# **Design Document for Creating a Profession specific information retrieval system**

**Description:**

* A tf-idf based Search Engine for searching about 75 speeches from the Prime Minister Narendra Modi. The main purpose of this project is to understand how vector space based retrieval models work.
* More on [Tf-Idf](https://en.wikipedia.org/wiki/Tf%E2%80%93idf). Install all the dependencies using pip3.

## **The program/application can be broken down into the various subparts (actual file names also added) :**

Untitled2.py: Jupyter notebook which contains the following functionalities:

* Stemming
* Removal of stopwords and duplicates
* Creating the posting list
* Creating the Inverted Index
* Ranked Retrieval of top 10 most relevant results
* Auto-completion and suggestions
* Page Ranking based on TF-IDF scores

### **Order of executing the files.**

$ sudo python3 untitled2.py

Or we can run untitled2.py on Jupyter notebook

## **Installation:**

We installed the following packages:

os, nltk, csv, math

### **Installing nltk**

$ pip3 install nltk

$ python3

>>> import nltk

>>> nltk.download()

Packages: all

## **CORPUS USED:**

<https://www.kaggle.com/shankarpandala/mann-ki-baat-speech-corpus>

Corpus is in a collection of text files of the Prime Minister’s speeches.

## **DATA STRUCTURES USED:**

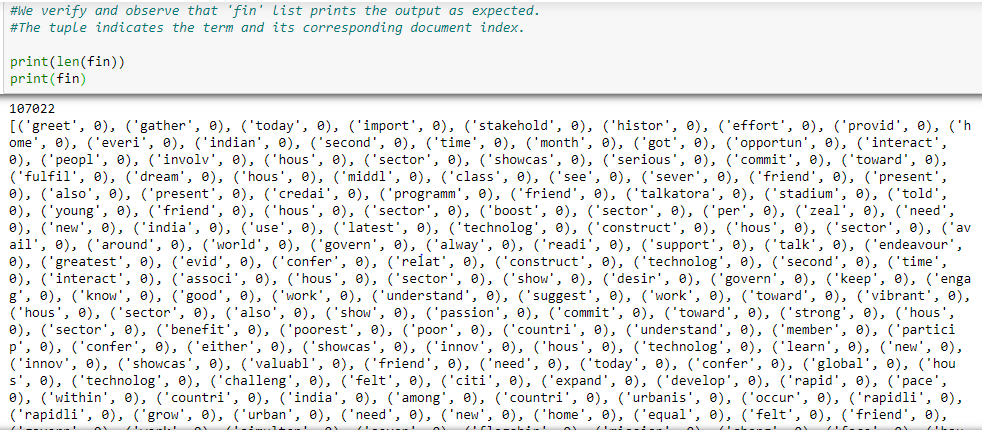
### **Tuple**

**Tuple** is a collection of Python objects much like a list. The sequence of values stored in a tuple can be of any type, and they are indexed by integers. The important difference between a list and a tuple is that tuples are immutable.

Example: ('aadhar', 64)

### **List (list)**

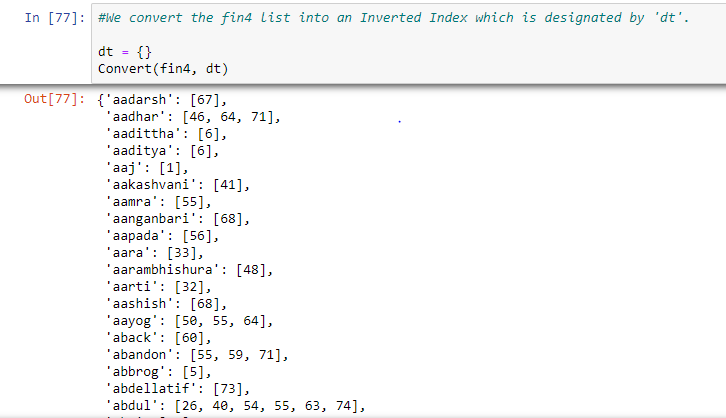
Contains lists enclosed within a list It will contain the stemmed tokens from each file in the corpus as individual lists. All are appended to make a list. Example:



*Snapshot of a list of tuples used in the project.*

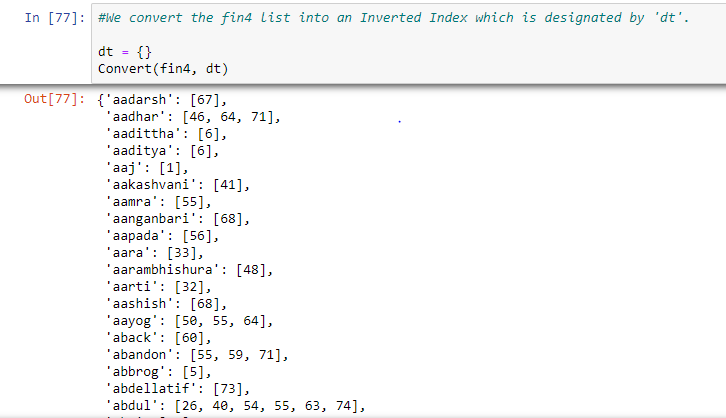
### **Dictionary (dict)**

Contains key-value pair. Dictionary keys are case sensitive, same name but different cases of Key will be treated distinctly.   
Example:[1: 'I', 2: 'am', 3: 'fine']  
  
In our project, Inverted Index has been stored using the dictionary data structure where  
< key: distinct-normalized-terms-in-corpus, value: posting-list >



### **Vocabulary**

Will contain a dictionary of all the unique words in the corpus. Example:



*Snapshot of the Inverted Index*

A nested dictionary containing the following structure explained through the following example:(Numbers are just representational )

## **Preprocessed Data:**

Preprocesses Data has been stored for the inverted index containing modified terms as dictionary arranged in alphabetic increasing order and the posting list(sorted) stored as lists.

