**Document Keyword Extraction**

**Libraries Used**

1. CountVectorizer – for calculation frequency of each word in the texts.
2. Pandas – for handling data structures.
3. Stopwords – English language stop words
4. TfidfTransformer – transforming frequency distribution of words to tf-idf(Term Frequency – Inverse Document Frequency)
5. Re – for working with regex
6. Counter – for maintaining counts for each word (while using search for keywords)
7. Json – for working with json formatted data

**Functions**

1. sort\_on\_count – Based on the tf-idf value, sort the tf-idf vector
2. extract\_topn – extract the top n features (keywords) using the tf-idf vector previously sorted.
3. Tokenize – tokenize a given sentence (text)
4. Probability – probability of each word present in the entire document corpus (‘train.txt’)
5. Known – returns the set of known words for a given keyword (fuzzified)
6. edit\_dist\_1 – returns the set of words which have a edit distance of 1 with any provided word.
7. edit\_dist\_2 – returns the set of words which have a edit distance of 2 with any provided word.
8. Find – finds a given keyword present in the short text.

**Overview**

The code does the following :-

1. **Keyword extraction** – Tf-idf has been used for keyword extraction. The text is loaded into two variables ; *train\_docs and test\_docs* . The frequency count of each word is computed on the training document. Based on the count vector trained, values are transformed as TF-IDF.

Based on the TF-IDF vector and count vector previously trained, keywords are found from the test documents using sort\_on\_count and extract\_topn.

1. **Search Keywords** – Based on the keywords extracted in the previous part, each keyword is searched for in the short text based on edit distance algorithm.