Vectorization

# What is vectorization?

In Natural Language Processing (NLP), text vectorization is a process of converting text and words and their associated features into a meaningful vector (array) of numbers. These vectors are also known as **Word Embeddings.**

Word Embeddings help us to perform basic NLP tasks such as text classification, document clustering, feature extraction and information retrieval along with other higher degree tasks such as text summarisation and similarity metrics.

# The need for vectorization

Machine learning algorithms operate on a numeric feature space, expecting input as a two-dimensional array where rows are instances and columns are features. Natural Language Processing (NLP) deals with the study and analysis of natural languages in their raw text form.

To perform machine learning on text, we need to transform our documents into vector representations such that we can apply numeric machine learning. This process is called feature extraction or more simply, vectorization, and is an essential first step toward language-aware analysis.

# How do we vectorize text?

There are multiple variations of word embeddings such as Bag of Words, TF-IDF, GloVe, Word2Vec, Doc2Vec, ElMo and BERT. The simplest one is the **Bag of Words** model and other models are often built on top of it.

## Step 1: Text Preprocessing

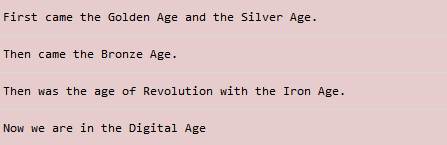
Text carries a lot of noise with it, in the form of redundant or unnecessary words, symbols and other lexicons. These must be removed to make our feature set more concise and reduce the dimensions of each feature.

* The first step in text preprocessing is to convert every text to lowercase and remove lexicons such as symbols and punctuations.
* We then remove useless words that add no additional information to the sentence. These are known as **stopwords** and they include:
  + **Determiners** – Determiners tend to mark nouns where a determiner usually will be followed by a noun examples: the, a, an, another
  + **Coordinating Conjunctions** – Coordinating conjunctions connect words, phrases, and clauses examples: for, an, nor, but, or, yet, so
  + **Prepositions** – Prepositions express temporal or spatial relations examples: in, under, towards, before
* Certain words exist in their inflectional or derivative forms. All these words need to be transformed into their base or root form. This process of converting words to a common base form is known as **Stemming**. Example, stemming converts ‘playing’, ‘playful’ and all other forms to ‘play’.
  + Stemming follows a rule-based approach and comes under the umbrella of *rule-based NLP.*
  + It uses a set of hard and defined rules to reduce a word to its base form. These rules are based on the suffixes of the words. The rules for the Snowball Stemmer can be found at [here](http://snowball.tartarus.org/algorithms/porter/stemmer.html).
  + While this works for some cases, in most cases it does not.

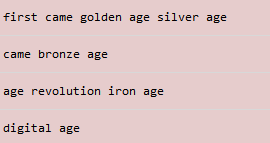
## Step 2: Generating a Bag of Words

A Bag of Words is a simple word embedding model where words are mapped to their frequency in each document. It is called a “bag” of words because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

Let us assume we have 4 documents –



After preprocessing, we obtain the following:



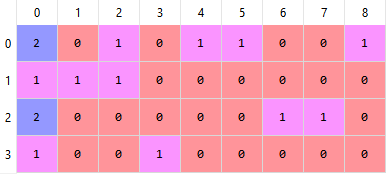
Our dictionary of features consists of all the unique words across all documents. These words help us in creating the vector for each document. For the given example, our dictionary of features is:



We can convert each sentence into a Bag of Words by creating a matrix known as the **Document Term Matrix**, where we have documents as rows, and feature names as columns. To prepare a Document-Term matrix, we find the frequency of every term in our dictionary in our document and create a vector out of these counts.

Hence, the value for each cell (i,j) is given by the frequency of term *j* in document *i.*

After performing these steps, we obtain the following matrix.



The first row corresponds to document 1, second to document 2 and third to document 3.

The following matrix obtained is known as a Bag of Words model for the given set of documents.

# How can we improve the Bag of Words Model?

The term frequency of a document does not always give us a better understanding of the features. It is required to associate these features with an importance factor that helps us gauge how essential or relevant these features are in a document.

The Bag of Words model is too simple and cannot represent these weights. Common words that occur across all documents will have an extremely high term frequency (like the word “age” in the above example). We want words that occur rarely to have a higher importance since it is these words that help us differentiate one document from another (like the words “iron” or “golden”).

This can be done by converting the matrix into a **TF-IDF (Term Frequency – Inverse Document Frequency)** matrix.

# How does TF-IDF help?

TF-IDF normalises the matrix by scaling down the frequencies of words that commonly occur across all documents and scaling up the frequencies of words that occur less frequently. Put simply, **the higher the TF-IDF score, the rarer the term and vice versa**.

TF-IDF was invented for document search and information retrieval. It works by increasing proportionally to the number of times a word appears in a document but is offset by the number of documents that contain the word. So, words that are common in every document, such as this, what, and if, rank low even though they may appear many times, since they do not mean much to that document.

This method can be used to answer questions like **how important a term** is in a document and which **terms have the highest relevance** in a document.

# How do we calculate TF-IDF?

There are two parts to calculating the TF-IDF score of a word.

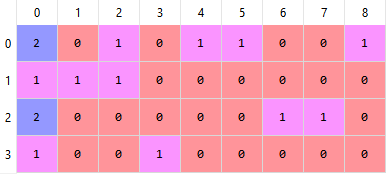
## Step 1: Calculating Term Frequency, TF:

Term Frequency gives us how frequently a word appears in the document. This is already calculated when we first convert our documents into a Bag of Words model.

