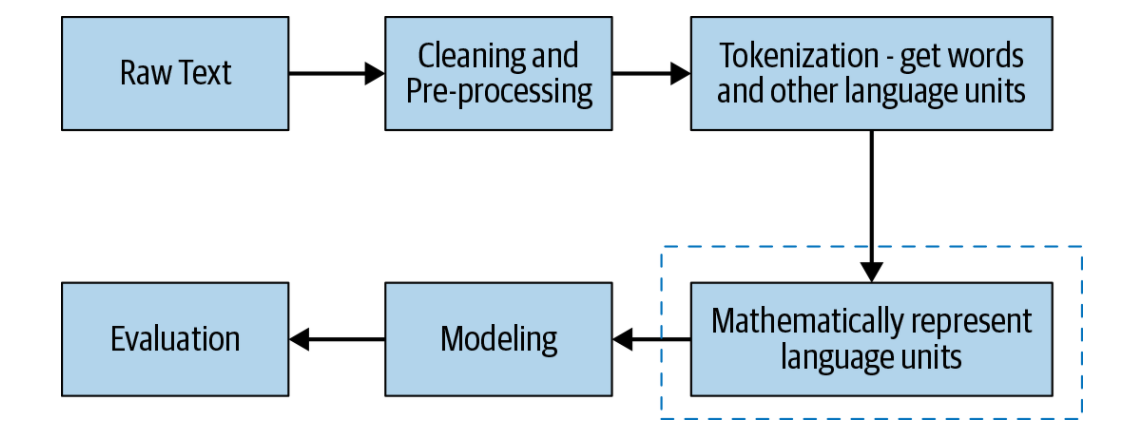
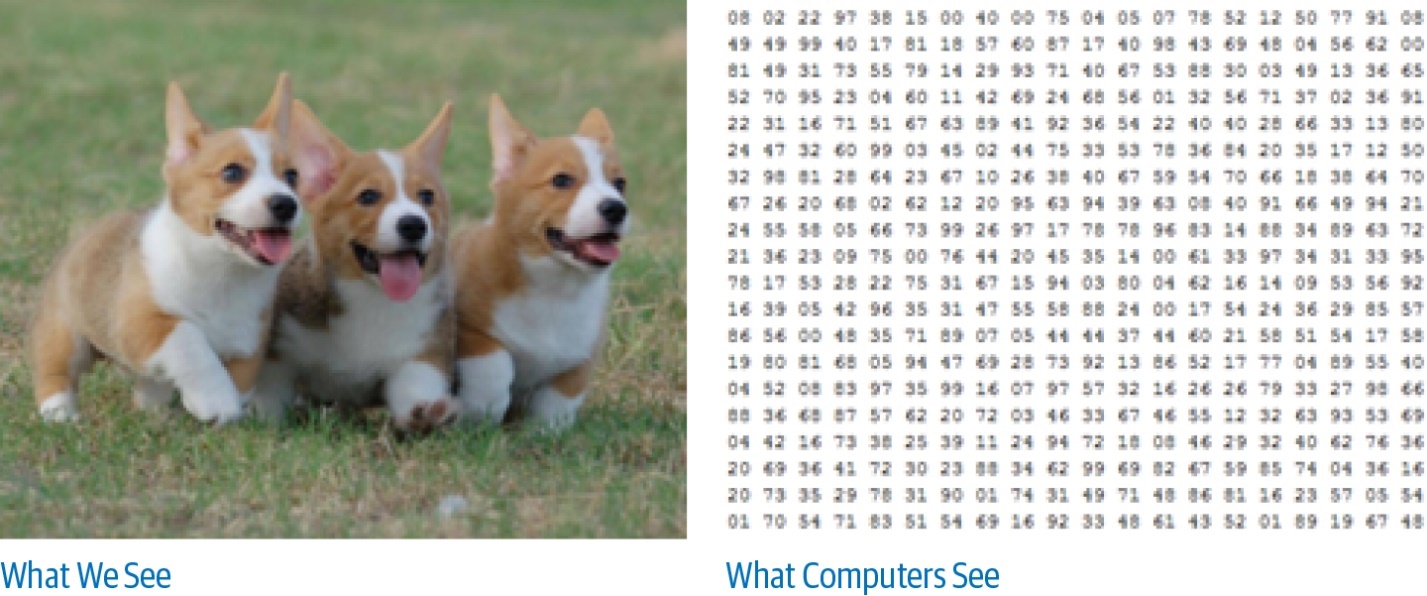
Step 2- Feature Engineering

