# Pre processing

Uses gensim to generate TFIDFs and subsequently WORDIDS. These are generated using the Wikipedia corpus by running make\_wiki.py file. These TFIDFs and WORDIDs are used by LSA encoder.

1. Create bag of words of each sentence.(stopwords and punctuations are removed)

2. Generate weights of the words in the bag of words.(using tfidf)

3. Pass these weights to Singular Value Decomposition.

4. The indices of the top weights of the right singular matrix of the SVD are used to generate bit arrays.

Gensim is used for the above

# Training and testing the model

Change the training and testing file names in train\_test.py file. Populate the category names and ids in category\_list.csv file. The predictions will be received in result\_output.csv file.

# Files details

Filename: a.py

Input: text corpus (similar to wiki corpus)

Output: mm corpus

Filename: category\_list.py

Input: training\_set.csv

Output: category\_list with labels category\_count with its counts refined\_training\_set.csv

Filename: split\_final.py

Input: refined\_trainig\_set.csv

Output: splitted the training set into number of categories having equal records in it. Created train\_set.csv.created test\_set.csv

Filename: driver\_function.py

Input: test\_set.csv, train\_set.csv

Function: starts the categorization

Filename: lsa\_encoder.py

Input: mm corpus from a.py word\_ids.txt from a.py

Output: tfidf\_model model

Returns: bit arrays

Filename: language\_encoder.py

Function: parent of lsa\_encoder.py

Filename: sdr\_classifier.py

Input: active indices

Output: probability distribution

Filename: make\_wiki.py

Input: wiki corpus

Output: tfidf and wordIds for LSA

# Execution Steps

Step1:

Create mm corpus using a.py file. Input is the corpus which is going to then be splitted into the train and the test set.

Step 2:

Run the category\_list.py file. It takes the corpus which has to be splitted into the train and the test set.

This file gives three files as mentioned above.

Step 3:

The output files in the above step are then given to the split\_final.py file. Removes the stop words and single character words. The code in split\_final.py splits refined\_corpus.csv into train\_set.csv and test\_set.csv with ratio 90 - 10 respectively.

Step 4:

Run the driver\_function.py. This file takes input from the output of the previous step. This file uses lsa\_encoder.py file to convert text into bit arrays by creating bow and tfidf model. These bit arrays are then passed to the spatial\_pooler.py in the nupic library to get converted into Sparse Distributed Representations (SDRs). These sdrs are then passed on to the sdr\_classifier.py to get trained and classify into pre-defined categories.