# Bag-of-words vs TFIDF vectorization

Whenever we apply any algorithm to textual data, we need to convert the text to a numeric form. Hence, there arises a need for some pre-processing techniques that can convert our text to numbers. Both bag-of-words (BOW) and TFIDF are pre-processing techniques that can generate a numeric form from an input text.

## Bag-of-Words:

The bag-of-words model converts text into fixed-length vectors by counting how many times each word appears.

Let us illustrate this with an example. Consider that we have the following sentences:

- **Text processing is necessary.**
- **Text processing is necessary and important.**
- **Text processing is easy.**

We will refer each of the above sentences as documents. If we take out the unique words in all these sentences, the vocabulary will consist of these 7 words:

**{‘Text’, ’processing’, ’is’, ’necessary’, ’and’, ’important, ’easy’}.**

To carry out bag-of-words, we will simply have to count the number of times each word appears in each of the documents.

| **Document** | Text | preprocessing | is | necessary | and | important | easy |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

Hence, we have the following vectors for each of the documents of fixed length -7:

**Document 1**: [1,1,1,1,0,0,0]

**Document 2**: [1,1,1,1,1,1,0]

**Document 3**: [1,1,1,0,0,0,1]

## *Limitations of Bag-of-Words:*

- If we deploy bag-of-words to generate vectors for large documents, the vectors would be of large sizes and would also have too many null values leading to the creation of sparse vectors.
- Bag-of-words does not bring in any information on the meaning of the text. For example, if we consider these two sentences – "Text processing is easy but tedious." and "Text processing is tedious but easy." – a bag-of-words model would create the same vectors for both of them, even though they have different meanings.

## Term Frequency Inverse Document Frequency (TFIDF):

TFIDF works by proportionally increasing the number of times a word appears in the document but is counterbalanced by the number of documents in which it is present. Hence, words like 'this', 'are' etc.,that are commonly present in all the documents are not given a very high rank. However, a word that is present too many times in a few of the documents will be given a higher rank as it might be indicative of the context of the document.

## *Term Frequency:*

Term frequency is defined as the number of times a word (i) appears in a document (j) divided by the total number of words in the document.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

## *Inverse Document Frequency:*

Inverse document frequency refers to the log of the total number of documents divided by the number of documents that contain the word. The logarithm is added to dampen the importance of a very high value of IDF.

$$idf(w) = log(\frac{N}{df_t})$$

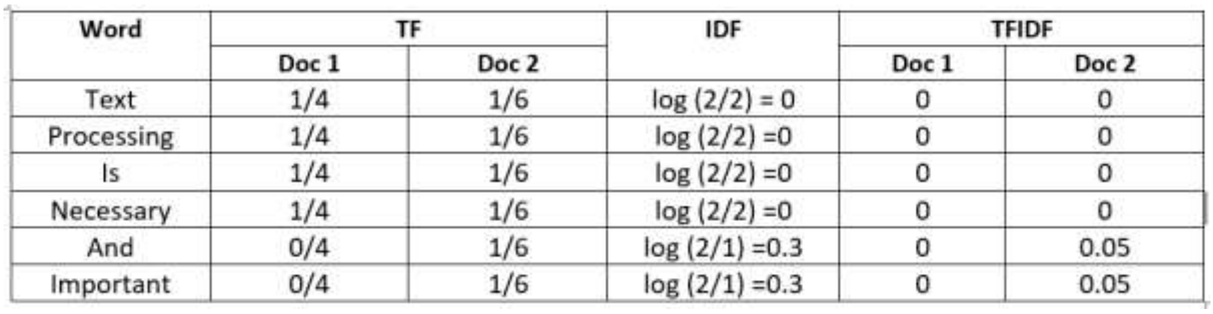
TFIDF is computed by multiplying the term frequency with the inverse document frequency.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Let us now see an illustration of TFIDF in the following sentences, that we refer to as documents.

**Document 1:** Text processing is necessary.

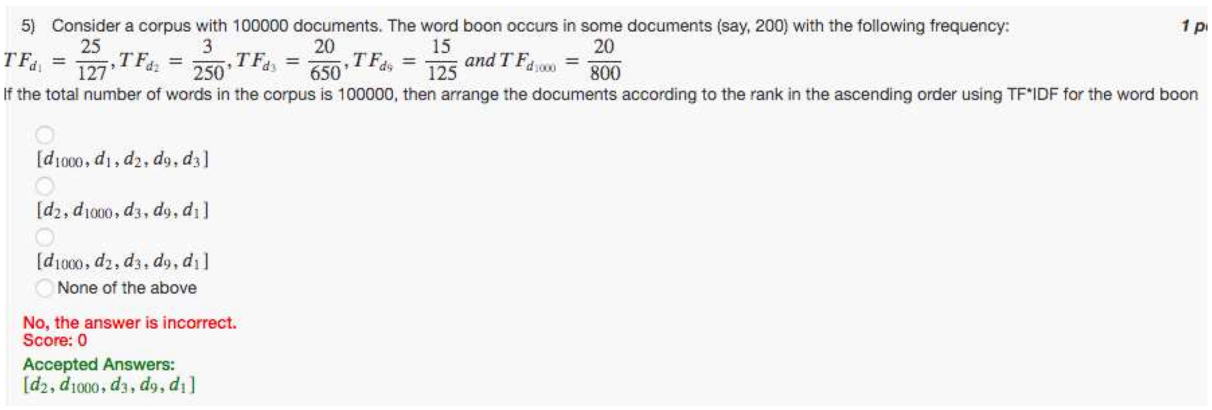
**Document 2:** Text processing is necessary and important.



