**Lab 1 - Exercises about Information Retrieval Assessment and Search**

A – Data processing and Basic Text Handling

1. Analyze the examples of programs in <https://www.nltk.org/book/> -- Especially, study examples of Chapter 1 and Chapter 2 regarding the use of various corpora datasets and frequency plots. You may also inspire from other tutorials on data plot in python such mathplotlib library. Write a program for importing Brown corpus and displays the frequency of individual words. Save the result as a separate text file on your local drive.
2. Write a script that displays the histogram of the thirty most frequent words in the corpus. Use appropriate annotation for the axis and title.
3. Repeat 2) considering the thirty less frequent words, and another graph for the thirty words whose frequency is in the middle range of frequencies (You are free de use your own approach to identify those words, but explain your reasoning by providing appropriate comments in your script).
4. Write a program that calculates the length of the words in Brown corpus, and displays the graph showing the variation of the word-length with respect to the frequency (For this purpose, all words with the same length should have their frequency summed up to get the frequency of that length).
5. Consider the modal words (will, must, might, may, could, can). Write a script that calculates the frequency of each of these words in Brown corpus. Write another script that calculates the length of the sentences in terms of number of words and number of characters of sentences where these modal word occurred.
6. Consider the default stop-word list for English language available in NLTK book. Write a script that calculates the number of stop words in each sentence of the brown corpus and the length of the sentence in number of words and characters. Then write a script that displays the frequency of number of stopwords versus the length of the sentence (both for number of words and number of characters).

B- Information Retrieval 1

1. Consider an academic journal of your own choice and collect 20 abstracts using a method of your own (can be a simple manual copy-and-past operation) in a single file. Save this file in your local desk. Save also the keywords mentioned in each abstract file (if the journal allows for keywords, otherwise you may use words of title of the paper as keywords) in another separate file.
2. Consider an information retrieval system where a keyword plays the role search query. Write a script that uses logical query-matching for five queries of your own choice (from the list of keywords) to find out whether a given query is found in the document or not, so that for each keyword input, the program outputs 1 if a logical matching is found (the given keyword is found in the abstract) and 0, otherwise.
3. Now instead, of compiling the abstracts into a single file, we want to keep each abstract as a separate file, labeled as A1, A2, …, A20. Write a script that constructs an inverted file of the abstract files. Then suggest a program that employs a simple string matching operation to output the list of files (abstract-file (A0, A1,..A20)) for each keyword.
4. We want to relax the assumption of exact matching between keywords and words of the abstract and allow the matching to be considered correct if 90% of the characters of the keywords are found in one word of the abstract. Write a script that implements this reasoning and display the result of your search operation.

C – Information Retrieval 2

1. Consider the access to Wikipedia as data repository source (see the example script provided in laboratory folder). Use beautiful soup program as in Chapt 3 NLK online book example to crawl all text issued from Wikipedia pages (no need to crawl the links of these pages).
2. Use scikit-learn library, countVectorizer and tf-Idf vectorizer to construct the tf-idf matrix representation of all documents (all identified Finish towns + Finland pages). Refine the search when different preprocessing strategies were employed (with and without stopwords, your own list of stopwords, lemmatization, …)
3. Consider a query “I will visit Oulu this summer and possibly Espoo”. Write down a program that inputs the query and outputs its tf-idf representation.
4. Write down a program that evaluates the search result of the query to each document by computing the inner product of the query tf-idf representation and each of the tf-idf document. Display the documents who achieved high score.
5. Repeat the above reasoning when you use countVector instead of tf-idf representation.