**Text Mining and Search**

# Contents

[Contents 1](#_Toc156757363)

[Intro 2](#_Toc156757364)

[Part of Speech tagging and NER (Named Entity Recognition) 4](#_Toc156757365)

[POS (Part of Speech) tagging 4](#_Toc156757366)

[NER (Name Entity Recognition) 4](#_Toc156757367)

[Natural Language Processing (NLP) 5](#_Toc156757368)

[Words Embeddings 5](#_Toc156757369)

[Vector semantics 5](#_Toc156757370)

[Count-based Models 6](#_Toc156757371)

[Predictive Models - Word2vec 7](#_Toc156757372)

[Statistical language models 7](#_Toc156757373)

[Introduction to Information Retrieval 8](#_Toc156757374)

[Indexing and data structures 9](#_Toc156757375)

[Basic IR (Information Retrieval) Models 10](#_Toc156757376)

[Boolean Model 10](#_Toc156757377)

[Vector Space Model 10](#_Toc156757378)

[Web Search 11](#_Toc156757379)

[Document Gathering 12](#_Toc156757380)

[Link Analysis 13](#_Toc156757381)

[Metasearch 14](#_Toc156757382)

[Recommender systems 14](#_Toc156757383)

[Evaluating Information Retrieval Systems 15](#_Toc156757384)

[Evaluating Recommender systems 16](#_Toc156757385)

[Test Mining Tasks 17](#_Toc156757386)

[Text classification 17](#_Toc156757387)

[Text Clustering 19](#_Toc156757388)

[Text Summarization 19](#_Toc156757389)

[Topic Modeling 21](#_Toc156757390)

# Intro

The text mining is a new and growing field that aims to extract meaningful information from the natural language of the text. It can be defined as the process that analyzes texts to extract useful information for specific purposes.

It is applied in:   
• Sentiment Analysis   
• Document Summarization   
• Hotel/News Recommendation   
• Text Analytics in Health Care, Financial Services, etc.

The difference from Data Mining, where information is in input data, often hidden, unknown, and difficult to extract, is that in Text Mining, the information to be extracted is clear and explicit in the text. This information is extracted in a form convenient for computer processing, making the human intermediary unnecessary. Information Retrieval has contributed to the birth of text analysis.

Problems in text mining:   
• Textual documents in an unstructured form are not directly accessible and usable by computers.   
• Dealing with huge collections of documents or text streams.   
• Unstructured data.   
• The natural language of the text contains ambiguity at the lexical, syntactic, semantic, or pragmatic level.

Tasks related to text mining:   
• Text Summarization: produces a concise representation of its input suitable for human consumption.   
• Information Retrieval: based on text analysis to find matches to a user query, it often relies on Text Summarization.   
• Content-Based Recommender Systems: textual content in a stream is personalized to the user's profile.   
• Text Classification: to assign natural language documents to predefined categories based on their content; it is applied in various applications (e.g., Sentiment Analysis) useful for ML.   
• Document Clustering: no predefined category, but it aims to find groups of documents that share similar topics.   
• Topic Identification and Tracking.

Mining structured information from texts means:   
• Entity Extraction: identifying linguistic constructions representing objects or entities (people, places, etc.).   
• Information Extraction: filling models with natural language input.   
• Learning Rules from Texts: extracting rules that characterize the content of the text itself.

There are two main branches of research in Text Mining:

• Predictive Analysis of Text: computational programs that automatically recognize or find particular concepts in a text space; examples include:

* Opinion Mining: automatic detection of the expressed opinion (positive or negative) in a text space.
* Sentiment/Affect Analysis: automatic detection of the emotional state of the author in a text space (usually, the set of emotional states is predefined).
* Bias Detection: automatic detection of the author's inclination towards a particular point of view (usually, predefined points of view).
* Information Extraction: automatic detection of the connection of a short sequence of words to an entity (person, location, role, etc.).
* Relation Learning: automatic detection of pairs of entities sharing a particular relationship.
* Text-driven Forecasting: monitoring incoming texts and predicting events and external trends.
* Temporal Summarization: monitoring incoming texts about an event and predicting when a sentence should be included in an ongoing summary of the event.

• Explanatory Analysis of Text: Developing a computational program that automatically discovers interesting and useful trends or patterns from a collection of texts. Examples include:

* Text Clustering
* Topic Modeling

A document is composed of several components:  
• Text  
• Structure (content, logic, and layout)  
• Other Media (images, sound, etc.)  
• Metadata (external attributes)

In particular, metadata is information about information. This information is cataloging and descriptive, structured in a way that allows pages to be correctly searched and processed by the computer. Metadata can be descriptive, concerning the characteristics of a document (author, title, etc.), or semantic, concerning the subject addressed in the document.

The initial phases of Text Processing are:  
1. Tokenization: A token is a sequence of characters that form a meaningful word after various processing. The goal of Tokenization is to divide a text stream into dense units of meaning (tokens). The methods used are usually Regular Expressions and Statistical Methods.  
2. Normalization: "Normalizing" words to make them uniform in terms of accents, case, punctuation, and typos.  
3. Stemming and Lemmatization: Lemmatization reduces flexible terms to a base form with the advantage of reducing the size of the analyzed dictionary. Stemming, on the other hand, reduces words to their roots (the risk is losing the precise meaning of words). The Porter algorithm is one of the most common stemming English algorithms.  
4. Stop words: Elimination of the most common words.

A common and useful representation of texts is the Bag-of-Words, a table that lists the words contained in the texts. If the table simply indicates presence/absence, it is called a Binary term-document incidence matrix. If, instead, the words are counted within each text, we have the term-document count matrix (each document is a count vector).

Zipf's Law (1949) describes the frequency of an event in a set (in this case, a word) according to its rank. Given a collection, it orders the words w in descending frequency order f(w) in the collection (with an increasing order of ranks). The product of the frequency of word usage and the rank order is approximately constant. Thus, the frequency of w, f(w), is proportional to 1/r(w):

Different collections have different values of K. This explicitly states that "head words" are highly recurrent but lack meaning, while "tail words" make up the majority of the vocabulary but occur rarely in documents.

Luhn Analysis (1958): not all words in a text describe the content with the same accuracy/informativeness. Luhn noted that the frequency with which some words appear in the text provides important indications about the meaning of words. Furthermore, the position of these words in sentences is another important parameter indicating the significance of the sentences.

The idea is to assign weights to terms representing a document. The discriminative power of significant words (Zipf's curve) shows that the ability of words to discriminate content in a document is highest in the intermediate position between the two cutoff levels.

As a result, lesser weights can be assigned to more common words (weighted terms) or eliminate them altogether (stop list) and still obtain meaningful terms by also eliminating the less frequent ones.

Based on Luhn's analysis, two factors have been proposed to assign weights to words:   
• Corpus-wise: some terms provide more information about the content of the document.   
• Document-wise: not all terms are equally important.

To measure these factors, two heuristics are used:

• TF (Term Frequency): Defined as the number of times the term t occurs in the document d. Usually, it is normalized with respect to the maximum occurrence. 

• IDF (Inverse Document Frequency): The document frequency df indicates the number of documents d in which the term t appears. This is an inverse measure of the informativeness of t. IDF is defined as the logarithm of N/df to penalize the effect of the same: 

These two heuristics combine into the tf-idf weight, better known as the weight scheme in Information Retrieval. This weight increases when the number of occurrences of the term in the document increases and when the rarity of the term in the collection increases. 

