BAG OF WORDS AND TF-IDF

- Laxminarayen N V

BAG OF WORDS

- The Bag of Words (BoW) model is the simplest form of text representation in numbers. Like the term itself, we can represent a sentence as a bag of words vector (a string of numbers).

- the three types of movie reviews
- Review 1: This movie is very scary and long
- Review 2: This movie is not scary and is slow
- Review 3: This movie is spooky and good

BAG OF WORDS

- We will first build a vocabulary from all the unique words in the above three reviews. The vocabulary consists of these 11 words: 'This', 'movie', 'is', 'very', 'scary', 'and', 'long', 'not', 'slow', 'spooky', 'good'.
- We can now take each of these words and mark their occurrence in the three movie reviews above with 1s and 0s. This will give us 3 vectors for 3 reviews:

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

DRAWBACKS OF USING A BAG-OF-WORDS (BOW) MODEL

- If the new sentences contain new words, then our vocabulary size would increase and thereby, the length of the vectors would increase too.

- Additionally, the vectors would also contain many 0s, thereby resulting in a sparse matrix (which is what we would like to avoid)

- We are retaining no information on the grammar of the sentences nor on the ordering of the words in the text.

TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF)

- Let's first put a formal definition around TF-IDF. Here's how Wikipedia puts it:

- *"Term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus."*

TERM FREQUENCY (TF)

- Let's first understand Term Frequent (TF). It is a measure of how frequently a term, t, appears in a document, d:

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

Here, in the numerator, n is the number of times the term "t" appears in the document "d". Thus, each document and term would have its own TF value.

EXAMPLE

- We will again use the same vocabulary we had built in the Bag-of-Words model to show how to calculate the TF for Review #2:

Review 2: This movie is not scary and is slow

- Here,

Vocabulary: 'This', 'movie', 'is', 'very', 'scary', 'and', 'long', 'not', 'slow', 'spooky', 'good'

Number of words in Review 2 = 8

TF for the word 'this' = (number of times 'this' appears in review 2)/(number of terms in review 2) = 1

EXAMPLE

- Similarly,
- TF(‘movie’) = 1/8
- TF(‘is’) = 2/8 = 1/4
- TF(‘very’) = 0/8 = 0
- TF(‘scary’) = 1/8
- TF(‘and’) = 1/8
- TF(‘long’) = 0/8 = 0
- TF(‘not’) = 1/8
- TF(‘slow’) = 1/8
- TF(‘spooky’) = 0/8 = 0
- TF(‘good’) = 0/8 = 0

TF TABLE

Term	Review 1	Review 2	Review 3	TF (Review 1)	TF (Review 2)	TF (Review 3)
This	1	1	1	1/7	1/8	1/6
movie	1	1	1	1/7	1/8	1/6
is	1	2	1	1/7	1/4	1/6
very	1	0	0	1/7	0	0
scary	1	1	0	1/7	1/8	0
and	1	1	1	1/7	1/8	1/6
long	1	0	0	1/7	0	0
not	0	1	0	0	1/8	0
slow	0	1	0	0	1/8	0
spooky	0	0	1	0	0	1/6
good	0	0	1	0	0	1/6

INVERSE DOCUMENT FREQUENCY (IDF)

- IDF is a measure of how important a term is. We need the IDF value because computing just the TF alone is not sufficient to understand the importance of words:

$$idf_t = \log \frac{number\ of\ documents}{number\ of\ documents\ with\ term\ 't'}$$

EXAMPLE

- We can calculate the IDF values for the all the words in Review 2:
- IDF('this') = log(number of documents/number of documents containing the word 'this') = log(3/3) = log(1) = 0
- Similarly,

- IDF('movie',) = log(3/3) = 0
- IDF('is') = log(3/3) = 0
- IDF('not') = log(3/1) = log(3) = 0.48
- IDF('scary') = log(3/2) = 0.18
- IDF('and') = log(3/3) = 0
- IDF('slow') = log(3/1) = 0.48

IDF TABLE

- We can calculate the IDF values for each word like this. Thus, the IDF values for the entire vocabulary would be:

Term	Review 1	Review 2	Review 3	IDF
This	1	1	1	0.00
movie	1	1	1	0.00
is	1	2	1	0.00
very	1	0	0	0.48
scary	1	1	0	0.18
and	1	1	1	0.00
long	1	0	0	0.48
not	0	1	0	0.48
slow	0	1	0	0.48
spooky	0	0	1	0.48
good	0	0	1	0.48

TF-IDF

- **Hence, we see that words like "is", "this", "and", etc., are reduced to 0 and have little importance; while words like "scary", "long", "good", etc. are words with more importance and thus have a higher value.**
- We can now compute the TF-IDF score for each word in the corpus. Words with a higher score are more important, and those with a lower score are less important:

$$(tf\_idf)_{t,d} = tf_{t,d} * idf_t$$

EXAMPLE

- We can now calculate the TF-IDF score for every word in Review 2:
- TF-IDF('this', Review 2) = TF('this', Review 2) * IDF('this') = 1/8 * 0 = 0
- Similarly,
- TF-IDF('movie', Review 2) = 1/8 * 0 = 0
- TF-IDF('is', Review 2) = 1/4 * 0 = 0
- TF-IDF('not', Review 2) = 1/8 * 0.48 = 0.06
- TF-IDF('scary', Review 2) = 1/8 * 0.18 = 0.023
- TF-IDF('and', Review 2) = 1/8 * 0 = 0
- TF-IDF('slow', Review 2) = 1/8 * 0.48 = 0.06

TF-IDF TABLE

Term	Review 1	Review 2	Review 3	IDF	TF-IDF (Review 1)	TF-IDF (Review 2)	TF-IDF (Review 3)
This	1	1	1	0.00	0.000	0.000	0.000
movie	1	1	1	0.00	0.000	0.000	0.000
is	1	2	1	0.00	0.000	0.000	0.000
very	1	0	0	0.48	0.068	0.000	0.000
scary	1	1	0	0.18	0.025	0.022	0.000
and	1	1	1	0.00	0.000	0.000	0.000
long	1	0	0	0.48	0.068	0.000	0.000
not	0	1	0	0.48	0.000	0.060	0.000
slow	0	1	0	0.48	0.000	0.060	0.000
spooky	0	0	1	0.48	0.000	0.000	0.080
good	0	0	1	0.48	0.000	0.000	0.080

END NOTES

- Bag of Words just creates a set of vectors containing the count of word occurrences in the document (reviews), while the TF-IDF model contains information on the more important words and the less important ones as well.

- Bag of Words vectors are easy to interpret. However, TF-IDF usually performs better in machine learning models.

FURTHER READ

- While both Bag-of-Words and TF-IDF have been popular in their own regard, there still remained a void where understanding the context of words was concerned. Detecting the similarity between the words ‘spooky’ and ‘scary’, or translating our given documents into another language, requires a lot more information on the documents.

- This is where Word Embedding techniques such as Word2Vec, Continuous Bag of Words (CBOW), Skipgram, etc. come in.