Two separate columns for document frequency of each term as well the corresponding idf scores been stored in advance.

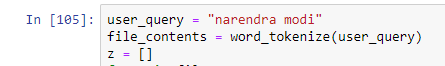
For tf scores the term frequency for each term stored in the form of logarithms in advance.

The data has been stored in the .csv files:

* dict.csv - contains the Inverted Index along with the document frequency and the IDF scores
* vct.csv - contains the TF scores in various documents

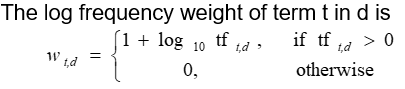
## **Boolean Retrieval:**

Traditional Inverted Index alongwith OR query has been used to achieve boolean retrieval.

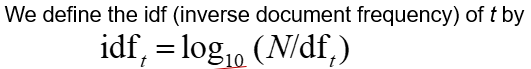




## **Formula Used:**

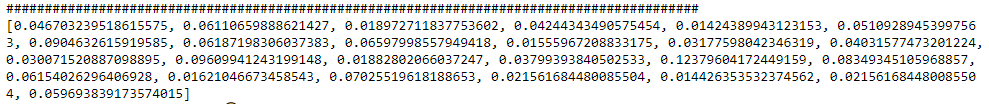


where tft,d is the term frequency of the term *t* in document *d*.



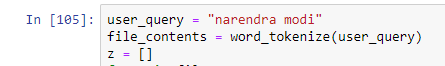
Where *N* is the number of documents in the corpus and *dft* is the document frequency of the term *t*.

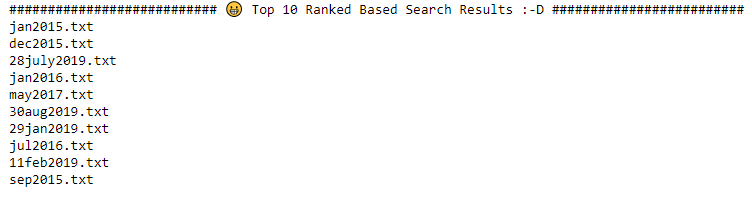
## **TF-IDF Scores:**



## **Ranked Retrieval:**

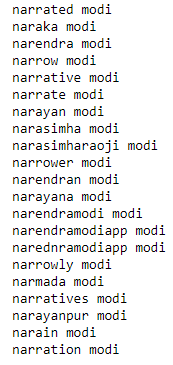
Vector space model has been used for ranked retrieval of the relevant documents.





## **Auto-completion and Wildcard Queries:**





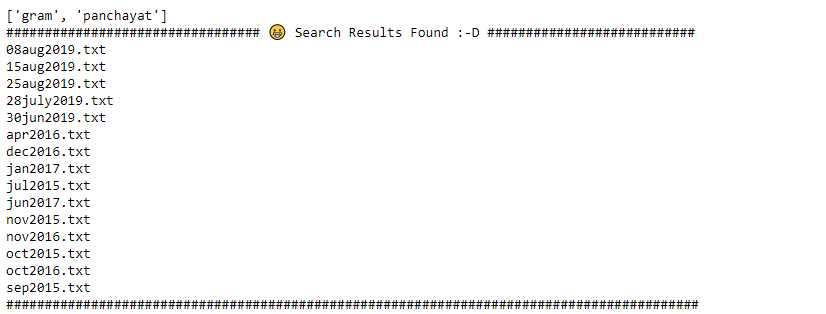
## **Machine specs:**

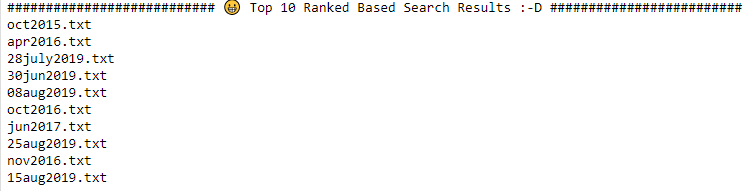
## **Results**

Index building time:

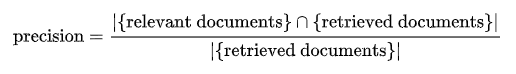
* Stemming (using PorterStemmer()) ~10s
* Indexing ~145s
* Information Retrieval ~3-4s
* **Precision and Recall:**







***Precision :***



11/15 = 73.33*% (for inverted index)*

100% (for ranked retrieval system) *[ Based on query = “Gram - Panchayat” ]*

***Recall:***

****

100*% (for inverted index)*

10/15 = 66.67% *(for ranked retrieval system) [ Based on query = “Gram - Panchayat” ]*

## **Members**

Pratik (2016B4A70549H)

Anshuman Pati (2016B4A70470H)

Piyush Jain (2016A3PS0885H)