WEEK- 11

NATURAL LANGUAGE PROCESSING

Session 6

11.12 IF-TDF VECTORIZATION

All the methods discussed in the previous sessions were based on the Bag of Words model which is simple and works well. But the problem with that is that it treats all words equally. As a result, it cannot distinguish very common words or rare words. So, to solve this problem, TF-IDF comes into the picture. TF-IDF is made up of two terms: Term Frequency and Inverse Document Frequency

1. Term Frequency

Term frequency denotes the frequency of a word in a document. For a specified word, it is defined as the ratio of the number of times a word appears in a document to the total number of words in the document. Or, it is also defined in the following manner:

It is the percentage of the number of times a word (x) occurs in a particular document (y) divided by the total number of words in that document.

$$TF(\boldsymbol{term}) = \frac{Number\ of\ times\ \boldsymbol{term}\ appears\ in\ a\ document}{Total\ number\ of\ items\ in\ the\ document}$$

For Example, Consider the following sentence

'Cat loves to play with ball'

For the above sentence, the term frequency value for word cat will be: tf('cat') = 1/6

This number will always stay ≤ 1, thus we now judge how frequent a word is in the context of all of the words in a document.

2. Inverse document frequency

Inverse document frequency looks at how common (or uncommon) a word is amongst the corpus. It measures the importance of the word in the corpus.

Before we go into IDF, we must make sense of DF – Document Frequency. It's given by the following formula:

$$DF = \frac{Documents\ containing\ word\ W}{Total\ number\ of\ documents}$$

DF tells us about the proportion of documents that contain a certain word. IDF is the reciprocal of the Document Frequency.

$$IDF = log\left(\frac{Total\ number\ of\ documents}{Documents\ containing\ word\ W}\right)$$

The intuition behind using IDF is that the more common a word is across all documents, the lesser its importance is for the current document. A logarithm is taken to dampen the effect of (normalize) IDF in the final calculation. The final TF-IDF score comes out to be:

$$TFIDF = TF * IDF$$

Implementation

```
from sklearn.feature_extraction.text import TfidfVectorizer
sents = ['coronavirus is a highly infectious disease',
   'coronavirus affects older people the most',
   'older people are at high risk due to this disease']
```

```
tfidf = TfidfVectorizer()
transformed = tfidf.fit_transform(sents)

import pandas as pd
df = pd.DataFrame(transformed[0].T.todense(),
                  index=tfidf.get_feature_names_out(),
                  columns=["TF-IDF"])
df = df.sort_values('TF-IDF', ascending=False)
df
```

	TF-IDF
infectious	0.490479
highly	0.490479
is	0.490479
coronavirus	0.373022
disease	0.373022
older	0.000000
this	0.000000
the	0.000000
risk	0.000000
people	0.000000
affects	0.000000
most	0.000000
are	0.000000
high	0.000000
due	0.000000
at	0.000000
to	0.000000

References:

1. https://www.analyticsvidhya.com/blog/2021/06/part-5-step-by-step-guide-to-master-nlp-text-vectorization-approaches/
2. https://neptune.ai/blog/vectorization-techniques-in-nlp-guide
3. https://towardsdatascience.com/nlp-in-python-vectorizing-a2b4fc1a339e
4. Infosys Springboard – Natural Language Processing for developers