# Advanced Data Mining and Machine Learning
# *Text Mining*

Academic Year 2022-2023

Alessandro Renda

*Slides adapted from a collection by professor Pietro Ducange*

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

IN SUPREMÆ DIGNITATIS 1343

UNIVERSITÀ DI PISA

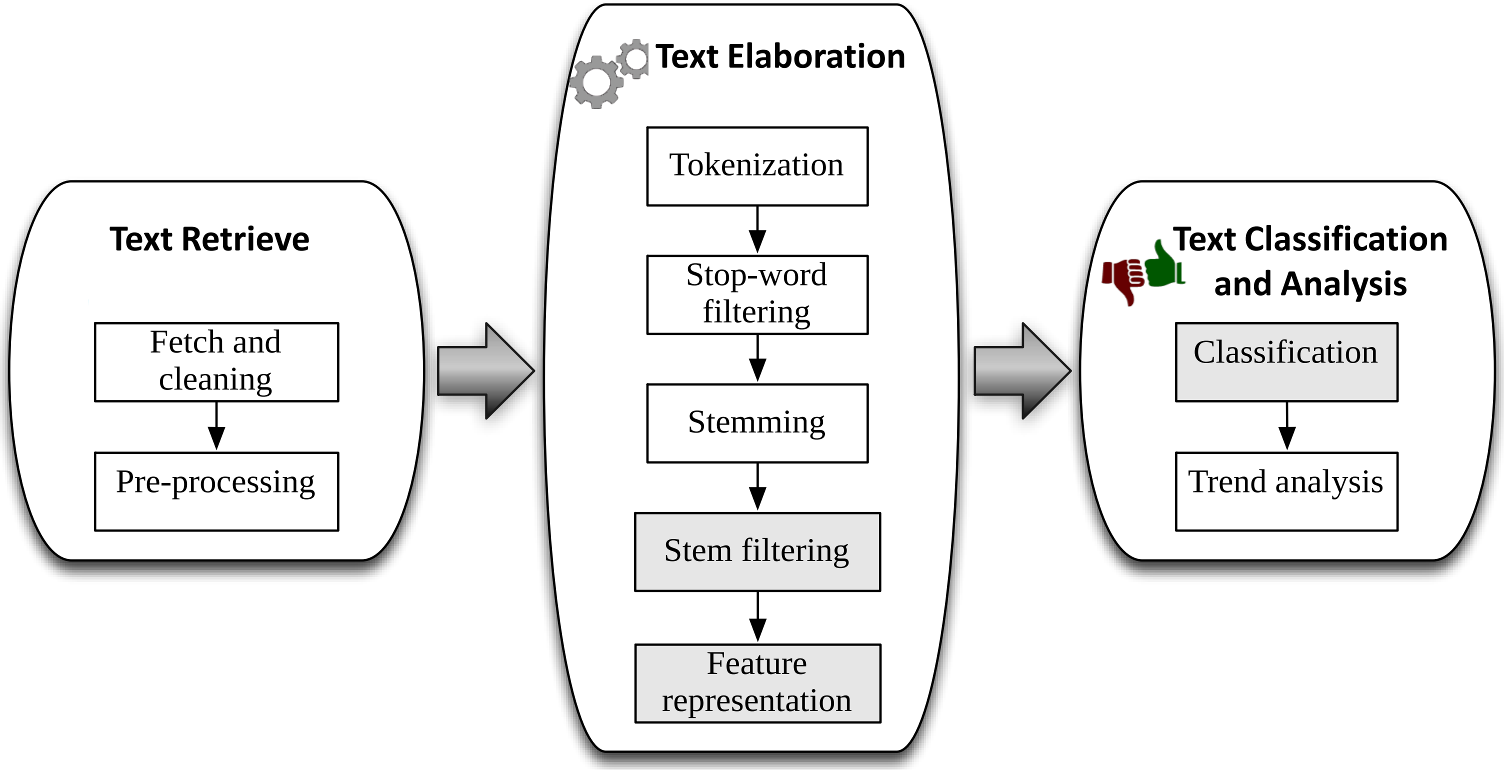
# How can we extract information from TEXTs?