# Part of Speech tagging and NER (Named Entity Recognition)

To recognize sentences in a text, three approaches are possible:  
1. N-grams  
2. Identifying syntactic sentences using a POS (Part of Speech) tagger  
3. Storing the position of words in indices and using proximity operators in queries (e.g., search engines)

## POS (Part of Speech) tagging

Once all the preliminary phases of Text Processing are completed, POS tagging aims to identify a word in a text (corpus) with a tag corresponding to a specific part of speech (lexical class marker).

Firstly, a set of tags needs to be chosen, which can vary in size and purpose (the most commonly used is the Penn Treebank). The 9 traditional classes of words are Nouns, Verbs, Adjectives, Prepositions, Adverbs, Articles, Interjections, Pronouns, and Conjunctions.

Applications of POS tagging can be diverse, including grammatical analysis (parsing), word prediction in speech recognition, and machine translation.

Unfortunately, a word can correspond to several POS tags (tag ambiguity), so a fundamental problem is determining the POS tag for a particular instance of a word.

To disambiguate POS, there are different approaches:   
• Many words have a most likely tag.   
• Tags tend to occur regularly with other tags (conditional probabilities of words).   
• POS likelihoods can be used to estimate the likelihoods of phrases.

To perform POS tagging, there are two main approaches:

1. Rule-Based Tagging (e.g., EnCG ENGTWOL tagger)
2. Stochastic or Probabilistic tagging (Hidden Markov Model tagging)

## NER (Name Entity Recognition)

Named Entity Recognition techniques identify words that refer to names of interest in specific applications (such as people, companies, locations, product names). The text is mapped to identify names (task: most likely mapping?).

Two possible approaches:

* RULE-BASED: they use lexicons (lists of words and phrases to categorize names); they are also used to find identity names; these rules are developed manually (trial and error) or using ML techniques.
* STATISTICAL (Hidden Markov Model): they use a probabilistic model of words within and around an entity. Probabilities are estimated using training data (manually annotated texts). One approach is represented by Hidden Markov Models (HMM).

They solve the problem of ambiguity by using context, modeled using a generative model of a sequence of words. Markov Models describe a process as a collection of states and transitions between them.

Markov property: the next word in a sequence depends only on a small number of preceding words. Each state has a set of possible output values that, in turn, have probabilities.

In other words, each state is associated with a probability distribution over words (the most probable sequence of words for that category). To recognize entity names, the sequence of labels with the highest probability for the sentence is found.

The Viterbi algorithm implements a Markovian tagging process and can be used for recognition.

## Natural Language Processing (NLP)

It applies to language to provide knowledge from a string of text. It is based on:  
• Morphology (stemming, Stop Words, etc.)   
• Syntax (how words are connected to each other)   
• Semantics (what is their combined meaning)   
• Pragmatics (Context-Aware Applications)   
• Discourse   
• Inference

Natural Language Processing is designed to make human communication efficient, although it is full of ambiguity! Ambiguity is indeed the biggest problem to address. This can occur at the level of:  
• Word (ambiguous POS and/or sense)   
• Syntax   
• Anaphora   
• Presupposition

NLP applications are diverse, from Machine Translation to Dialog Systems, through information extraction. To teach a computer from a text, a generic pipeline is used:

1. Syntactic Parsing (grammatical analysis of the text)
2. Relation Extraction (identifies relationships between entities)
3. Logic Inference (converts blocks of text into a more formal representation)

NLP is distinguished in:  
• Better NLP -> Better Text Mining   
• Bad NLP -> NOT Bad text Mining, useful in particular contexts

There is an inverse relationship in tasks between their dependence on NLP and the scalability of the task. In general, simple models with high robustness and efficiency are sought.

# Words Embeddings

## Vector semantics

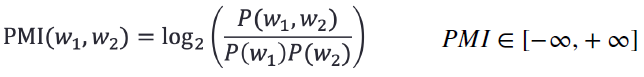
Word Embeddings refer to a set of techniques in Natural Language Processing (NLP) where words (or phrases) from the vocabulary are mapped to vectors of real numbers. Conceptually, it requires mathematical embeddings from a high-dimensional space to a much lower-dimensional vector space.

Models that generate this mapping include: • Count-based Models (Distributed Semantics models) • Predictive Models (Neural Networks models)

Usually, word similarity is not calculated through the term-document representation but through a matrix that presents words in both columns and rows. The key concept is that of "Context Words," which will be explored further.

There are 3 possibilities for representing words as vectors:

1. 1-Hot vectors, where each element represents a word. This is not very useful, as the resulting matrix is very sparse, and the number of dimensions is too high (increases linearly with the addition of words to the vocabulary). It also fails to capture any information about word similarity.
2. Distributed Representation, where each word in the vocabulary is associated with one of the k "context dimensions" representing the properties associated with words in the vocabulary.   
   Pros: In simpler (and restricted) scenarios, we can create a k-dimensional mapping by manually choosing contextual dimensions. This way, similar words have similar vectors, and the resulting matrix is much less sparse. We can add words to the vocabulary without increasing the matrix's dimensionality, and relationships between words are captured and maintained.   
   Cons: However, when text dimensions are very high, manually assigning vectors is practically impossible (for this reason, some algorithms have been developed to create models from large text corpora).
3. Window-based Co-occurrence Matrix, based on the distributional hypothesis, according to which: "Words which are similar in meaning occur in similar contexts" and "You shall know a word by the company it keeps." The central idea is to represent words in a context, defined as the simultaneous occurrence of different words. Various context granularities can be used: documents, sentences, phrases, n-grams. The most common way to create a word vector is still by counting the number of times a word occurs simultaneously with all other words. A matrix is generated, and the distinction between target words and context words is arbitrary (we are free to exchange their roles).

The most frequent words dominate the vectors. To solve this problem, a scheme such as:  
• TF-IDF  
• Pointwise Mutual Information (PMI): it indicates whether two words occur together more often than in the case of independence. Negative values can create computational difficulties, so they are replaced by zero in Positive Pointwise Mutual Information: 

However, PMI is biased towards rarer words. To overcome this, two solutions have been proposed, representing Weighted (P)PMI:

* Adjusting contextual probabilities by a factor α ∈ [0,1]; This helps PPMI raising probabilities of rarer words.
* Adding k-smoothing: adding a value k to each frequency in the term-context matrix.

## Count-based Models

Count-based models calculate statistics about the simultaneous occurrence of certain words with the words near them in very large text corpora and map these statistics into a small, dense vector for each word. These models "learn" the vectors through a dimensionality reduction of the term-context matrix, factorized into a matrix of words and features.

The most common count-based models are:   
• Latent Dirichlet Allocation (LDA) (suitable for topic modeling)   
• Singular Value Decomposition (SVD)

SVD reduces the dimensions of the co-occurrence matrix X using linear algebra methods. In particular, each row of the resulting matrix U is the dense k-dimensional representation of each word in the vocabulary.

Issues related to SVD include the change in dimensions of the matrix U whenever the corpus size changes, its sparsity, and the quadratic cost. To address these issues, hacks have been proposed, such as ignoring common words like names and conjunctions, applying a ramp window that weighs simultaneous occurrences based on the distance between words in the document, and using Pearson correlation by setting negative values to 0 rather than using raw count or PPMI.

