Feature Extraction

Bag of Words

and

Term frequency inverse document frequency(Tf-IDF)

BOW (Count vectorizer)

- 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

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

Vector of Review 1: [1 1 1 1 1 1 1 0 0 0 0]

Vector of Review 2: [1 1 2 0 0 1 1 0 1 0 0]

Vector of Review 3: [1 1 1 0 0 0 1 0 0 1 1]

Drawbacks of using a Bag-of-Words (BoW) Model

In the above example, we can have vectors of length 11. However, we start facing issues when we come across new sentences:

1. If the new sentences contain new words, then our vocabulary size would increase and thereby, the length of the vectors would increase too.
2. Additionally, the vectors would also contain many 0s, thereby resulting in a sparse matrix (which is what we would like to avoid)
3. We are retaining no information on the grammar of the sentences nor on the ordering of the words in the text.

Tfidf

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.

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) = 1/8

Term Frequency

Length of every vector = size of vocabulary

- 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

We can calculate the term frequencies for all the terms and all the reviews in this manner:

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

Concept : Word present in all document is least relevant

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:

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

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

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

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:

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

We can now calculate the TF-IDF score for every word in Review 2:

$$\text{TF-IDF('this', Review 2)} = \text{TF('this', Review 2)} * \text{IDF('this')} = 1/8 * 0 = 0$$

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

Similarly, we can calculate the TF-IDF scores for all the words with respect to all the reviews:

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

We have now obtained the TF-IDF scores for our vocabulary. TF-IDF also gives larger values for less frequent words and is high when both IDF and TF values are high i.e the word is rare in all the documents combined but frequent in a single document.

- Till now tf-idf has been applied to one word only .
- Tf-idf vectorizer can also be applied to n gram(e.g. Word bi gram) in which it will calculate relevant word bigram.

Example #2

- *Example: If we are given 4 reviews for an Italian pasta dish.*
- *Review 1 : This pasta is very tasty and affordable.*
- *Review 2: This pasta is not tasty and is affordable.*
- *Review 3 : This pasta is delicious and cheap.*
- *Review 4: Pasta is tasty and pasta tastes good.*

- Now if we count the number of unique words in all the four reviews we will be getting a total of 12 unique words.

• Below are the 12 unique words :

1.‘This’
2.‘pasta’
3.‘is’
4.‘very’
5.‘tasty’
6.‘and’
7.‘affordable’
8.‘not’
9.‘delicious’
10.‘cheap’
11.‘tastes’
12.‘good’

Now if we take the first review and plot count of each word in the below table we will have where row 1 corresponds to the index of the unique words and row 2 corresponds to the number of times a word occurs in a review.

(H

1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	1	1	1	0	0	0	0	0

ty and

af

1	2	3	4	5	6	7	8	9	10	11	12
0	2	1	0	1	1	0	0	0	0	1	1

es good.

- BOW doesn’t work very well when there are small changes in the terminology we are using as here we have sentences with similar meaning but with just different words.
- This results in a vector with lots of zero scores called a sparse vector or sparse representation.
- Sparse vectors require more memory and computational resources when modeling and the vast number of

There are simple **text cleaning techniques** that can be used as a first step, such as:

- *Ignoring case*
- *Ignoring punctuation*
- *Ignoring frequent words that don’t contain much information, called stop words, like “a,” “of,” etc.*
- *Fixing misspelled words.*
- *Reducing words to their stem (e.g. “play” from “playing”) using stemming algorithms.*

N-grams Model:

- A more sophisticated approach is to create a vocabulary of grouped words. This changes both the scope of the vocabulary and allows the bag-of-words to capture a little bit more meaning from the document.
- In this approach, each word or token is called a "**gram**". Creating a vocabulary of two-word pairs is, in turn, called a **bigram** model. Again, only the bigrams that appear in the corpus are modeled, not all possible bigrams.

> *An N-gram is an N-token sequence of words: a 2-gram (more commonly called a bigram) is a two-word sequence of words like "please turn", "turn your", or "your homework", and a 3-gram (more commonly called a trigram) is a three-word sequence of words like "please turn your", or "turn*

TF-IDF:

tf–idf or **TFIDF**, short for **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.

The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

tf–idf is one of the most popular term-weighting schemes today; 83% of text-based recommender systems in digital libraries use tf-idf.

Transcription includes:

· ***Counts***. *Count the number of times each word appears in a document.*

· ***Frequencies***. *Calculate the frequency that each word appears in a document out of all the words in the document.*

Term frequency :

- Term frequency (TF) is used in connection with information retrieval and shows how frequently an expression (term, word) occurs in a document.
- Term frequency indicates the significance of a particular term within the overall document. It is the number of times a word w_i occurs in a review r_j with respect to the total numbe

$$TF(w_i, r_j) = \frac{\text{No. of times } w_i \text{ occurs in } r_j}{\text{Total no. of words in } r_j}$$

TF can be said as what is the probability of finding a word in a document (review).

Inverse document frequency:

- The inverse document frequency is a measure of how much information the word provides, i.e., if it’s common or rare across all documents.
- It is used to calculate the weight of rare words across all documents in the corpus.
- The words that occur rarely in the corpus have a high IDF score. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient)

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

- N: total number of documents in the corpus $N = |D|$
- $|\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $\mathrm{tf}(t, d) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$.

Term frequency–Inverse document frequency:

- TF–IDF is calculated as

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

A high weight in tf–idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms.

Since the ratio inside the IDF's log function is always greater than or equal to 1, the value of IDF (and tf–idf) is greater than or equal to 0.

As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the IDF and tf–idf closer to 0.

TF-IDF gives larger values for less frequent words in the document corpus. TF-IDF value is high when both IDF and TF values are high i.e the word is rare in the whole document but frequent in a document.
TF-IDF also doesn’t take the semantic meaning of the words.

Let’s take an example to get a clearer understanding.

- Sentence 1: The car is driven on the road.
- Sentence 2: The truck is driven on the highway.
- In this example, each sentence is a separate document.
- We will now calculate the TF-IDF for the above two documents, which represent

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

From the mentioned table, we can see that the TF-IDF of common words was zero, which shows they are not significant.

On the other hand, the TF-IDF of “car”, “truck”, “road”, and “highway” are non-zero. These words have more significance.

Reviewing- TFIDF is the product of the TF and IDF scores of the term.

TF = number of times the term appears in the doc/total number of words in the doc

IDF = ln(number of docs/number docs the term appears in)

Higher the TFIDF score, the rarer the term is and vice-versa.

TFIDF is successfully used by search engines like Google, as a ranking factor for content.

The whole idea is to weigh down the frequent terms while scaling up the rare ones.

Word2Vec :

- **Word Embedding** is a word representation type that allows machine learning algorithms to understand words with similar meanings. It is a language modeling and feature learning technique to map words into vectors of real numbers using neural networks, probabilistic models, or dimension reduction on the word co-occurrence matrix. Some word embedding models are Word2vec (Google), Glove (Stanford), and fastest (Facebook).
- Word2Vec model is used for learning vector representations of words called “word embeddings”.
- This is typically done as a preprocessing step, after which the learned vectors are fed into a discriminative model (typically an RNN) to generate predictions and perform all sorts of interesting things.
- It takes the semantic meaning of words.