- Text mining refers to the process of ***automatic extraction*** of meaningful information and knowledge from ***unstructured text***;

- ***Text Mining (TM)*** encompasses ***data mining (DM)***, ***machine learning (ML)***, ***statistics***, and ***Natural Language Processing (NLP)***;

- The main difficulty in text mining is caused by the ***vagueness*** of natural language:
  - ***people***, unlike computers, are perfectly able to ***understand*** idioms, grammatical variations, slang expressions, or to contextualize a given word;
  - conversely, ***computers*** have the ability, lacking in humans, to quickly ***process*** large amounts of information.

# Text Classification Platform (BOW)




- The platform is completely general. We will describe it relying on a case study of tweets classification

# Fetch, Cleaning and Preprocessing

- Streams of text can be fetched from different sources, such as a Micro Blogging Site, based on some ***search criteria*** (e.g., keywords, time or location of posting, hashtags);

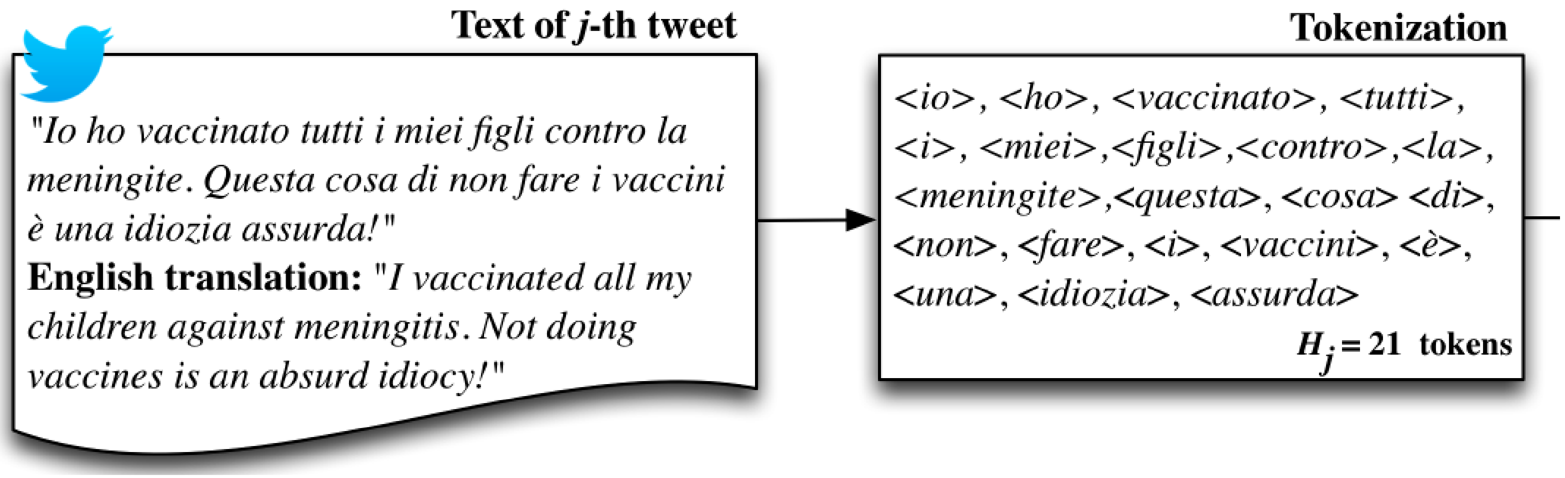
- Raw text must be cleaned. For example, we may need to discard:
  - i) ***duplicate*** texts (possibly fetched in different searches)
  - ii) text written in ***languages different*** from the one taken into consideration;

- Text can be preprocessed by applying a ***Regular Expression*** filter, in order to extract ***only the actual text*** and remove all ***useless meta-information***, such as links, hashtags, timestamp, and emoticon.

