# NLP- Neuro Linguistic Programming

## What is a corpus?

A collection of text

## How to use text for classification?

ML algorithm needs some sort numerical feature vector data in order to perform classification task. So if we have to classify the text then we need to convert text or the Corpus into some feature vector

We need to create Bag of words, where each unique word in the text is represented by a number

* Normalize text –
  + Remove punctuations
  + Convert to lower case
  + Stop words,
  + apply stemming to get root words,
  + use BeautifulSoup to remove HTML tags,
  + Porter Stemming for Part of speech
* Convert Text to bag of words using CounterVectorizer
* Apply ML Algorithm like RF, Naïve Bayes

## What is bag of words?

* The bag-of-words model is a simplifying representation used in [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing) and [information retrieval](https://en.wikipedia.org/wiki/Information_retrieval) (IR).
* Also known as the [vector space model](https://en.wikipedia.org/wiki/Vector_space_model).
* In this model, a text (such as a sentence or a document) is represented as the [bag (multiset)](https://en.wikipedia.org/wiki/Multiset) of its words, disregarding grammar and even word order but keeping [multiplicity](https://en.wikipedia.org/wiki/Multiplicity_(mathematics))
* The bag-of-words model is commonly used in methods of [document classification](https://en.wikipedia.org/wiki/Document_classification) where the (frequency of) occurrence of each word is used as a [feature](https://en.wikipedia.org/wiki/Feature_(machine_learning)) for training a [classifier](https://en.wikipedia.org/wiki/Statistical_classification)

Example:

Text= “U dun say so early hor... U c already then say...”

Bag of words

(0, 4073) 2

(0, 4638) 1

(0, 5270) 1

(0, 6214) 1

(0, 6232) 1

(0, 7197) 1

(0, 9570) 2

Where say is number 9570 that appears twice in text

## What is Tokenization?

Tokenization is a term used to describe the process of converting the normal text strings in to a list of tokens (words)

## What is vectorization?

Process to convert a text as a list of tokens (lemmas) into a vector that machine learning models can understand. This is done Using **CounterVectorizer,** which will convertcollection of text documents to a matrix of token counts

There are three steps using the bag-of-words model:

1. Count how many times does a word occur in each message (Known as term frequency)
2. Weigh the counts, so that frequent tokens get lower weight (inverse document frequency)
3. Normalize the vectors to unit length, to abstract from the original text length (L2 norm)

## What is TF – Term Frequency?

TF is the measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

**TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).**

## What is TF -IDF- Term Frequency – Inverse Document Frequency?

TF-IDF is the measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

**IDF(t) = log\_e(Total number of documents / Number of documents with term t in it).**

## What id TF-IDF weight?

* tf-idf weight is a weight used in information retrieval and text mining.
* Ttf-idf weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus.
* The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.
* Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

Example:

Consider a document containing 100 words wherein the word solar appears 3 times.

The term frequency (i.e., tf) for Solar is then (3 / 100) = 0.03.

Now, assume we have 10 million documents and the word Solar appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as log(10,000,000 / 1,000) = 4.

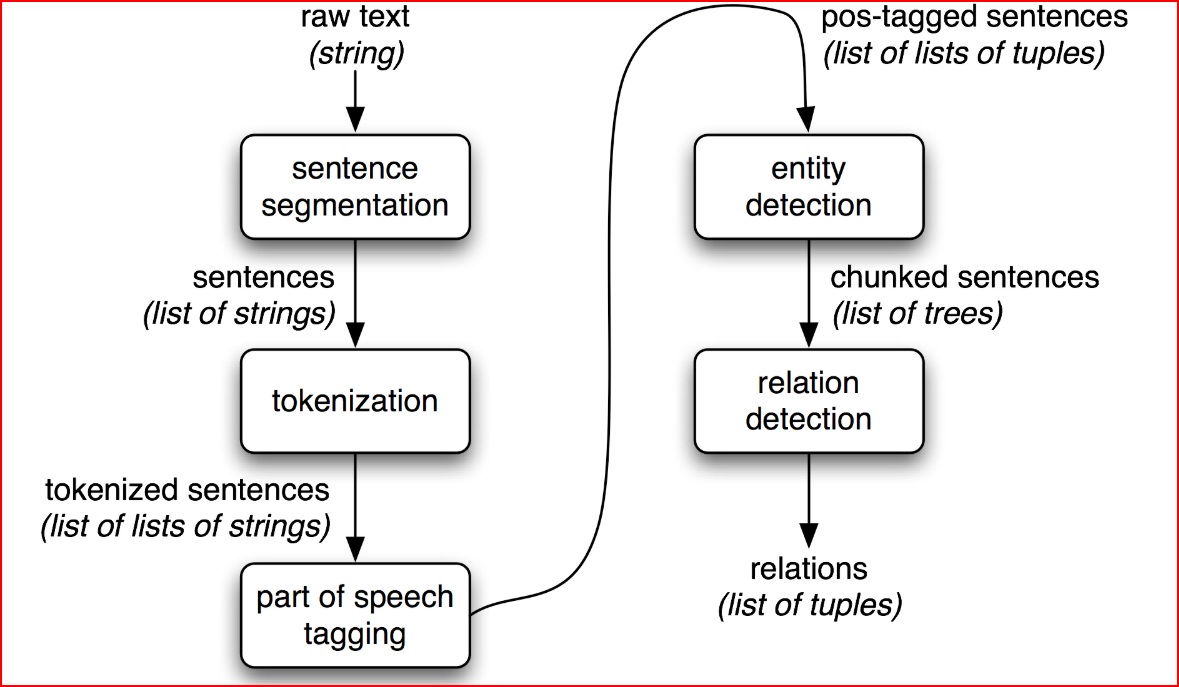
Thus, the Tf-idf weight is the product of these quantities: 0.03 \* 4 = 0.12.

## What is Collocation and Bigram?

* A **collocation** is a sequence of words that occur together unusually often.
* red wine is a collocation, whereas the maroon wine is not.
* A characteristic of collocations is that they are resistant to substitution with words that have similar senses; for example, maroon wine sounds definitely odd.

To get a handle on collocations, we start off by extracting from a text a list of word pairs, also known as **bigrams**. This is easily accomplished with the function bigrams():

# Information Extraction



* Raw text of the document is split into sentences using a sentence segmenter,
* Each sentence is further subdivided into words using a tokenizer.
* Each sentence is tagged with part-of-speech tags, which will prove very helpful in the next step, **named entity detection**.
* In Named Entity detection step, we search for mentions of potentially interesting entities in each sentence.
  + basic technique for entity detection is **chunking**, which segments and labels multi-token sequences. The smaller boxes show the word-level tokenization and part-of-speech tagging, while the large boxes show higher-level chunking. Each of these larger boxes is called a **chunk**.
* Finally, we use **relation detection** to search for likely relations between different entities in the text.