# Lab Assignment 4

## CIS 490/590 Big Data

**Sunnie Chung**

## Information Retrieval: Document Similarity Measure
## Preprocessing to Build Document Vectors for Web Page Content Analysis

A document can be represented as **a Bag of Words** where a document is represented in a set of thousand terms as features with Term Frequency(TF) - the *frequency* of each term occurring in each document - and Inverted Document Frequency. For this transformation, you have to build an Inverted Index (which is a Term Look_Up Table or Term Dictionary with Term Frequency (TF) and Document Frequency (DF). See how to build an Inverted Index for TF-IDF in Slides 10 – 14 in the Lecture Notes.

For text analysis tasks, for example, to find the Top N most related documents in a collection per a given user query (topics) in a Question Answering (QA) System, each document can be transformed to be represented as a vector of weights on the topic terms (topic words/keywords/phrases in bi-gram or a tri-gram) in TF-IDF.

**In this Lab4, Find the most related webpage out of 7 webpage collection per given 7 user given topics below:**

Construct a document vector with the 7 given topics below with their frequency count for each topic word in the following 6 webpages. The 7 terms for topics are the following 4 keywords and 3 phrases (bi-grams) as below.

User Search Topic Terms:
**research, data, mining, analytics, data mining, machine learning, deep learning**

Doc1:
https://www.edx.org/course/data-science-machine-learning
Doc2:
https://en.wikipedia.org/wiki/Engineering
Doc3:
http://my.clevelandclinic.org/research
Doc4:
https://en.wikipedia.org/wiki/Data_mining
Doc5:
https://en.wikipedia.org/wiki/Data_mining#Data_mining
Doc6:
http://cis.csuohio.edu/~sschung/

Right Click on each webpage -> view sourcing then save the html file as .txt file. Those 6 text files are your input files to process to count frequency of each topic word to construct 6 document vectors for the 6 docs as below.

| | doc1 | doc2 | doc3 | doc4 | doc5 | doc6 |
|---|---|---|---|---|---|---|
| research | | | | | | |
| data | | | | | | |
| mining | | | | | | |
| analytics | | | | | | |
| data mining | | | | | | |
| machine learning | | | | | | |
| deep learning | | | | | | |

Notes:

1. We want to count only the words appeared in the Webpage Text as the Content of the page, not the words included inside any tags <……> or any system generated scripts or html codes. You can count the words appeared in the title bar as well as in the Menu in the webpage

2. We do not want to count any subexpressions that are a part of another words.

   "Spin" should not be counted as "pin"

3. However, No case sensitive: Insert, insert, INSERT, insert are all counted as a same word.
4. The words from a same stem are counted as a same word.
   For example, program, programming, programed, programmable are all counted as "program".
   You can directly add OR conditions with all the variations of the words that are from a same stem to count all as a same word.

5. We want to count for a phrase (bi-grams or tri-grams) by counting occurring of 'data mining', for example, when 'data' immediately followed by 'mining'.
   This is usually done to add a discovered bi-gram or tri-gram in the term dictionary as a single term, for example, 'data mining' is added as a single term with 'data_mining' in the term dictionary with its frequency.

6. See FAQ for Lab4_IR on the class webpage for more guides.

**Common NLP Preprocessing Procedures for Text Analysis**

Minimum Requirements:

1. Remove all the special symbols like punctuation mark, question mark using the character deletion step of translate
2. Remove all stop words (Search for Stop word Lists or Python or R Libs)
3. Do Stemming to Reduce inflected (or sometimes derived) words to their word stem.
4. Convert uppercase to lowercase

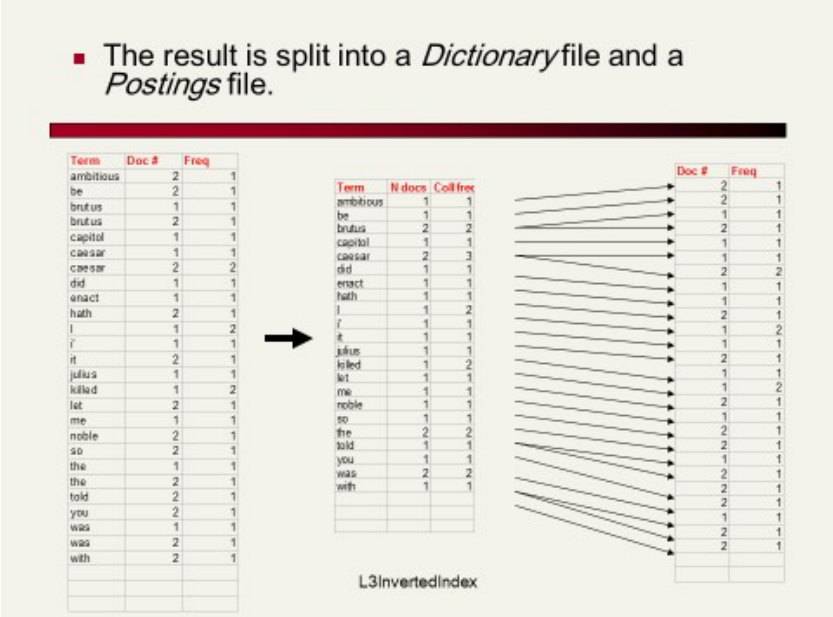
For more accurate Text Preprocessing, see Lab2 Section.

You can build your term dictionary with each term frequency (Inverted Index) first to construct 6 document vectors for the given Topics. Although IDF is highly recommended to add, you can ignore document frequency for simplicity for this lab.
You may omit to build an Inverted Index (Term Dictionary) with TF-IDF.
You can directly construct 6 document vectors to count from your parsing script for this lab.

**Extra Credit:**
**Inverted Index (Term Dictionary) Construction with TF-IDF will be counted as an Extra Credit.**

- You can use or adopt any online word count program for this Lab if you want. For example, import java.util.StringTokenizer;
- You can make any assumptions to simplify the program.
- Briefly make notes on these in your report.

You can create a Term Dictionary with TF (and DF) in a Table format or in MongoDB with the scheme. Your Term Dictionary with TF and DF would look like either one of those tables below.




For an Inverted Index in Mongo DB, See the Common NLP Tasks in the NLP Class Section.