# Tokenization

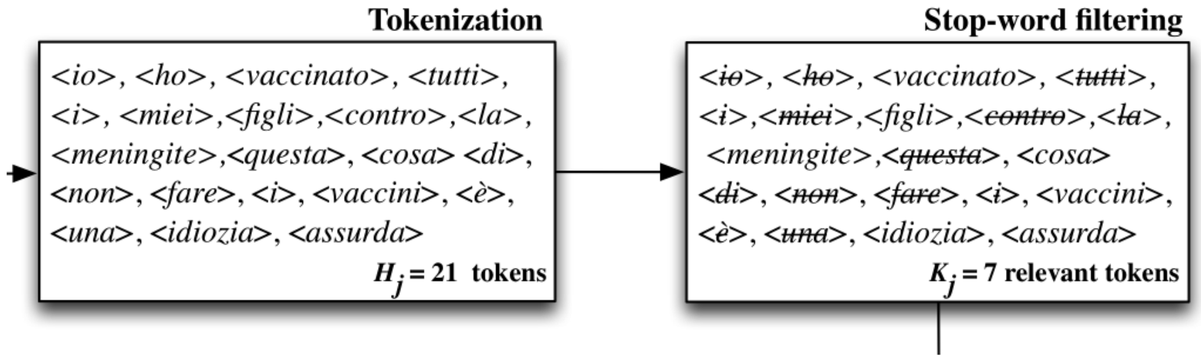
- Tokenization consists in transforming a stream of characters into a stream of ***processing units*** called *tokens*, e.g., words;

- Thus, during this step, after removing punctuation marks, non-text characters and special symbols (e.g., accents, hyphens), each text is represented as a set of words.



# Stop-word Filtering

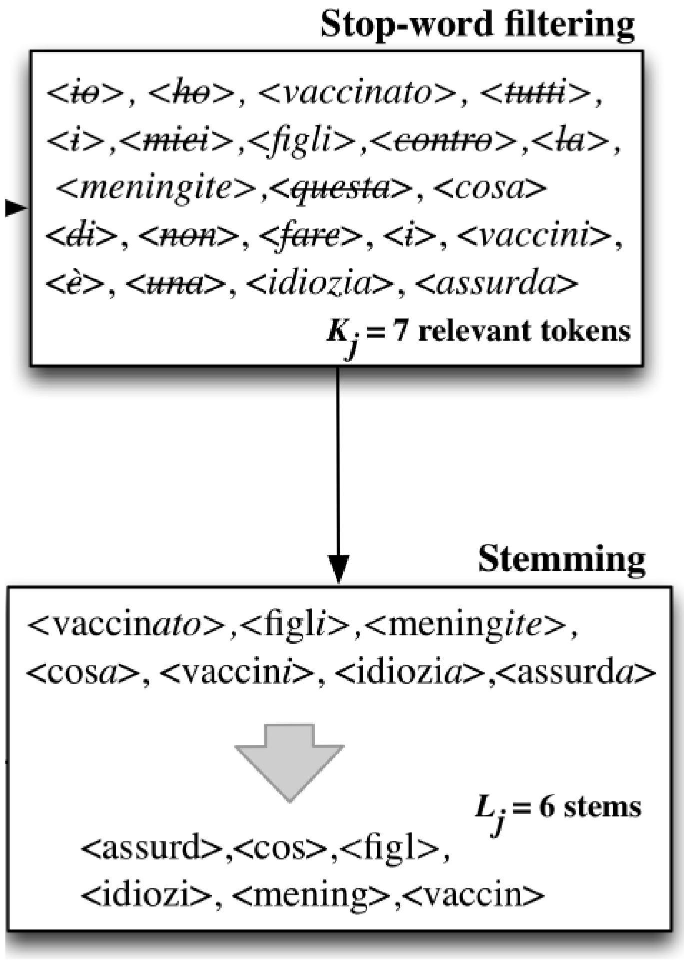
- This step consists in removing ***stop-words***, i.e., words providing little or no useful information to the text analysis, and can hence be considered as ***noise***;
- Common stop-words include ***articles***, ***conjunctions, prepositions, pronouns***, etc;
- Other stop-words are those typically ***appearing very often*** in sentences of the considered language (***language-specific stop-words***), or in the particular context analyzed (***domain-specific stop-words***);
- At the end of this step, each text is cleaned from stop-words, and thus reduced to a sequence of ***relevant tokens***.