• GloVe (Global vectors for Word Representation) The model influences statistical information by training only on non-zero elements in the term-context matrix rather than on the entire sparse matrix (as in SVD) or on individual contextual windows in large corpora (as in Word2vec). Global corpus statistics are captured directly by the model. The relationship between two words can be examined by studying the ratio of their probability of simultaneous occurrence with various context words k (if word k is associated with i but not with j, the ratio is large, otherwise it is small; if it is associated with both i and j, then it will be close to 1). This needs to be cleaned from values that are too high (much greater than 1), which correlate well with specific properties of word i, and values that are too small (much less than 1), which correlate well with specific properties of word j.

The starting point in modeling lies in the ratios of probabilities of simultaneous occurrences rather than the probabilities themselves. A simplification can be found; however, this treats all simultaneous occurrences equally, whether rare or frequent. To address this issue, the model resorts to the least squares regression model:

The weighting function must satisfy the following properties:

* f(0)=0 should decrease rapidly enough for x to be finite.
* f(x) should be non-decreasing, so that rare co-occurrences are not overvalued.
* f(x) should be small enough for large values of x, so that frequent co-occurrences are not overvalued.

Pros: Count-based models are efficient in terms of rapid training and utilization of statistics. They are scalable to huge corpora but still perform well on smaller ones.

Cons: On the downside, they have been developed to capture word similarities and disproportionately emphasize large counts.

## Predictive Models - Word2vec

Word2Vec allows the automatic creation of collections of similar concepts from raw texts without requiring specific language skills from the user. Raw texts are implicitly used as supervised training data. To achieve good performance, very large texts (>10 million words) with as many different words as possible are needed.

Word2Vec is based on neural networks with word embeddings and is similar to language modeling, except that it predicts the context rather than the next word. In practice, this translates into minimizing the loss function. The goal is to adjust the vector representation of words so that the loss is minimized.

Architectures:

* Skip-Gram (predict “surrounding words” - like context words - based on the “current word”)
* CBOW (Continuous Bag of Words) (predict the “current word” based on “surrounding words”)

Both algorithms produce word vectors.

Training methods:

* Hierarchical Softmax (and Naïve Softmax)
* Negative Sampling

Formally, in skip-gram, the goal is to maximize the probability of any contextual word given the current central word.

Pros: predictive models can capture complex patterns behind word similarities and perform well in various tasks. Cons: however, they do not scale well with the corpus size and utilize statistics inefficiently.

# Statistical language models

A Statistical Language Model is a model that specifies the probability distribution in a sequence of words. It can be interpreted as a probabilistic mechanism for generating texts (hence it is called a generative model). The problem of language modeling involves learning a probability distribution of sentences that satisfies: Immagine che contiene Carattere, testo, calligrafia, bianco

Descrizione generata automaticamente

The simplest method involves estimating the number of times a sentence occurs in the training data. A more efficient approach is to calculate probabilities using the chain rule of probability. The Markov Assumption simplifies the computation, stating that the next word in a sequence depends only on a small number of previous words.

There are different types of language modeling:  
- Unigram Language Model: distribution of word frequencies in the language does not depend on previous words.  
- N-gram Language Model: probabilities of words depend on the previous n words.

Common problems include zero-frequency, which occurs when encountering new words. Instead of assigning them a probability of 0, the probabilities of unseen words are shifted. Three types of methods are used to address this issue:  
- Discounting methods (Laplace correction, Good-Turing, etc.)  
- Interpolation methods (Jelinek-Mercer, Dirichlet prior, Witten-Bell)  
- Automatic parameter estimation (Zhai-Lafferty method)

There are three types of applications of language modeling in Information Retrieval:  
1. Probability of generating a query from the language model of the document.  
2. Probability of generating a document from the language model of the query.  
3. Comparing the language models of the query and the document topics.

In these cases, probabilities are estimated through maximum likelihood, and smoothing of probabilities is performed to account for unseen words. Two common smoothing methods are:  
- Jelinek-Mercer Smoothing: Proportional to term frequency and inversely proportional to the collection frequency.  
- Dirichlet Smoothing

# Introduction to Information Retrieval

IR (Information Retrieval) is an old Computer Science problem that gained importance with the birth and development of the World Wide Web and the subsequent growth of available information. Systems are sought to identify information useful to the user. To achieve this, it is necessary to interpret the content of texts, images, videos, and audio, and understand the user's needs.

Interest plays a significant role but is difficult to measure as it is subjective. There are three major types of systems for accessing information:

1. **Database Management Systems (DBMS)** - require query formulation. Originally created to handle large amounts of data related to business applications. The semantics and conditions of the data are well-defined, and their retrieval satisfies strict selection conditions expressed through a query language.
2. **Information Retrieval Systems (search engines)** - require query formulation. Developed to handle large amounts of text, they find information related to a topic. Documents must be interpreted, and errors in the results are tolerated. Once the content of the document is interpreted and its representation constructed, given a query, they generate a classification (ranking) of documents based on their relevance. It consists of two main components: the information source and the user in need. The system is the third component (intermediary) that interprets documents and user needs.
3. **Information Filtering Systems (Recommender Systems)** - require user profiling (no query).

To find information, there are two approaches:

* **Pull Technology**:  
  - The user explicitly and dynamically requests information.  
  - Browsing (hypertext)  
  - Retrieval (IR systems)  
  - Browsing and Retrieval (digital libraries and web searches)
* **Push Technology**:   
  - The user is automatically updated with information of probable interest.  
  - Systems for recommending information (goods/services)  
  - Information Filtering

Data are elementary facts that must be interpreted to generate information. To achieve automatic information management, two problems must be solved: a technical one, to make the representation of information efficient, and a semantic one, to make its content effective.

Immagine che contiene testo, ricevuta, Carattere, schermata

Descrizione generata automaticamenteAn Information Retrieval System (IRS) is based on a mathematical model that provides a formal description of the document, query, and their comparison. It is essential to note that using the same formal framework in the representation of documents and queries ensures correct matching.

The main measures of efficiency of IRS are precision and recall: *ret* is the set of documents given after a query *d*; *rel* is the set of documents in the collection relevant to the query *d*.

Information retrieval is challenging for several reasons:

* Incompleteness of document representations
* Subjectivity of the relevance concept
* Ambiguity in the meaning of terms
* Vagueness in user requests
* Uncertainty in result correctnessù
* Approximation of comparison mechanisms

Therefore, it is an activity that must be approached to deal with uncertainty and inaccuracy.

## Indexing and data structures

The automatic indexing of a textual document is the process aimed at associating indices (index terms) with the text. Typically, full-text indexing is performed. The use of indices makes retrieval efficient, based on the keywords specified in a query (e.g., the analytical index in a book). All the indices extracted/associated with all documents in the considered collection form the dictionary of the collection. Usually, the representation of the text consists of a dynamic inverted file structure that must provide an immediate response to user requests, is constructed offline before the search, and optimizes access to the dictionary. A classic, static organization (e.g., Document-term) would not be efficient because the resulting matrix is very sparse.