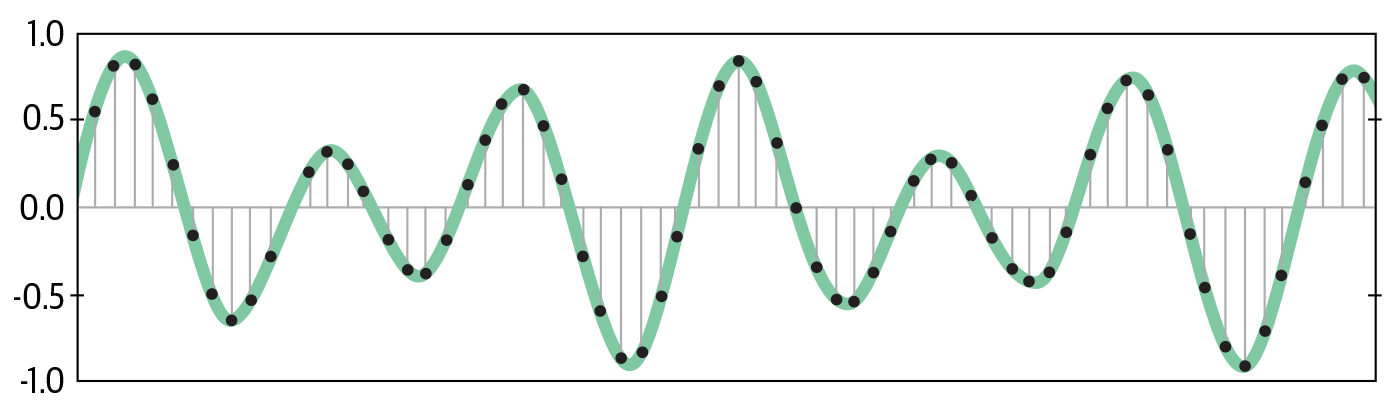
REPRESENT EACH WORD AS A NUMBER- conversion of raw text to a suitable numerical form



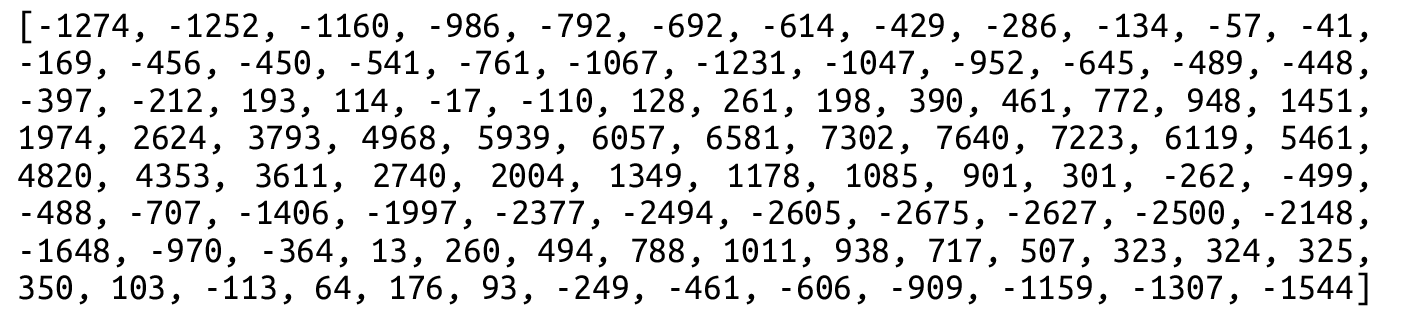
Image/ Video-------- > text



Speech- Wave sample and record its amplitude( height)



Numerical Vector



|  |
| --- |
|  |

## Bag Of Words

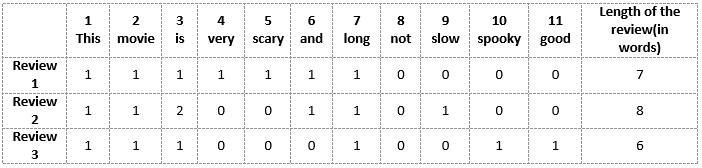
Capture how often a word occurs in a document i.e. the counts or the frequency

- sentence as a bag of words vector -word frequency vector, because it only counts the *frequency* of words

* 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:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/BoWBag-of-Words-model-2.png)

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]