# Stemming

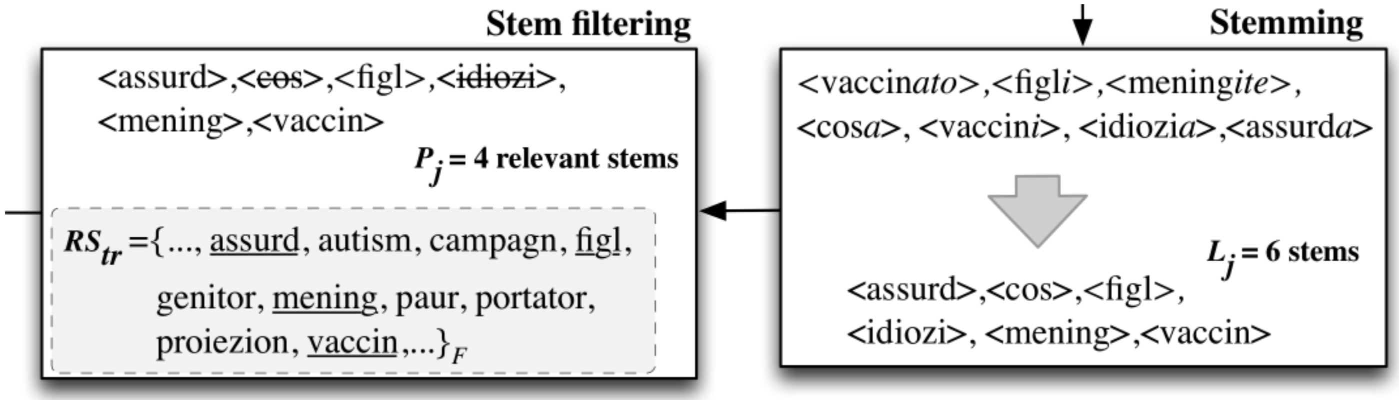
- ***Stemming*** is the process of reducing each token (i.e., word) to its stem or ***root form***, by removing its suffix, in order to group words having closely related semantics;

- Hence, at the end of this step each text is represented as a sequence of stems.



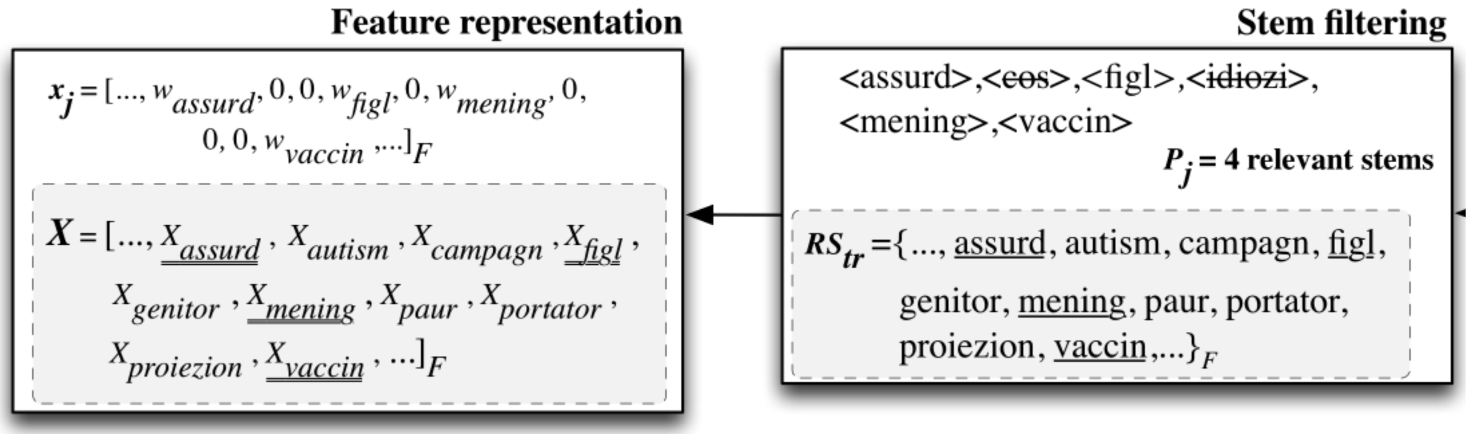
# Stem Filtering

- During this stage, the number of stems of each text are reduced, by removing ***noisy stems*** and maintaining only the most relevant ones;
- Thus, each text is cleaned from stems not belonging to the set of ***relevant*** stems.
- The set of relevant stem can be provided as a ***vocabulary*** or identified through a ***supervised learning stage***, using the corpus of training documents.