The dynamic structure consists of two files:

1. **Dictionary:** It contains the index terms, their frequency in the dictionary (to efficiently calculate IDF and optimize Boolean queries), and the pointer to the posting list in the posting file. It can be organized in one of the following three structures:
   * **Linear Structure:** Indices are stored in the dictionary in alphabetical order. It can be quickly explored with binary search and is suitable for sequential evaluation, besides using memory space efficiently. The disadvantage is that it needs to update indices whenever a new term is inserted.
   * **Binary Tree Structure:** Each tree node has 2 children. The search starts from the root node, and each node represents a binary test based on which the search proceeds to one of its two sub-nodes. It is efficient in search (the number of comparisons is at most log n), but it has the disadvantage of needing to rebalance the data structure (maintaining balance property) whenever the dictionary is modified.
   * **B-Tree Structure (B = Balanced):** In a balanced tree of order d, each node has a variable number of children and contains several terms (at most d) and pointers to sub-trees. All nodes except the root must contain a number of terms between a and b. If is the i-th term in an intermediate node: all terms contained in the children nodes from the first to the i-th are lexicographically smaller than . It can be quickly explored (the maximum number of accesses to the B-Tree of order d is d^n, where n is the number of levels, the height of the tree). New terms can be added quickly, and it efficiently uses memory space. The disadvantage is that it is inefficient in sequential search and becomes unbalanced after many insertions.
2. **Posting File:** It consists of the posting list, which includes:
   * The identifier of the document (unique document identifier, DocID) associated with file name or URL.
   * The frequency of the term in the document.
   * The position in the document for each occurrence of the term, which can be expressed as the number of words from the beginning of the document, the number of bytes from the beginning of the document, or the number of sections, paragraphs, etc., in the sentence. The position is optional and is only used by context-aware queries.

To optimize the Posting File, compression techniques are applied to the terms in the dictionary, DocIDs (replacing it with the interval DocID(i) - DocID(i-1)), numerical values denoting term occurrences, and the position of terms in the document text (dividing the text into blocks and referring to them instead of the exact position, making contextual search more complex).

There are several alternatives to the inverted file structure, including:

* **Suffix-Tree/Array Structure:** Used for alphanumeric sequences (Genetic Data).
* **Signature Structure (Bit Mask):** Documents are divided into fixed-length blocks, and each is assigned a signature (with a hash function) that will be matched with the query. Used in multimedia data.

# Basic IR (Information Retrieval) Models

## Boolean Model

It is based on Set Theory. A document is formally represented as a set of index terms associated with binary weights. A query is formally represented as a boolean expression that uses boolean operators AND, OR, and NOT (which can be replaced by BUT when dealing with negation, for efficient use of the inverted file).

The matching mechanism applies set operations. Relevance is modeled as a binary property of documents, the Retrieval Status Value (RSV), which takes values of 0 or 1. The dictionary file is explored, while the posting list is returned. The order of evaluation is crucial and must be specified, as the query produces different results when read from right to left or left to right (no commutative property). Therefore, a priority has been defined:

So: a AND b OR c AND b -> (a AND b) OR (c AND b)

Boolean operators often create confusion, so it is essential to remember that: OR = union, AND = intersection, BUT = AND NOT = difference.

A complete evaluation involves a recursive inspection, where intermediate lists are saved in memory to keep track of partial results. To optimize evaluation, it is useful to consider term frequencies in the vocabulary and prioritize less frequent terms (or groups of terms/indices) in the evaluation.

## Vector Space Model

A vectorial space is defined by a set of linear dependent vectors, defining the vectorial space dimensions. In the case of vector representations of documents, they have a dimension of |V|, corresponding to the set of index terms. Terms serve as axes in the space, while documents are points in these vector spaces. Specifically, each document can be seen as a vector of weights, with one component for each term.

Regarding the choice of weights, one can resort to the following systems:

* Binary
* tf (term frequency): the frequency of the term in the document (also adjusted by logarithm, normalized according to the number of words in the document, or based on the maximum frequency of a term in the document)
* tf-idf: the product of tf and idf.

The vector space model is based on linear algebra. There are two fundamental principles:

1. Both documents and queries are represented in the |V|-dimensional vector space of indices in the considered collection.
2. Documents are classified based on their proximity to the query in the space.

An important assumption is the independence of terms in the documents. Their vector representation serves as an orthonormal basis for the space, and terms are identified by the Base Vector.

Relevance is no longer evaluated in binary terms, as in the boolean model, but is gradual. It is proportional to the proximity of the vector identifying the query with the vector identifying the document.

There are various ways to calculate similarity. The simplest is the dot product. Documents are then sorted in ascending order based on the angle with the query (or equivalently, in descending order based on their similarity = cosine. The cosine is a monotonically decreasing function in the interval [0°, 180°]).

A document is then returned even if it does not completely satisfy the query (unlike the boolean model). Documents of different sizes are normalized for comparison by dividing each component by its length, using, for example, the Euclidean norm. For normalized vectors, cosine similarity is equal to the dot product.

When a term is a good separator between documents that contain it and those that do not, it increases the average distance between documents in the collection, thus producing a less dense document space. A high-frequency term in all documents is not a good index for the document space, as it increases the density of the document space.

In addition to the dot product and the cosine coefficient, there are other indices for measuring similarity:

* Jaccard Coefficient
* Dice Coefficient

The advantages of the Vector Space Model (VSM) are:

* Term weighting improves the quality of results.
* Partial matching allows finding documents that approximate the conditions expressed by the query.
* The cosine formula allows sorting documents according to the degree of similarity with the query.

The disadvantages are:

* The assumption of independence between terms has no real basis.
* Terms that are not present in the query should not have any influence on retrieval.
* The query language is not expressive enough.

## Web Search

The Web can be seen as a vast distributed collection of web pages containing different types of data (images, videos, texts, sounds). Millions of servers are interconnected on the Internet, and in the World Wide Web, there are estimated to be more than 1.7 billion websites.

The WWW can be thought of as a dynamic web graph where nodes consist of URLs, and edges between node x and node y represent links. A graph G is a pair G={V,E}, consisting of a finite set of V vertices and E edges (vertex pairs). A directed graph is a graph where edges have a direction. The relation <—> is an equivalence relation, whose classes are strongly connected components of the graph.

The web has a "Bow-Tie" structure:

* Source Component: contains approximately 24% of web pages; pages point directly or indirectly to the giant component but cannot be reached.
* Isolated Component: cannot be reached by or reach the giant component; links exist between source components and well components that do not pass through the giant component (tubes).
* "Well" Component: 24% of web pages; can be reached by the giant component, but it is not possible to go back (many documents fall into this category as they lack links).
* Giant Connected Component: contains approximately 30% of web pages.

User needs can be of different types:

* Informational: the user wants to know something about a certain topic.
* Navigational: the user wants to visit a specific page.
* Transactional: the user wants to perform an operation through web mediation.
* Mixed.

There are various search tools, such as direct URL search, link-based search (browsing), or the use of web services (search engines, web portals, and recommender systems).

Web search engines were born in '94 and followed the same growth trend as DBMS. Three generations of search engines are recognized:

1. Use only textual data on the page for indexing (word frequency).
2. Use specific web data such as link analysis, click-through data, and anchor text (Google belongs to this generation).
3. Still experimental, tries to "recognize the need behind the query" through semantic analysis, focusing on user needs more than the query, determining context, user interaction, and integrating textual and search analysis.

The structure of a web search engine is very similar to that of a classic IRS, with the addition of a new document collection component. A web page corresponds to a document in a traditional IR. Web search considers the part of the web that is publicly indexable and excludes pages that require permissions, dynamic pages, etc.

The Deep Web is the part of the web whose contents are not indexed by classic search engines. This includes dynamic pages, pages with technically restricted access, and unlinked content—pages not linked by other pages.

## Document Gathering

There are two ways to collect documents:

* Pages are directly submitted to search engines by their owners.
* Search engines are equipped with a software agent called a crawler that navigates the web to send or update new pages to the indexing server. The crawler uses certain URLs known as starting points (well-known interesting points of access, called collection seeds) and then visits other web pages through links that connect one page to another.

