, 2005.