**TF-IDF**:

<https://stackabuse.com/python-for-nlp-creating-tf-idf-model-from-scratch/>

One of the main problems associated with the bag of words model is that it assigns equal value to the words, irrespective of their importance.

The idea behind the TF-IDF approach is that the words that are more common in one sentence and less common in other sentences should be given high weights.

Like the bag of words, the first step to implement TF-IDF model, is tokenization.

Once you have tokenized the sentences, the next step is to find the TF-IDF value for each word in the sentence.

The TF value refers to term frequency and can be calculated as follows:

TF = (Frequency of the word in the sentence) / (Total number of words in the sentence)

IDF refers to inverse document frequency and can be calculated as follows:

IDF: (Total number of sentences (documents))/(Number of sentences (documents) containing the word)

It is important to mention that the IDF value for a word remains the same throughout all the documents as it depends upon the total number of documents. On the other hand, TF values of a word differ from document to document.

Finally, the TF-IDF values are calculated by multiplying TF values with their corresponding IDF values. Those values are TF-IDF vectors for each word in the respective sentences.

To find the TF-IDF value, we first need to create a dictionary of word frequencies.

It is important to mention that to mitigate the effect of very rare and very common words on the corpus, the log of the IDF value can be calculated before multiplying it with the TF-IDF value. In such case the formula of IDF becomes:

IDF: log((Total number of sentences (documents))/(Number of sentences (documents) containing the word))

Higlish stopwords taken from: <https://github.com/TrigonaMinima/HinglishNLP/blob/master/data/assets/stop_hinglish>

<https://www.analyticsvidhya.com/blog/2021/07/bag-of-words-vs-tfidf-vectorization-a-hands-on-tutorial/>

**Gensim**:

Punkt Sentence Tokenizer:

This tokenizer divides a text into a list of sentences by using an unsupervised algorithm to build a model for abbreviation words, collocations, and words that start sentences. It must be trained on a large collection of plaintext in the target language before it can be used.

The NLTK data package includes a pre-trained Punkt tokenizer for English.