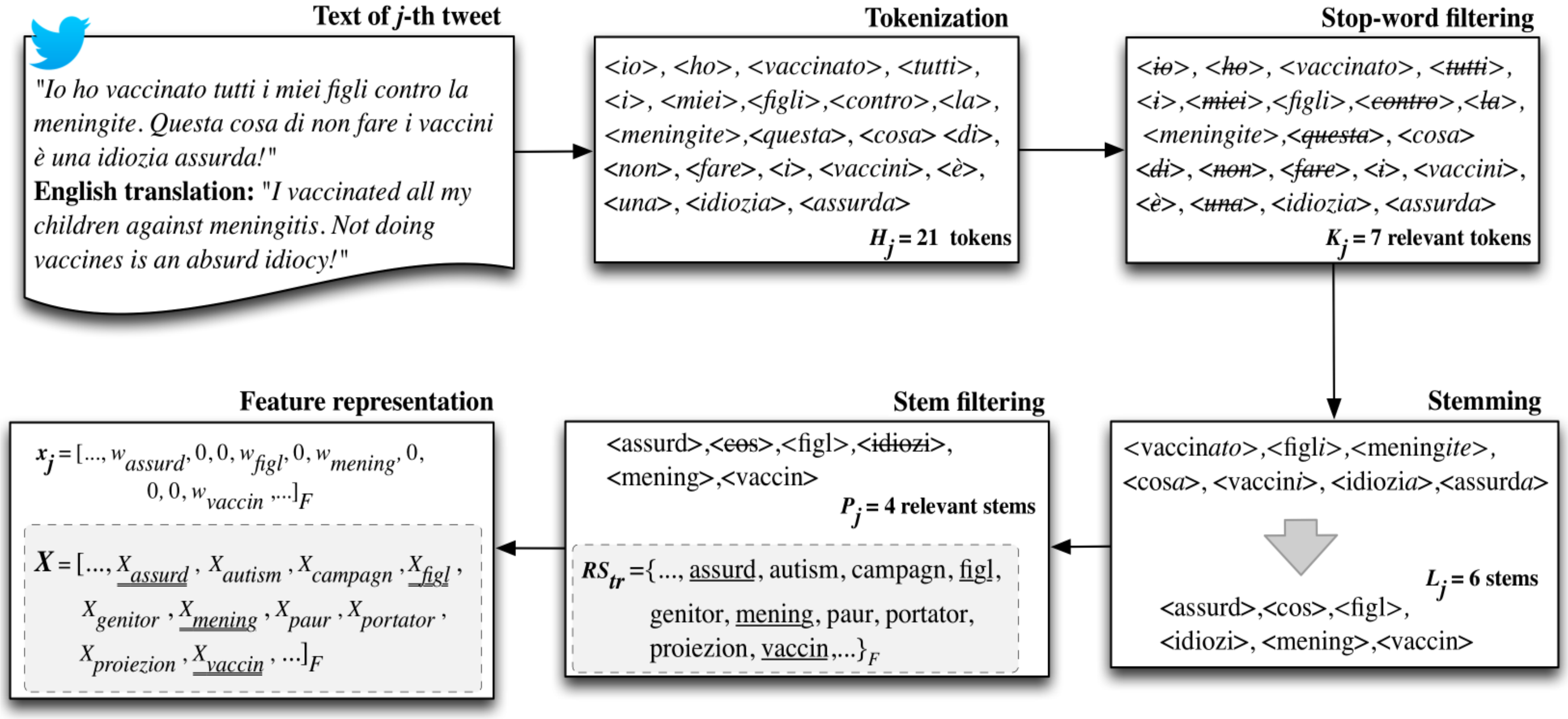


# Feature Representation

- ***Feature representation*** consists in building for each text the corresponding vector of numeric features, i.e., in order to represent all the texts in the same *F*-dimensional feature space;
- The set of *F* features corresponds to the set of relevant stems;
- Each text is thus associated with a ***vector*** of binary or numeric ***features***