A crawler must collect pages by traversing the web graph. This runs on local servers and sends requests to remote servers. The crawling process has the following architecture:

1. Initialize a page queue with some known URLs (popular or user-submitted).
2. Select a URL from the queue.
3. Fetch the page.
4. Look for other URLs on the web page.
5. Discard URLs that cannot be parsed or have already been analyzed.
6. Add URLs to the queue with a breadth-first or depth-first strategy.
7. If time is not up, return to step 2.

There are various challenges that a crawler must address:

* **Seed Selection:** Influences the set of visited pages. Choosing a page located in the giant component can result in visiting the entire web, except for the left side of the bow-tie. However, access to the deep web, representing over 80% of the entire web, is not possible.
* **Which Pages to Download:** Selection is based on content type (header content type). Tools are needed to analyze hypertext formats different from HTML.
* **Which Links to Follow:** Criteria must be established to determine which URLs to follow and which ones to ignore.
* **What to Save for Each Page:** The choice affects the amount of disk space required; pages should be kept in compressed format.
* **Politeness:** Avoid overwhelming a host with consecutive requests, respect possible blacklisting (robots.txt protocol), and provide information (user-agent) during the visit to allow site managers to address potential complaints.
* **Visit Strategies:** How to select the next page to visit (in the visiting algorithm). Two strategies are breadth-first (useful for sites covering related topics) and depth-first (efficient for general crawling).

A centralized crawler is highly inefficient as it spends most of the time waiting for I/O operations. The natural solution is to use multi-process or distributed crawlers.

* **Multi-process:** Within each agent, many processes perform parallel visits or other functions.
* **Distributed:** Composed of many agents performing part of the crawling, visiting a portion of URLs. Agents work on various machines connected via a LAN (intra-site parallel crawler) or a geographical network (distributed crawlers). There may be a need for a central coordinator controlling the crawling progress. Two types of distributed crawlers are those with a central coordinator and fully-distributed crawlers.
  + **With a Central Coordinator:** Tracks visited URLs, decides to assign new URLs based on workload, and communicates links to the central coordinator. It efficiently distributes the workload but requires a high exchange of information, and the coordinator represents a bottleneck and a single point of failure.
  + **Fully-distributed Crawlers:** Agents are entirely free.
  + **Static Assignment:** The set of URLs to visit is partitioned in advance (static assignment), allowing politeness operations and avoiding overlapping, but there is a risk of losing a portion of the web if an agent stops functioning.

**Focused Crawling:** Visits only pages considered relevant to a topic. Consists of two components: a page relevance classifier based on specific topics and a distiller of hub pages containing many relevant outlinks. The crawler visits pages based on the relevance score determined by the classifier and distiller. A useful tool for extracting data from the web is Scrapy.

## Link Analysis

The most significant difference between Web Search and traditional search lies in the fact that links pointing to a page provide a measure of its popularity. Links between pages indicate the existence of a relationship between them. The two primary criteria for estimating relevance are:

1. **Topicality:** Refers to the thematic relevance of the page. The most common models are the Boolean Model and the Vector Space Model.
2. **Popularity:** Represented by link analysis. It utilizes the web's structure to estimate the popularity of pages. The popularity of a page increases when the number of its in-links grows (and may also depend on its out-links). Popular pages are more likely to contain relevant information compared to non-popular ones.

There are two different measures of popularity:

* **Popularity independent from the query (global analysis):** PageRank.
* **Query-dependent popularity (local analysis):** HITS (Hypertext Induced Topic Search).

PageRank is an algorithm used by Google and simulates a random walk of the user on the web. A page with a high PageRank value has many in-links or few in-links from pages with high PageRank. Retrieval is a combination of an estimate of topicality (coherence with the topic) and PageRank value. This algorithm is modeled as a Markov chain process where states are web pages, and transitions are moves from one page to another.

It is assumed that a user visiting a page p can either randomly move to another page with a probability d (the "dumping factor"), e.g., 0.2, or move to a connected page through an out-link with a probability 1-d, following the example above, 0.8. The PageRank of a page p is the probability of accessing page p. It is calculated recursively, starting from any set of values and iterating until convergence.

HITS, also known as "the connectivity Analysis Approach," is a method for finding "authority pages" that respond to a generic query. This arises from the need to introduce a new quality measure to filter the (too many) documents returned by a query. Authority pages contain relevant information, while hub pages are pages that point to useful pages (used as "link sources").

The authority of a page depends on the number of its in-links (but not only, as this is already a characteristic of popular pages but not necessarily authoritative), while hub pages are identified based on their out-links. There is a mutual reinforcement between hubs and authorities:

* The best authority pages have in-links from good hub pages.
* The best hub pages have out-links directly to good authority pages.

The HITS algorithm initializes a set of pages returned by a query, ordered by content criteria, and expands the set, which contains pages that point to or are pointed to by those in the set. Finally, it performs the classification into authority pages and hub pages iteratively.

## Metasearch

Meta-search engines are interfaces that allow the simultaneous submission of the same query to various search engines and merge the results into a single ordered list. The underlying assumption is that many experts in finding information are better than just one. Their effectiveness depends on the algorithm used to merge the lists.

Given a set of ranked lists of documents produced by individual IR systems in response to a query, one can consider the relevance score (if available) calculated by the matching mechanism or the ranks associated with a document (its positions in the ranked lists). It is necessary to define a reliable aggregation strategy that acts on relevance scores or ranks to produce, for each document, a global relevance score and a global rank.

The final step involves reordering the global relevance scores or ranks to obtain a sorted list of documents. The most common metasearch engines are Yippy, Carrot2, and Searx.ù

# Recommender systems

The Recommender systems help connect users with items, reducing the information overload and providing sales assistance. There are different designs for Recommender Systems (RS) that depend on data availability and domain characteristics.

They are widely used as they bring value to users by exposing new and interesting items, compressing the choice set, providing guidance in the options space, etc. They also provide value for providers by increasing the chances of up/cross-selling, conversion rate, click-through rate (CTR), offering personalized services to users, enhancing customer loyalty, persuasion and promotion opportunities, and gaining more knowledge about users, etc.

20% percent of items accumulate 74% of all positive ratings, creating a highly unbalanced product/popularity curve with a long tail. It is precisely on this tail that RS works well, recommending almost unknown items that the user might like.

In general, given a user model (ratings, preferences, demographic data, etc.) and a set of items (with or without descriptions), the goal is to find the relevance score for items (binary or ranking), recommending items assumed to be most relevant.

Recommender systems are based on Information Filtering, the process of monitoring large amounts of dynamically generated information to push subsets of information to the user that are most suitable for their interests (based on their informational needs).

