# Lab Notebook

## Data cleaning:

Raw data was of the order of gigabytes. Therefore, it is necessary to do the pre-processing to remove the unwanted noise from the data.

I used the raw files in text format.

**Things that I removed-**

* All the non ASCII characters – This time, we are trying to build a visualizer based on English characters only. So, anything that falls outside of the ASCII charset can be eliminated.
* Numbers- Based on my assumption, the numbers really do not mean anything unless they appear with some word or phrase for example N1H1. I eliminated all the numbers because they were also contributing to a lot of noise like page numbers, figure numbers.
* Special Characters- Special characters do not add any meaning to the document clustering algorithm. So I removed them as well.
* Stop Words: These are those words which do not convey any meaning for example a, an, the, etc. I got rid of all of them. The Stop Word collection from Python NLTK is not good enough. So, I had to add more words to it.
* Single/Double Letter words: In my assumption, all those words having length of less than three characters, did not add much value to the clustering algorithm. I eliminated them also.

As a result of this, the size of the input file was reduced substantially. Now, my document clustering algorithm would have to process data of much smaller in size.

**Problems faced:**

There were a lot of Unicode characters in the text which threw a lot of exceptions while parsing the raw text. I had to figure out ways to overcome them.

**TF-IDF Calculation:**

**Term Frequency**: measures how frequently a term occurs in a document

**Inverse Document Frequency**: measures how important a term is

TF-IDF can be very useful in these cases –

* Compare documents in a set with other documents using cosine similarity
* Query the existing document set
* Plagiarism detection

**Serialization:**

I utilized MySQL to store the TF-IDF scores for the terms in the entire corpus

**Visualization:**

I utilized D3 visualization library which takes input as JSON and populates the visual elements

## Future course:

1. In future, I want to extend this approach of TF-IDF calculation to use Apache Spark. Spark is very efficient for iterative computation. It could be very useful for Big Data processing on a cluster.
2. I’ll also try to see the effect of other algorithms used in document clustering such as – K-Means clustering. I tried to implement K-Means but was stuck at what should be the value of K in clustering.
3. I’ll also want to compare the performance and the accuracy of the results between Java libraries and Python libraries.
4. I also need to figure out ways how I can identify the digits used phrases (like N1H1) with other extraneous digits (like page numbers, figure numbers, indexes, etc).
5. For really huge data set, I want to try this out using Hadoop Map Reduce.

## Libraries Used:

<http://bl.ocks.org/mbostock/4063269>