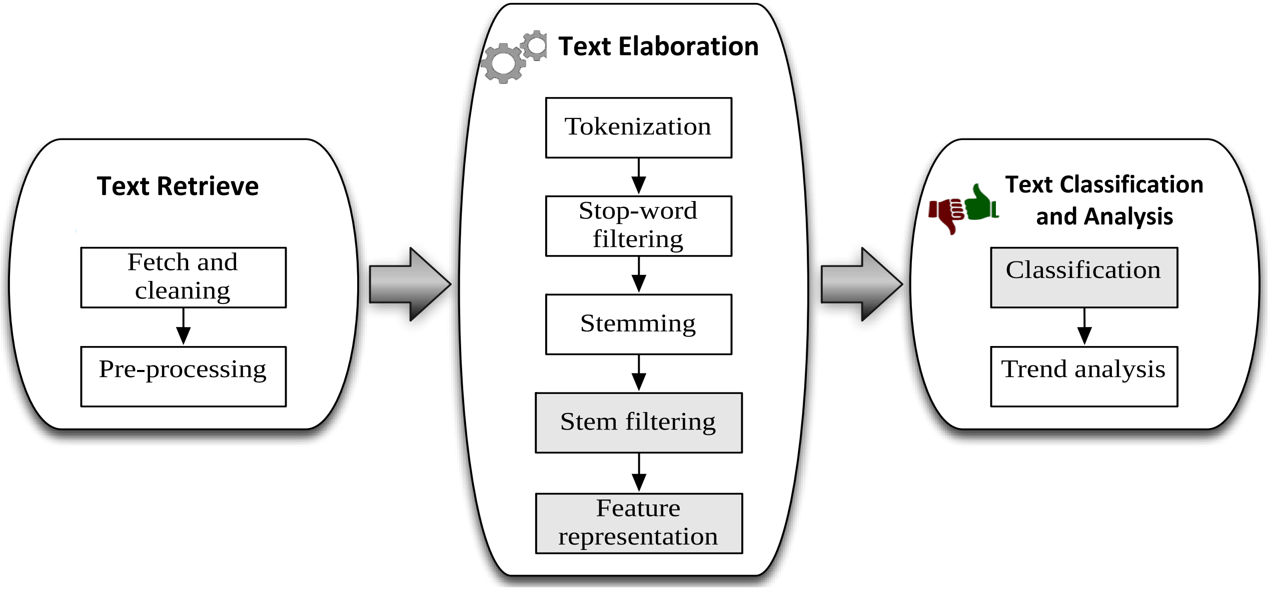


# The Entire Text Elaboration Process



# The Supervised Learning Stage

- We need to:
  1. Identify the set of ***relevant stems***;
  2. Compute the ***weights*** associated with each of them;
  3. Set the values of the ***parameters*** of the supervised ***classification*** model.




- To this aim, we need a collection of $N_{tr}$ labeled texts as training dataset;
- Each text of the training set undergoes the text mining steps: *tokenization*, *stop-word filtering*, and *stemming;*
- The complete set of $Q$ stems is generated ***putting together*** all the stems extracted from the set of training text after the stemming step.