| Word | TF | | IDF | TFIDF | |
|---|---|---|---|---|---|
| | Doc 1 | Doc 2 | | Doc 1 | Doc 2 |
| Text | 1/4 | 1/6 | log (2/2) = 0 | 0 | 0 |
| Processing | 1/4 | 1/6 | log (2/2) =0 | 0 | 0 |
| Is | 1/4 | 1/6 | log (2/2) =0 | 0 | 0 |
| Necessary | 1/4 | 1/6 | log (2/2) =0 | 0 | 0 |
| And | 0/4 | 1/6 | log (2/1) =0.3 | 0 | 0.05 |
| Important | 0/4 | 1/6 | log (2/1) =0.3 | 0 | 0.05 |

The above table shows how the TFIDF of some words are zero and some words are non-zero depending on their frequency in the document and across all documents.

The limitation of TFIDF is again that this vectorization doesn't help in bringing in the contextual meaning of the words as it is just based on the frequency.



5) Consider a corpus with 100000 documents. The word boon occurs in some documents (say, 200) with the following frequency: **1 p**

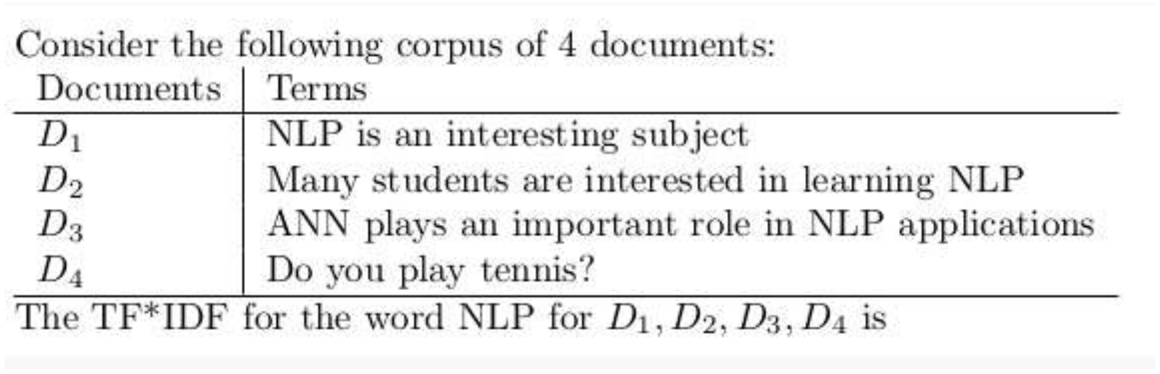
$TF_{d_1} = \frac{25}{127}, TF_{d_2} = \frac{3}{250}, TF_{d_3} = \frac{20}{650}, TF_{d_9} = \frac{15}{125}$ $and$ $TF_{d_{1000}} = \frac{20}{800}$

If the total number of words in the corpus is 100000, then arrange the documents according to the rank in the ascending order using TF*IDF for the word boon

- $[d_{1000}, d_1, d_2, d_9, d_3]$
- $[d_2, d_{1000}, d_3, d_9, d_1]$
- $[d_{1000}, d_2, d_3, d_9, d_1]$
- None of the above

No, the answer is incorrect.
Score: 0

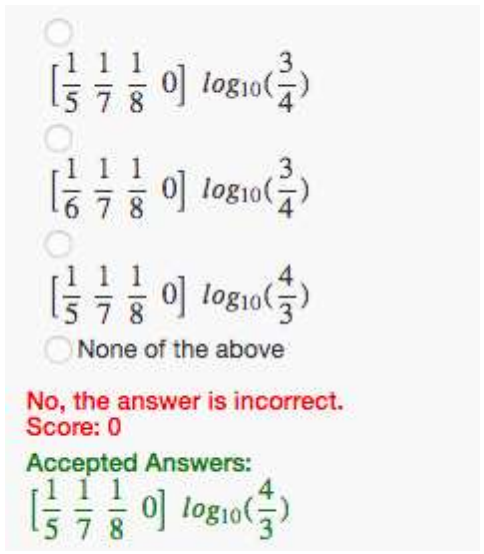
Accepted Answers:
$[d_2, d_{1000}, d_3, d_9, d_1]$



Consider the following corpus of 4 documents:

| Documents | Terms |
|---|---|
| $D_1$ | NLP is an interesting subject |
| $D_2$ | Many students are interested in learning NLP |
| $D_3$ | ANN plays an important role in NLP applications |
| $D_4$ | Do you play tennis? |

The TF*IDF for the word NLP for $D_1, D_2, D_3, D_4$ is



- $[\frac{1}{5}\ \frac{1}{7}\ \frac{1}{8}\ 0]\ log_{10}(\frac{3}{4})$
- $[\frac{1}{6}\ \frac{1}{7}\ \frac{1}{8}\ 0]\ log_{10}(\frac{3}{4})$
- $[\frac{1}{5}\ \frac{1}{7}\ \frac{1}{8}\ 0]\ log_{10}(\frac{4}{3})$
- None of the above

No, the answer is incorrect.
Score: 0

Accepted Answers:
$[\frac{1}{5}\ \frac{1}{7}\ \frac{1}{8}\ 0]\ log_{10}(\frac{4}{3})$