There are four paradigms:

1. **Collaborative Filtering:** most common approach, used by e-commerce and easy to apply to other products (books, music…); uses the approach “wisdom of the crowd”, to make suggestions based on products liked by customers with similar taste to yours. They can be:
   * **Memory-based (or heuristic-based):** Makes predictions considering all available data, measuring user or item similarity through Pearson correlation or cosine similarity. They can be:
     1. **User-based:** Utilizes the user-item ratings matrix and returns a numerical prediction indicating the extent to which the current user will appreciate a certain item, along with the top-N list of recommended items. Not all the neighbors have the same importance, this can be solved by weighting items with high variance. The problems with this method arise because it’s not possible to scale data (problem if there are more users than items).
     2. **Item-based:** Uses item similarity instead of user similarity. It seeks the user's appreciation measure for item B similar to item A. Not all the neighbors are used for prediction, only a small amount (20/50 more or less). These models suffer from the “Cold Start problem” (difficult to assign ratings to new items).
   * **Model-based:** Learns a model from the collection of ratings, used for making predictions. Techniques include Bayesian networks, clustering models, latent semantic models, Markov decision processes, etc. They have an offline phase of pre-processing and a learning phase.
2. Content-Based Filtering: Recommender systems do not require information about items, so they are penalized in collecting information. Content-based filters aim to learn user preferences and recommend items similar to their preferences. Techniques are historically applied to textual documents but can be applied to various item types. One of the most used techniques is the algorithm nearest neighbor.

These models are good to model short-term interests and are usually used in combination with other models for long-term interests. They are rarely used alone in commercial tasks.

1. Knowledge-Based Filtering:
2. Hybrid Filtering:
   * **Monolithic Hybridization Design:** Combines features/knowledge from different paradigms with some pre-processing effort.
   * **Parallelized Hybridization Design:** The least invasive design where weights can be dynamically learned in a weighting/voting scheme.
   * **Pipeline Hybridization Design:** Pre-processes some inputs for subsequent cascading or meta-level refinement of recommendation lists or model learning.

## Evaluating Information Retrieval Systems

The task of evaluating an Information Retrieval System (IRS) is quite challenging, so a methodology for evaluation has been developed to apply to working systems. The components of the standard evaluation experiment for an IRS include:

* The IRS (seen as a black box)
* A collection of documents
* A collection of queries
* A set of users
* A basic evaluation criterion
* One or more measures
* The experiment design

There are general assumptions in text collections:

* The relevance of a document for a user is considered binary.
* The relevance of a document is independent of others.
* The user can discover relevant documents in the collection without the system's assistance.

The most common measures of IRS effectiveness are:

* **Precision:** The fraction of returned documents that are relevant to the given query.
* **Recall:** The fraction of relevant documents that are returned in response to the given query.

The goal of an efficient search engine is to return as many relevant documents as possible while minimizing the number of irrelevant ones.

Several problems are associated with these measures:

* The true Recall is often challenging to calculate, as the total number of relevant documents is unknown.
* Precision and Recall are correlated; hence, it makes more sense to use combined measures like the F1-measure.

How can a document be considered relevant to a query?

* It exactly answers the user's question.
* It partially answers the user's question.
* It provides contextual information on the topic of interest.
* It triggers the user's memory of "forgotten knowledge."

How is the relevance of a document perceived?

* It is subjective based on a user and their informational needs but measurable.
* Users may agree more or less on the relevance of a document.
* The response can be good or bad, complete or partial.
* The response can provide indications for future explorations.

In general, evaluation measures can be distinguished into Set-based, those using all documents, and rank-based, which consider the document's position in the query result list.

Precision@k measures the percentage of relevant documents among the top k returned documents. Recall@k measures the fraction of relevant documents found in the first k positions of a result list ordered by rank, based on the number of relevant documents in the collection.

The Mean Average Precision (MAP) calculates the average of the average precision (AP) across the set of queries. AP is the average Precision@k calculated at all positions k where a relevant document is found.

A commonly used evaluation measure in search engines is the Discounted Cumulative Gain (DCG), which assesses the ability to place highly relevant documents in the initial positions of the results list. It assumes that relevance is not quantified in a binary manner but in numerical values (RSV). Ranks (positions) in the result list are considered and transformed into a list of relevance values (the higher the position, the lower its position number, e.g., 1, the higher the relevance).

The larger the rank, the less useful the document is for the user. Therefore, a function is needed to progressively reduce the value of documents as the rank increases. A common choice is to use the logarithm of the rank as a weight, with the base varying based on the user's "patience" in scanning results (low b -> impatient user, high b -> patient user).

The DCG (Discounted Cumulative Gain) can be normalized into the normalized Discounted Cumulative Gain (nDCG), defined as the DCG at rank k normalized by the value of the DCG at rank k of the ideal result list (iDCG(k)). The ideal list presents the results in perfect order of document relevance.

## Evaluating Recommender systems

In Recommender Systems, the evaluation theme has borrowed paradigms and measures from similar tasks, such as classification and Information Retrieval. The measures can be set-based, rank-based, and user-related. There are three evaluation protocols:

* **Offline Evaluations:** based on pre-collected datasets of user-item interactions; the evaluation goal is prediction accuracy. Different predictive tasks include:
  + Rating prediction: predicting the rating a user will give to an item (typical in Collaborative Filtering). Measures include RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), Normalized RMSE, and Normalized MAE based on the range of ratings, and Average RMSE and Average MAE, useful for imbalanced test sets.
  + Usage prediction: Predicting the items the user deems relevant. Measures are divided into set-based (Precision, Recall, FPR, F1-measure, ROC, etc.) and rank-based (Precision@k, Recall@k, AP, MAP, Mean Reciprocal Rank).
  + Ranking prediction: Predicting the ranking of items, applicable in online tests and user studies, based on ranking metrics. The assumption is that more relevant items are more useful when appearing at the beginning of the recommendation list. Two approaches are reference rating-based (Normalized Distance based Performance Measure - NDPM, Average Precision correlation metric) and utility-based ranking (Rank-score, DCG, nSCG, ARHR - average reciprocal hit rank).
  + Utility maximization: Maximizing the system's utility (profit, time spent on the site).
* **Online Evaluations:** involve users (unaware of the test) in real applications. Compare different test systems by directing a small percentage of traffic to various alternative recommendation engines, recording interactions with the systems. Randomly sampling (redirecting) users is important for a pure comparison and requires significance testing. Online evaluations directly measure the overall goals of the system, such as long-term profit or user retention. They can be used to understand how these objectives are influenced by system properties, such as accuracy and recommendation diversity, and the trade-off between properties.
* **User Studies:** conducted by recruiting a set of test subjects and asking them to perform various tasks requiring interaction with the recommender system. Observe user behavior, and test the influence of the RS on behavior. Use both quantitative and qualitative measures. Tests can be between subjects (each subject is assigned a candidate method to test) or within subjects (each subject tests a set of candidate methods on different tasks, which is more informative).

Most applications use offline evaluations, although they are often not suitable. Recently, more emphasis has been placed on user satisfaction, trying to answer some questions like: interactions with the recommender system are useful? Users are satisfied with the quality of the recommendations they receive? What motivates people to contribute through ratings and comments that enhance the system's prediction quality? What do users prefer more in receiving recommendations? The degree of novelty, or simply the convenience of not searching for products themselves?

In addition to accuracy, various evaluation metrics are used:

* **Coverage:** For both users (how many users can receive recommendations) and items (how many have been recommended) or, more generally, the percentage of users/items for which the RS can generate predictions.
* **Diversity/Novelty:** Avoiding monotonous lists by recommending items the user was not familiar with.
* **Serendipity:** Unexpected and surprising items can be valuable.
* **Learning Rate:** how quickly the Collaborative Filtering becomes predictor of taste when collecting data.
* **Confidence:** Describing the RS's ability to assess the quality of predictions.
* **User satisfaction metrics:** Acquired by interviewing users or measuring retention and usage statistics.
* **Site performance metrics:** Tracking an increase in items sold/downloaded, total user revenue, or retention.
* **Interestingness:** Average reciprocal hit-rank.