# From text to numbers

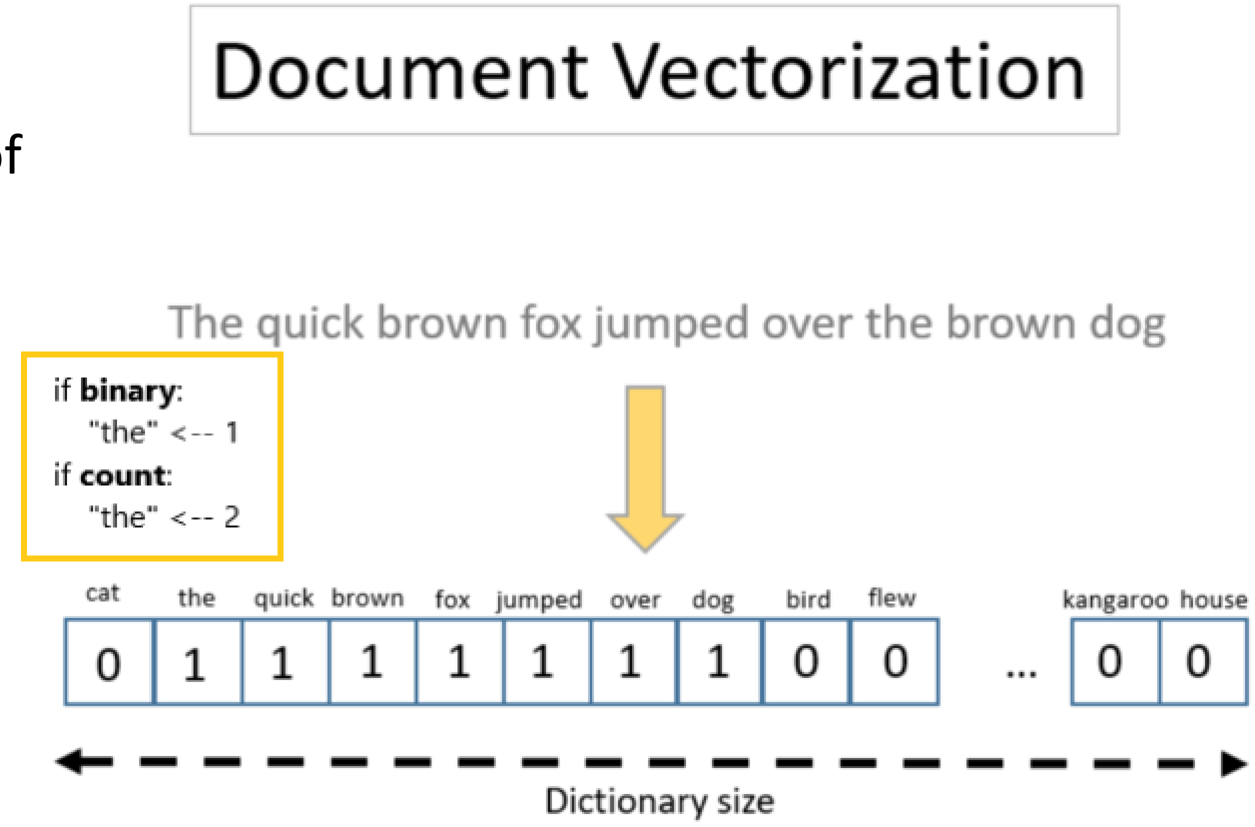
- To classify a text, it is necessary to ***transform*** it into a ***numerical vector***, which is then handled by a classification model

- The *Bag-Of-Word* (***BOW***) representation is one of the ***simplest*** and ***most used*** technique for text representation:
  - The set of features is composed by the words of the *vocabulary* inferred from the training set
  - Binary, integer or real representation

- Word Embedding (***WE***) methods are also widely used for text representation:
  - Words in a vocabulary are transformed into vectors of continuous real numbers

# BOW representation

- The set of F features (vocabulary) corresponds to the set of relevant stems
- Each text is thus associated with a vector of binary or numeric features



# TF-IDF: weighting scheme for BOW representation

Let $t$ be a term (e.g., word), $d$ a document (e.g., email), and $D$ the corpus (i.e., collection of documents):

- **Term Frequency (TF)** counts the number of times a term $t$ (word) appears in a document $d$ (i.e., $f_{t,d}$) adjusted by the length of the document (number of all words $t'$ in document $d$).

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

- **Inverse Document Frequency (IDF)**: counts the number of documents an individual term $t$ appears over all documents N (inverse fraction of the documents that contain the word, and evaluate the logarithm).

