**Web Page Preprocessing, Learning and Classification**

Avinesh Mohanaranjan: 216 325 854

Dawit Gulta: 219 251 156

Javier Guillen: 214 933 519

Richard Nguyen: 215 751 324

Vatsal Rajyaguru: 216 977 449

**York University**

**ITEC4305 M: Web Mining Assignment 1 Report**

**Dr. Jimmy Huang**

**February 10, 2023**

**Introduction**

Natural language processing (NLP) is the study of how to make natural human language understandable and usable to computer programs. Natural Language Toolkit (NLTK) is a Python package that can be used for NLP. A lot of the data that is used for analysis is unstructured and contains human-readable text. NLTK is an effective API to preprocess text data for further analysis with machine learning models.

The training data set given contains 468 Web pages, 232 of which are from a University of Texas department and 234 from a University of Washington department. The test data set given contains 206 Web pages from Cornell University. Each of the Web pages in the dataset belongs to one of the three classes: student, faculty, and course. The files are organized into a directory structure, one directory for each class. Each directory contains Web pages in plain text files.

The Internet is filled with words, and those words are nested within a hierarchy of HTML tags behind every webpage. Beautiful Soup is a Python library that can extract text from a webpage using those HTML tags. The webpage is automatically converted to Unicode. Beautifulsoup collaborates with a parser to navigate, search, and modify the parse tree. It converts a complex HTML document into a tree of Python objects. Once the text data is retrieved, it can be used as data for Natural Language Processing. This API was chosen because it helps preprocess and clean the data.

**Description of Web page classification system**

Our python code has two files, one for preprocessing the data and another to perform the classification on the dataset.

The preprocessing python file will list the directories of all the category datasets in both the train and test data set folder, it will then clean the data, it does this through of a series of methods, which are the following:

* Word extraction from the web pages
* Setting words to lowercase
* special character removal
* Tokenization of the words
* stop word removal
* word stemming

The newly preprocessed text is stored in a text file within the folder of their respective data set.

The web page dataset uses 3 different classifier algorithms, which are Naive Bayes, Decision tree and K-Nearest Neighbor. The Navies Bayes classifier algorithm was the most successful of the three, yielding more than 90% accuracy, while Decision Tree was more volatile with about 88% accuracy. KNN unfortunately yielded the lowest accuracy of the three, with a 73% accuracy. Naive Bayes is a class of probabilistic algorithms that computes the probability that any given data point will fall into one or more of a set of categories.

**Description of Implementation**

All Machine Learning algorithms require some level of data pre-processing. Methods for Natural language processing (NLP) can be used to remove text data that does not add information and text data that produces noise. This data needs to be removed because it can reduce the efficiency of the classification learning algorithm.

Stop words are like “a”, “an” and “in” which do not add meaning to text data and can be removed to focus on important words. In addition, converting all characters to lowercase eliminated noisy data. Words like School and school mean the same but can be represented as two different words if not converted to lowercase. Furthermore, tokenization was performed on the raw data. Dividing the text data into smaller units, we were able to locate word boundaries. Word boundaries are the ending point of a word and the beginning of the next word. This helped begin another method to preprocess the data which was stemming. Stemming is where words are reduced to their root by removing the unnecessary characters at the end that do not add significance to the word.

A directory array is created which stores six variables (for each directory). These variables are created to input directory paths for training and test datasets for the code to execute. The array is created so we can iterate through each directory to begin preprocessing. A counter is also initialized to keep track of the number of files processed. Beautifulsoup is used to extract the text from the web pages. All html tags are parsed into Python objects by retrieving the filename. HTML tags are removed. Preprocessing is done on the extracted words. Special characters are removed and stop words are initialized using if statements. Tokenization is executed to split entire text data into individual words. Stemming is initialized, PorterStemmer is used which is a pre-written stemmed class. The words are concatenated into variable text. We store the text in a new text file which is outputted into the directory as they originate.

For Navies Bayes to work, we first imported packages Count Vectorizer, accuracy\_score, and multinomialNB. Then imported the cleaned training and test data. Initialized variable categories to store the different classes. We converted the text data into numerical representations using count vectorizer. Count vectorizer means breaking down any text into a bag-of-words representation. Then we trained the model using MultinomialNB. Training the model will be the most important step in all classifier algorithms used because in training, we pass our prepared data to the machine learning model to observe patterns and evaluate predictions. Then we convert the text data into a numerical representation for the test data. The training dataset is the set the model learns from; the test dataset is used to check the accuracy of the model. Finally we perform the validation to obtain an accuracy score. We repeated the above steps for the other classifier models and also managed to obtain an accuracy score as well as the classification report.

**Learning a Classifier From The Training Data**

Through the use of the sklearn python packages we were able to import the Naive Bayes algorithm. The first step in learning the training data was to convert the data set into a numerical representation, we did this through the use of the count vectorizer method, which essentially gets the collection of text documents and makes a matrix of token counts with them. Using the new numerical dataset we are able to create a naive bayesian model, and use the training data on it. Once the train data is run through the model, we can use the test data to verify the accuracy of the model, as well as running some validation on it and giving us metrics on the model itself.

Weka is a free open source machine learning software developed by the University of Waikato, New Zealand. After the training data was uploaded, the data was automatically identified by the software and were classified according to the attributes. Weka TextDirectoryLoader detects the directory and uses the subdirectory names as class labels. In our project, it automatically detected the sub folders in the training data as class and the counts. There were 77 for ‘faculty’, 115 for ‘course’ and 274 for ‘student’. The model found there were about 7522 directory sizes. Sample of the output of the test data looks like the following:

Text, letter, email

Description automatically generated

**Analysis**

To train the model, we have used different classifiers. We used several classifier methods with the training set, cross-validation methods, and percentage split. Using our Naive Bayes classifier that was trained with the training data provided, the classifier yielded more than 90% accuracy on the data.

Based on the results we collected, the naive bayesian classifier algorithm had better results, this is due to the fact that naive bayes can learn fast with a very small dataset. Naive Bayes is a linear classifier, whereas K-NN is not; it is faster when applied to large amounts of data. K-nn is typically slower for large amounts of data because of the calculations required for each new step in the process. Naive Bayes is a very good algorithm as it treats all attributes/features as being independent of each other, this helps a lot in text classification.

**Conclusion**

We have used the three powerful algorithms to train the dataset. The models used to classify the dataset were tested by the test dataset. Each algorithm brought different outputs. The focus of this assignment was to obtain experience in preprocessing the raw data for web mining. We have used two methods to evaluate the accuracy of the models; these are Python and Weka. Our findings show that Bayes Naive gives a better accuracy than Decision Tree or K-nearest neighbour algorithms. It is also important to note that proper preprocessing is mandatory for an effective classification. During this study we observed, proper cleaning data and preprocessing are mandatory for a better result of the accuracy of the model.