# Test Mining Tasks

## Text classification

Classification is the activity aimed at predicting the class membership of data in a predefined set of classes. It is a ubiquitous technology in data science and is formulated as the task of generating classifiers, or models.

In classification, membership in classes does not have to be determinable with certainty (e.g., predicting prime numbers among natural numbers). Textual classification is a type of classification where the data is textual or partially textual. There are many types of classification, including:

* **Binary classification:** Each item belongs exactly to one of two classes (e.g., classification of spam emails).
* **Single-Label Multi-Class classification (SLMC):** Each item belongs exactly to one class, with the number of classes greater than two (e.g., articles classification).
* **Multi-Label Multi-Class classification (MLMC):** Each item can belong to 0, 1, 2, or more classes. It can be solved as n independent classification problems.
* **Ordinal classification:** Similar to SLMC classification, with the addition of an order among the classes.

Classification can be "Hard" or "Soft." Soft classification (SC) aims to find a score (probability, strength of evidence, confidence) of item d belonging to class c. Hard classification (HC) often involves training a soft classifier, where a threshold (t) is applied, and if the score is less than t, d is assigned to class *c1*, otherwise, it is assigned to *c2*.

HC is usually used for standalone classifiers, while SC, where scores are used to rank, is used for interactive classifiers, where humans are in the loop.

In text classification, classification can be performed with respect to various orthogonal dimensions, such as:

* **Topic:** Most frequently applied.
* **Sentiment:** Useful in marketing research, reputation management, CRM, social sciences, political sciences…
* **Language:** (language identification) Useful in query processing in search engines.
* **Genre:** Useful in classifying websites, etc.
* **Author:** (authorship attribution)
* **Native language:** (native language identification)
* **Gender:** (useful in science and cybersecurity)

Among the most common applications of Text Classification are:

* **Knowledge organization:** The goal is to organize knowledge, giving structure to the body of knowledge to make it more effective and easier to analyze/explore.
* **Filtering:** blocking irrelevant items from a dynamic stream. It uses the technique of Hard Classification. Examples include spam filtering or eliminating unwanted content (racist, violent, bullying, fake news).
* **Empowering IR tasks:** Functional improvement of the effectiveness of other tasks in Information Retrieval or Natural Language Processing, such as classifying queries based on intention, classifying questions by type, classifying named entities, word sense disambiguation, etc. Many of these tasks involve very short texts.

A primitive method for building a textual classifier involves knowledge engineering, i.e., manually constructing classification rules. However, this has the disadvantage of being costly to set up and maintain. A more efficient and effective approach is to have a learning algorithm learn from a set of hand-classified examples and use this algorithm to predict the class of new texts. This technique is called supervised learning. The training algorithm requires well-structured input, so documents need to be converted into vectors in a common vector space. The dimensions of the vector space are the features, and the number k of features is called the dimensionality of the vector space.

To generate this vector-based representation for a set of documents D, three steps are followed:

1. **Feature Extraction:** choose between unigram models (bag-of-words, common in topic classification) and n-gram models (sequences of n words). In the first case, the dimensionality k of the vector space depends on the words (or stems or lemmas) that appear at least once in the training set. It may be preceded by stop-word removal and stemming or lemmatization operations. In the second case, the higher the value of n, the greater the semantic significance and the dimensionality k of the resulting representation, but the lower the statistical robustness. Representations in both cases are sparse and high-dimensional. An alternative is character n-grams, useful in degraded texts or those resulting from OCR. These techniques are very useful in topic classification but may prove incorrect in other dimensions. For example, in author classification, features such as sentence length, punctuation, and conjunctions are used. In sentiment classification, however, the bag-of-words is not sufficient; deeper linguistic processes are required. The choice of feature design is dictated by the distinctions one wants to capture.
2. **Feature Selection or Feature Synthesis:** with the techniques previously illustrated, vectors of length or even greater are generated, which can lead to overfitting and high computational costs. Feature selection aims to identify features with more discriminative power to eliminate others. In the case of binary classification, a typical choice is the Mutual Information (MI) criterion, a measure of mutual dependence between two variables. Alternative choices include chi-square and log-odds. Feature Synthesis uses matrix decomposition techniques (SVD, PCA, LSA) to synthesize new features and replace the previously discussed features. These techniques are based on the principle of distributed semantics, which states that the semantics of a word are represented by the words it co-occurs with in a language corpus. It has the advantage of eliminating problems of polysemy and synonymy but is computationally expensive. A "new wave" of distributional semantics is represented by word embeddings.
3. **Feature Weighting:** assign a value to the feature in the vector representing the document: this value can be binary, representing the presence/absence of in , or numeric, representing the importance. The latter can be obtained through feature weighting functions.

For binary classification, any supervised learning algorithm can be used, as the "no-free-lunch" principle applies, meaning there is no learning algorithm that outperforms others in all contexts. Implementations must be able to handle the high dimensionality and sparse nature of TC representations. For non-binary classification, all "SLMC-ready" algorithms (those that can inherently handle multi-class problems) can be used. For others, combinations/cascades of binary versions are needed. For ordinal classifications, ad-hoc algorithms like SVORIM and SVOREX can be used.

Two important aspects in classifier evaluation are:

* **Efficiency:** The consumption of computational resources, at the training and classification levels. It is good practice to report both costs in reports.
* **Effectiveness (accuracy):** Refers to the correctness of the classifier's decisions.

## Text Clustering

Document clustering is the process of grouping a set of documents into clusters that are similar to each other and dissimilar to others. In contrast to supervised learning, there are no training data with known labels, and similarities must be sought within the data itself.

Among the applications of clustering in Information Retrieval (IR) are:

* **Search results clustering:** Clustering search results to provide a more informative presentation to the user.
* **Scatter-Gather:** Clustering the entire collection to provide groups of documents that the user can select or regroup until finding a cluster of interest. It benefits from providing different interfaces to the user (e.g., "search without typing").
* **Collection clustering:** Clustering the collection for a more effective presentation for exploratory browsing.
* **Language modeling:** The model of a document can be interpolated with a collection model. By replacing the collection model with the model of the cluster to which the document d belongs, more accurate estimates of the occurrence probabilities of terms in d can be obtained.
* **Cluster-based retrieval:** Clustering the entire collection to speed up the search by using only the documents in the cluster. This method identifies an initial set of documents that match the query, to which other documents from the same cluster are added, even if they have low similarity to the query.

## Text Summarization

Summaries have many advantages, such as reducing reading time, speeding up the selection process, improving the effectiveness of the indexing process, etc. Advanced summaries include automatic and personalized summaries. The former are less biased than human summaries, while the latter are useful in question/answer systems as they provide personalized information. The use of automatic or semi-automatic summaries allows commercial services to increase the number of documents they can process. Text summarization can be analyzed in three dimensions:

* Based on input type For single documents, it produces abstracts, outlines, and headlines, while for multiple documents, it provides the "essence" of the content.
* Based on the purpose
  + 1. Generic: The model makes no assumptions about the domain or context of the text and treats all inputs uniformly (used in the majority of cases).
    2. Domain-specific: The model uses specific domain knowledge for more accurate summarization.
    3. Query-based: The summary contains only information that addresses the informational needs of the user who submits the query. It is a kind of complex Q/A: it answers a question by summarizing a document that contains the information to construct the response. In the case of single documents, it creates "snippets" (e.g., Google), while for multiple documents, it creates a response that combines information from each document cohesively.
* Based on output type
  + - 1. Extractive summarization: Initially, the most important sentences are selected from the input text, and subsequently, using these sentences, the summary is created.
      2. Abstractive: It expresses the ideas of the document(s) using, at least in part, different words. The model forms its own sentences and phrases to provide a coherent summary, akin to what a human would generate. Despite being a more intriguing approach, it is very challenging.