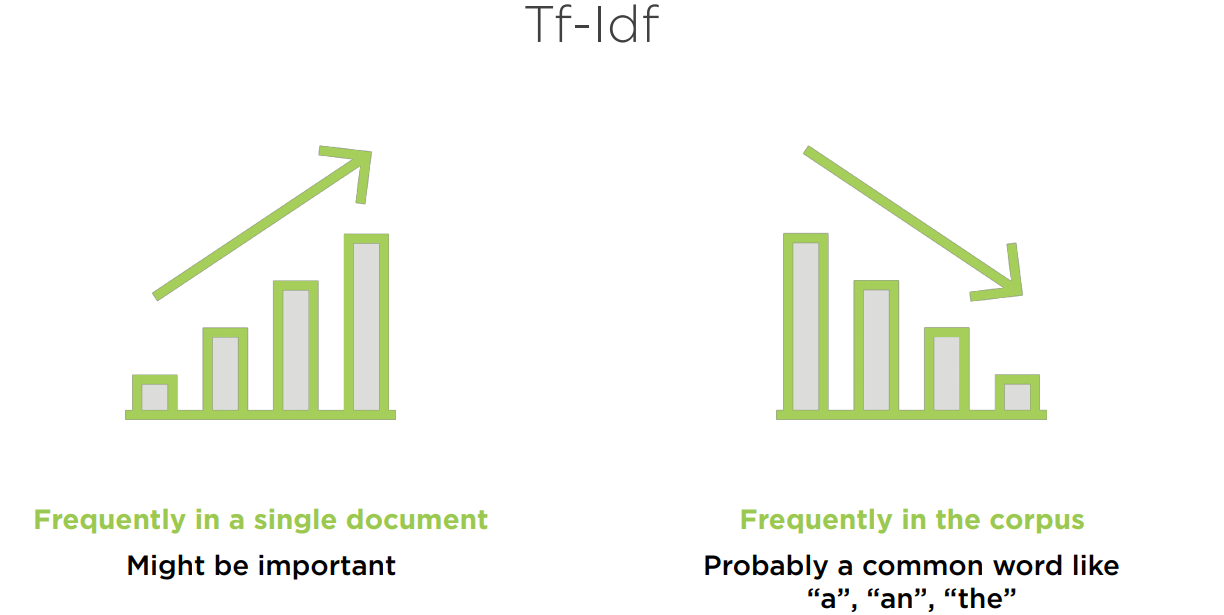
Bag of Words just creates a set of vectors containing the count of word occurrences in the document

Drawbacks of BOW

* Large vocabulary - enormous feature vectors
* Unordered - lost all context
* Semantics and word relationships lost

## TF-IDF- Captures how often a word occurs in a document as well as the entire corpus

It aims to quantify the importance of a given word relative to other words in the document and in the corpus



Explanation

*if a word*w*appears many times in a document*d*i but does not occur much in the rest of the documents*d*j in the corpus, then the word*w*must be of great importance to the document*d*i. The importance of*w*should increase in proportion to its frequency in*d*i, but at the same time, its importance should decrease in proportion to the word’s frequency in other documents*d*j in the corpus. Mathematically, this is captured using two quantities: TF and IDF. The two are then combined to arrive at the*TF-IDF score

TF (term frequency) measures how often a term or word occurs in a given document

TF(t,d)=(Number of occurrences of term t in document d)/(Total number of terms in the document d)=

IDF (inverse document frequency) measures the importance of the term across a corpus

*stop words like is, are, am, etc., are not important, even though they occur frequently. To account for such cases, IDF weighs down the terms that are very common across a corpus and weighs up the rare terms*

IDF(t)=log (Total number of documents in the corpus)/(Number of documents with term t in them )

TF-IDF score = TF \* IDF

|  |  |
| --- | --- |
| D1 | Dog bites man. |
| D2 | Man bites dog. |
| D3 | Dog eats meat. |
| D4 | Man eats food. |

V= [dog,bites,man,eats,meat,food]

Total number of docs in corpus= 4

D1= Dog bites man

**TF= no of times the word appears in the document/total number of words in the document**



Dog= 1/3= 0.33

Bites= 1/3= 0.33

Man= 1/3= 0.33

**IDF= total no of documents in the corpus/number of times the word appeared across documents**



Dog= IDF= log(4/3)=

Bites= IDF= log(4/2)= log(2)

Man= IDF= log(4/3)=log(1.33)

V= [Dog,bites,man,eats,meat,food]=



| **Word** | **TF score** | **IDF score** | **TF-IDF score** |
| --- | --- | --- | --- |
| dog | ⅓ = 0.33 | log2(4/3) = 0.4114 | 0.4114 \* 0.33 = 0.136 |
| bites | ⅙ = 0.17 | log2(4/2) = 1 | 1\* 0.17 = 0.17 |
| man | 0.33 | log2(4/3) =0.4114 | 0.4114 \* 0.33 = 0.136 |
| eats | 0.17 | log2(4/2) =1 | 1\* 0.17 = 0.17 |
| meat | 1/12 = 0.083 | log2(4/1) =2 | 2\* 0.083 = 0.17 |
| food | 0.083 | log2(4/1) =2 | 2\* 0.083 = 0.17 |
| *Table 3-2. TF-IDF values for our toy corpus* | | | |

The TF-IDF vector representation for a document is then simply the TF-IDF score for each term in that document. So, for D1 we get



| **Dog** | **bites** | **man** | **eats** | **meat** | **food** |
| --- | --- | --- | --- | --- | --- |
| 0.136 | 0.17 | 0.136 | 0 | 0 | 0 |

TF-IDF techniques are often used by search engines in scoring and ranking a document’s relevance given a keyword input. In Data Science, we can use it to get an idea of which words, and related information, are the most important in our text data

Important advantages - Frequency and relevance captured



## n-grams

**n-gram***refers to a sequence of n items from a given text*

I would love to visit the United States

Unigram- extracts one token at a time

extract two tokens at a time, it is called **bigrams**. If three tokens, it is called **trigrams**

extract pairs, triplets, quadruplets, and even quintuplets of tokens. These are called *n*-grams

“ice cream” as well as the “ice” and “cream”

“Mr Raghav”, “Mr”,”Raghav”