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

- **Term Frequency - Inverse Document Frequency (IDF)**: product of **TF** and **IDF**; weights down common words like "*the*" and gives more weight to rare words like "*software*".

$$TfIdf(t,d,D) = tf(t,d) \cdot idf(t,D)$$

# BOW representation drawbacks

- Using BOW representation (with or without TFIDF) results in ***large sparse vector*** for describing a text.

- This is mainly due to ***vast vocabularies*** that lead to represent  a text by a large vector comprised mostly of zero values.

**Document 1**

The quick brown fox jumped over the lazy dog's back.

**Document 2**

Now is the time for all good men to come to the aid of their party.

| Term | Document 1 | Document 2 |
|---|---|---|
| aid | 0 | 1 |
| all | 0 | 1 |
| back | 1 | 0 |
| brown | 1 | 0 |
| come | 0 | 1 |
| dog | 1 | 0 |
| fox | 1 | 0 |
| good | 0 | 1 |
| jump | 1 | 0 |
| lazy | 1 | 0 |
| men | 0 | 1 |
| now | 0 | 1 |
| over | 1 | 0 |
| party | 0 | 1 |
| quick | 1 | 0 |
| their | 0 | 1 |
| time | 0 | 1 |

*Image extracted from https://www.quora.com/What-is-the-bag-of-words-algorithm*

# Word Embeddings Representation

Check this example
https://projector.tensorflow.org/

- Words are represented by ***dense vectors***
- A vector represents the projection of the word into a ***continuous vector space***
- ***Semantically*** similar words have similar vectors
- A word embedding can be learned as part of a ***deep learning model***
- A text can be represented using a vector containing of the ***average*** values of the vectors representing each of its ***relevant tokens***.

$$\overset{W_1}{\begin{bmatrix} W_{11} \\ W_{12} \\ \\ \\ W_{1n} \end{bmatrix}} + \overset{W_2}{\begin{bmatrix} W_{21} \\ W_{22} \\ \\ \\ W_{2n} \end{bmatrix}} + \cdots + \overset{W_n}{\begin{bmatrix} W_{n1} \\ W_{n2} \\ \\ \\ W_{nn} \end{bmatrix}} = \overset{D}{\begin{bmatrix} \dfrac{W_{11}+W_{21}+\ldots+W_{n1}}{n} \\ \\ \\ \dfrac{W_{1n}+W_{2n}+\ldots+W_{nn}}{n} \end{bmatrix}}$$

# Word Embedding Learning Methods

- Three popular examples of methods of learning word embeddings from text include:
  - ***Word2Vec***: based on Neural Networks (https://code.google.com/archive/p/word2vec)
  - ***GloVe***: based on matrix factorization (https://nlp.stanford.edu/projects/glove/)
  - ***FastText***: based on Neural Networks (https://fasttext.cc/), created by Facebook

- A number of pre-trained Word Embeddings are available of the websites shown above.

- *Traditional* Word Embeddings (2013) have had a major impact in the field of text mining, but they still have some limits.
- Transformers (2017) and transformer-based language models further improved the state of the art, by enabling the achievement of unprecedented performance in NLP tasks.

# Useful References


- Weiss, Sholom M., et al. *Fundamentals of predictive text mining*. Springer, 2015.
- Mikolov, Tomas, et al. "*Distributed representations of words and phrases and their compositionality.*" Advances in neural information processing systems. 2013.
- Pennington, Jeffrey, et al. "*Glove: Global vectors for word representation*." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.
- Bojanowski, Piotr, et al. "*Enriching word vectors with subword information*." arXiv preprint arXiv:1607.04606 (2016).
- D'Andrea, Eleonora, et al. "*Monitoring the public opinion about the vaccination topic from tweets analysis.*" Expert Systems with Applications 116 (2019): 209-226.
- Ducange, Pietro, et al. "*An effective Decision Support System for social media listening based on cross-source sentiment analysis models.*" Engineering Applications of Artificial Intelligence 78 (2019): 71-85.
- Bechini, Alessio, et al. 2022. *A News-Based Framework for Uncovering and Tracking City Area Profiles: Assessment in Covid-19 Setting*. ACM Trans. Knowl. Discov. Data 16, 6, Article 125, 29 pages.