University of Essex

Department of Mathematical Sciences

Assignment 1: Indexing for Web Search

Submitted as part of the requirements for:

CE706-7-SP-CO: Information Retrieval

Name: Divya Arora
Registration Number: 1901423

Name: Aritra Ganguly
Registration Number: 1906467

Supervisor: Alba Garcia Seco de Herrera

Date of submission (21st February 2020)

Introduction

Computers can easily understand structured form of data but it is difficult for them to deal with unstructured data [2]. Here comes information retrieval into picture where useful information is extracted from the data available in unstructured format [1]. Our task is to transform the data present on the websites in somewhat structured form in order to get relevant information from the HTML data given in each url [3].

1 Main Body

Six stages discussed below are Engineering a Complete System, HTML Parsing, Pre-processing: Sentence Splitting, Tokenization and Normalization, Part-of-Speech Tagging, Selecting Keywords and Stemming or Morphological Analysis.

1.1 *Engineering a Complete System*

For developing a system first we need to read the URL in order to access the web content on the given websites. For carrying out this task we have used the library urlib.request.urlib is specifically used to read url and with the help of urlopen in-built function we can easily access the protocols like HTTP, FTP and many more [4]. Requests also play a significant role as it gives us the permission to send HTTP/1.1 requests.In our code segment we have added form data as "from-encoding" and parameters as "get-param" from the requests library.While reading the URL it is very important to know how many links are associated with each URL [4].

1.2 *HTML Parsing*

We are having data originally in the form of URL so it is obvious that we are having HTML tags also with in the given data. It becomes necessary for us to handle the HTML tags properly in order use data for any further analysis. BeautifulSoup is the library used here for doing HTML parsing of the data. It is the library designed to do parsing and extracting data from HTML

as per our needs. It has the capability to transform input documents to unicode and output documents to UTF-8. With a continuation, we are also extracting meta data out of each URL. In simple terms meta data is data about data. It serves our purpose by refining the data more which in turn makes it easy for us to analyse the data and use it as per our requirements. For extracting meta data we have Beautiful library and attr keyword [5].

1.3 *Pre-processing: Sentence Splitting, Tokenisation and Normalization*

Pre-processing is one of the most important step where we are getting knowledge from the data and preparing it for the further stage of analysis. Splitting is done on the data to understand the data even more by having words rather than sentences. Data Pre-processing has been carried out by using NLTK library which is well known library of python among all. It serves the tasks of Tokenisation, Normalization, Stemming and Part of Speech Tagging. As the name implies, Tokenisation refers to splitting where it can break text into sentences, sentences into words and so on. We have imported word-Tokenise from NLTK library for carrying out tokenisation. Normalization is the process of generalising text on same level in order to do the processing in a uniform way. For doing the normalization we have performed certain steps i.e. removing special characters, removing stop words, removing white space from text. Stemming and Lemmatization is also used here where stemming is converting words to its root form and Lemmatization is same as stemming with a difference of grouping words having same meaning to one word [6][7].

1.4 *Part-of-Speech Tagging*

NLTK is used primarily for text analysis and POS tagging is one of the great feature of this library. POS tagging is the process of grammatical tagging. While downloading NLTK it will provide us the tokenisers, chunks, other algorithms and all corpora. Description of text can also be called as tagging where each word is grammatically tagged in a corpus with a part of speech tag. Part of speech consists of nouns, verbs, adverbs, adjectives, pronouns, conjunction and others [8][9].

1.5 *Selecting Keywords*

tf.idf is the technique used here for identifying the significance level of each word in a document. Term frequency-inverse document frequency is the full form of tf.idf. The importance of word is directly proportional to number of times word occurred in a document but offset by the word frequency in a document. Term frequency is computed by dividing the number of times word appear in a document to the total number of words in that document. Inverse document frequency evaluate the amount of amount of information we are getting from each term. It evaluates the importance of term and determine the most frequent or least occurring words in the entire document.To carry out with this technique first we need to instantiate count vectorizer, generating word count for each word in corpus,using tf.idf transformer to compute idf values,generating tf.idf score for each document, form a count matrix. All words have their corresponding vectors to carry out with this technique. As we are working on vectors so we will use cosine similarity to extract important features. tf.idf transform the entire document into numeric form as it can't be processed in string form. For each term there is a need to find out how frequent they are in the document and based on their range decides their level of importance in the document. The basic steps for computing tf.idf is loading data , apply tf.idf technique by finding out vectors for each word followed by calculating cosine similarity and then extracting useful features with in the entire document. Cosine similarity is used to find out the similarity between two vectors and to identify most relevant link in entire document. It is preferred over euclidean distance because irrespective of being too far as per the euclidean distance documents still have some cosine angle between them with the fact that less the angle will be, the higher the similarity between the documents [10].

1.6 *Stemming or Morphological Analysis*

Stemming is the process of cutting down the words to their root words just like retrieval, retrieved, retrieves to retrieve. There are various algorithms used for stemming i.e, Potter's Stemming algorithm, Lovins Stemmer, Dawson Stemmer, Krovetz Stemmer, Xerox Stemmer, N-Gram Stemmer. Out of all, we have used Potter's algorithm. It works on the concept that

removing the inflexion ending from words in English language. It gives the optimal results as compared to other stemmers with having less probability of any error [11].

2 Discussion

We have used two libraries of python language to develop a system. First of all we are having hands on NLTK library used for text analysis and helps to carry out the task of tokenisation, post-tagging, stopwords, stemming and lemmatization. Secondly, sklearn library is used for using tf.idf technique by importing TFidVectorizer, CountVectorizer, TfidTransformer. System is developed in such a way where we first read the URL and getting the links associated with each URL. Then does the HTML parsing by using BeautifulSoup library as being the data of website it has html tags. It is important to convert the document into string and extract meta data from it in order to make our analysis more simpler. Further refinement of data is done while doing pre-processing where we carry out tokenization and normalization of data followed by pos tagging tf.id and stemming of the data. Finally, we find out the most relevant words and phrases that can be used as index terms for each iteration and saved the data in form of text file.

References

[1] https://www.tutorialspoint.com/natural$_l anguage_p rocessing/natural_l anguage_p rocessing_i nformation_r et$

[2] https://www.geeksforgeeks.org/introduction-to-natural-language-processing/

[3] https://www.javatpoint.com/text-data-mining

[4] https://www.geeksforgeeks.org/python-tools-world-web-scraping/

[5] https://www.geeksforgeeks.org/python-tools-world-web-scraping/

[6] https://www.kdnuggets.com/2018/03/text-data-preprocessing-walkthrough-python.html

[7] https://www.geeksforgeeks.org/python-lemmatization-with-nltk/

[8] https://www.geeksforgeeks.org/part-speech-tagging-stop-words-using-nltk-python/

[9] https://www.tutorialspoint.com/natural$_l$anguage$_p$rocessing/natural$_l$anguage$_p$rocessing$_p$art$_o$f$_s$peech$_t$ag

[10] https://www.geeksforgeeks.org/sklearn-feature-extraction-with-tf-idf/

[11] https://www.geeksforgeeks.org/introduction-to-stemming/