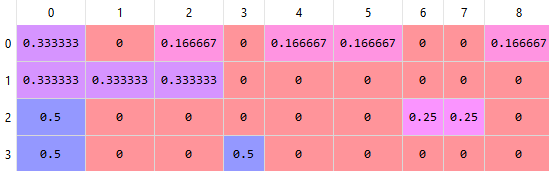
We further normalise these frequencies by scaling them and dividing by the total number of words in the document.



After doing the following, we convert

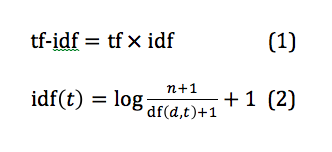


to

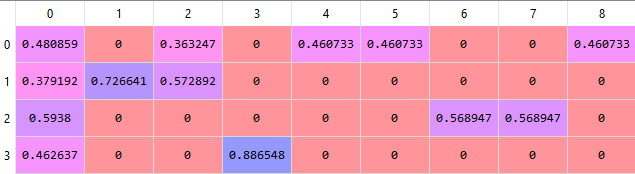


## Step 2: Calculating Inverse Document Frequency, IDF

Inverse Document frequency (IDF) is used for finding out importance and relevance of a word. It gives us how important a term is across all documents. It is a multiplier which is multiplied with the TF score to obtain a TFIDF score.



After doing this, we obtain the following matrix of features.

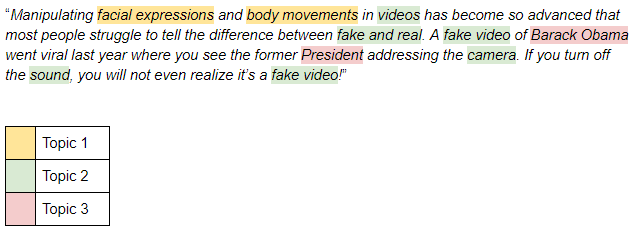


Topic Modelling

Topic Modelling in Natural Language Processing is an unsupervised learning algorithm that is used to extract topics (which are represented as a collection of words) from a collection of documents and hence is able to look for patterns or structures within them.

Topic Modelling is different from rule-based text mining approaches that use regular expressions or dictionary-based keyword searching techniques. It is an unsupervised approach used for finding and observing the bunch of words (called “topics”) in large clusters of texts.

Topics can be defined as “a repeating pattern of co-occurring terms in a corpus”. A good topic model should result in – “health”, “doctor”, “patient”, “hospital” for a topic – Healthcare, and “farm”, “crops”, “wheat” for a topic – “Farming”.



It is used for dimensionality reduction, where a term-document matrix can be converted into topic-document matrix which can sometimes be used to reduce the size of the feature matrix by more than 60%. It is also used in other Language Processing tasks such as text classification and building recommendation engines.

The concepts of Linear Algebra are used to perform topic modelling (although other methods do exist)

# LSA – Latent Semantic Analysis

Latent Semantic Analysis (sometimes also known as Latent Semantic Indexing, LSI) uses linear algebra to perform topic modelling. Namely, it uses Truncated Singular Value Decomposition(T-SVD) performed on the Document-Term matrix.

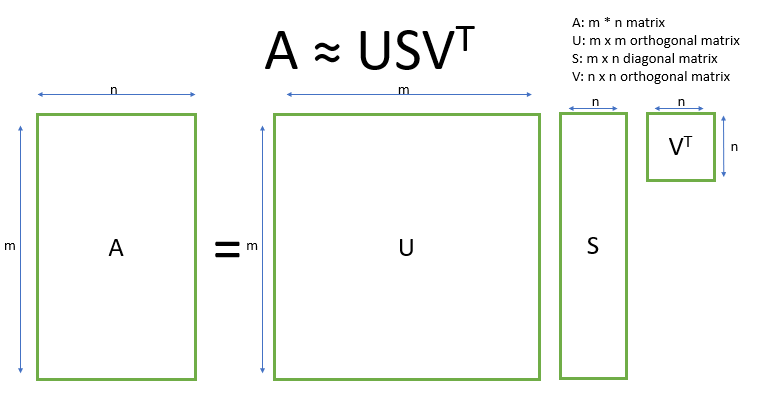
It has only 1 parameter ‘*k’,* which specifies the number of topics to extract from the given documents.

## How does it work?

1. The first step is to create a Term-Document matrix.
   1. We perform text preprocessing and strip each document of stopwords and punctuations or symbols
   2. We then lemmatize these words and convert derivative and inflectional forms of words to their root form
   3. We create a corpus of all the unique words spread across all documents to create our feature space
   4. For every document, we find the count of all feature (or term) in that document and store it as a vector. This helps us form the feature vectors for every document.
   5. We normalise these features by performing Tf-iDF normalisation on these vectors to counter the weights of frequently occurring words and thus form our Document-Term matrix.
   6. We take its transpose, also known as the **Term-Document Matrix**

The number of Documents is *m* and the number of features is *n*, making this matrix an (m x n) matrix.

1. We now perform SVD, Singular Value Decomposition
   1. We know that SVD is used to factorise or decompose a matrix into 3 smaller matrices, namely U, S and V. They are related to the matrix (say, A) as –



* 1. In truncated SVD, we take only the *t* column vectors of U and the *t* row vectors of V\* corresponding to the *t* largest singular values in S. The rest of the matrix is not needed. We hence reduce to dimensions of our features.