3 steps for extractive summarization:

1. Intermediate representation: Intermediate representation of the input that captures only the key aspects of the text. The representation can follow two approaches:
   1. Topic representation approach: deriving the intermediate representation from the text to extract the discussed topic. It dates back to Luhn's analyses and involves selecting sentences that have descriptive words (topic words/signatures). Frequently occurring content words are considered indicative of the article's topic, and a frequency threshold is used to identify descriptive words in the document to be summarized. The importance of a sentence is calculated based on the number of topic signatures it contains and the proportion of these signatures relative to the sentence's length. The two weight systems for common words are tf-idf and word probability, defined as the word's frequency relative to the input tokens and used by selecting sentences containing the words with the highest probability. This ensures that the word with the highest probability (assumed to be the most representative of the text's topic) is included in the summary. The process is iterative, adjusting down the probabilities of the selected words (this penalizes the repetition of the same word in the summary). In the case of tf-idf representation, clustering is performed on the tf-idf vectors by adding documents to the cluster and recomputing centroids. These centroids are considered as pseudo-documents consisting of words with tf-idf scores higher than a certain threshold, forming the cluster. Subsequently, the centroids are used to identify sentences in each cluster that are central to the cluster's topic. Sentences with more words than the cluster's centroid are considered central sentences. Two metrics are used: CBRU (Cluster-based relative utility) measures the relevance of a particular sentence to the topic representing the entire cluster, and CSIS (Cross-sentence informational subsumption) measures redundancy in sentences. The final score and sentence selection are determined by combining these metrics. An alternative representation technique is Latent Semantic Analysis (LSA), an unsupervised technique that derives a semantically implicit representation of text based on co-occurring words. It was initially proposed for the generic summarization of news documents, serving as a method for identifying important topics without the use of lexical resources such as WordNet. The underlying hypothesis of LDA is that sentences that frequently discuss the topic are often good candidates for summaries. The last technique for intermediate text representation in topic modeling is Bayesian Topic models, which are probabilistic models that discover and represent document topics.
   2. Indicator representation approach: Indicator-based representations aim to model the text representation based on a set of features and use them to classify sentences directly, rather than representing the input text's topic. Graph-based methods and Machine Learning techniques are employed to determine important sentences to include in the summary. The use of graph-based methods is influenced by the PageRank algorithm: it represents the document as a connected graph where sentences form the vertices, and their similarity forms the edges (only if greater than a certain threshold; otherwise, nodes are not connected). The most common measure of similarity is cosine similarity, used with tf-idf weights. Two outcomes are obtained from the graph: the partitions (sub-graphs) included in the graph create discrete topics covered in the document, while the second outcome is the identification of the most important sentences in the document as the most connected nodes. They can be used for both single and multiple documents and do not require specific linguistic processing, making them applicable to various languages. However, the use of tf-idf weights poses some limitations because it preserves only word frequency, not their semantics and syntax. The most commonly used models are TextRank and LexRank. The first was developed for single documents, while the second for multi-document summarization. TextRank uses cosine similarity on tf-idf vectors, while LexRank uses a measure based on the number of words two sentences have in common (normalized for sentence length). In the first model, edges are not weighted, while in the second, the similarity measure is used as the weight. In both models, sentences are ranked by applying a PageRank-like algorithm to the resulting graph. Additional features, such as the sentence's position and length, can be incorporated into LexRank. According to the Machine Learning approach, summarization is seen as a classification problem between summary sentences and non-summary sentences based on their features, such as sentence position in the document or paragraph, length, similarity to the title, presence of named entities, etc. (given a training set with the summaries).
2. Classify sentences: once the intermediate representation is obtained, each sentence is assigned a score indicating its importance (as seen for each technique in the previous paragraph). In general, for the topic representation approach, the score is relative to how much the sentence expresses the most important topics in the document and how it combines them. In contrast, for indicator representation, the weights of each sentence are determined by combining evidence from each indicator, using Machine Learning techniques to discover the indicator weights.
3. Select sentences for the summary: the summarizer must select the best combinations to form a summary within a paragraph. There are three approaches:
   1. **Best n:** The n most important sentences, when combined to achieve the desired length, are selected to form the summary.
   2. **Maximal Marginal Relevance (MMR):** Sentences are selected in a greedy iterative manner. At each step, the importance of a sentence is recalculated as a linear combination of its original importance weights and its similarity to the already chosen sentences (those most similar to the selected sentences will be discarded). A typical problem with MMR occurs when a very long and relevant sentence is evaluated as the most informative. This sentence likely contains noise and limits space for the remaining sentences. One solution is to penalize very long sentences.
   3. **Global Selection:** The optimal collection of sentences is selected based on constraints aiming to maximize overall importance, minimize redundancy, and, for some approaches, maximize coherence.
4. Evaluation: Evaluating summarization tasks is a very challenging task because there is no ideal summary for the document (or documents), and the optimal summary is highly subjective. Additionally, common standard evaluation metrics are lacking. For automatic summary evaluation, it is crucial to decide and specify the most important parts of the text to preserve, ensure that the evaluator can automatically identify crucial pieces of information, and maintain coherence and grammatical correctness in the summary's readability. The simplest way to evaluate a summary is, therefore, through human evaluation, focusing on grammar, non-redundancy, integration of the most important information, structure, and coherence. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is the most common metric for the automatic evaluation of summaries by comparing them with human reference summaries. There are three variants:

* **ROUGE-n:** Based on the comparison of n-grams in the summary (p) with those in the reference summary (q).
* **ROUGE-L:** Uses the concept of Longest Common Subsequence. The longer the LCS between two sentences in the summaries, the more similar they are (although more flexible than ROUGE-n, this implies that n-grams are consecutive).
* **ROUGE-SU:** Considers bi-grams as unigrams and allows the insertion of words between the first and last words of the bigrams, avoiding the need to evaluate word sequences.

## Topic Modeling

Topic Modeling is an unsupervised machine learning technique aimed at scanning a set of documents, detect words and phrases patterns within them and automatically clustering word groups which best characterize a set of documents. It provides a set of words that makes sense together (*topic*). The main differences between Text Clustering and Topic Modelling is that the first procedure groups documents into different clusters based on similarity (or distance) measures, while the second one groups words into different clusters where each word have a probability of occurrence for the given topic.

The ***L***atent ***S***emantic ***A***nalysis (**LSA**) is a Topic Modeling Technique which computes how frequently words occur in the documents and assumes that similar documents will contain the same distribution of word frequencies. The main idea is decompose the Document Term Matrix into a Document Topic and Topic Term matrix. The standard method for computing word frequencies is the Tf-Idf for each term in a document. It can be decomposed using SVD.

Furthermore LSA can only capture partially polysemy, but this is not a problem because it can find the predominant sense in a document. An alternative of this techniques is the *Probabilistic LSA* (**pLSA**), which identifies and distinguishes words in different contexts without using the dictionary. In particular, it allows to disambiguate polysemy and discloses topical similarities grouping together words sharing the same context. It finds a probabilistic model with hidden (or latent) topic that can generate the observed data in the Document Term Matrix. pLSA expresses data in three different variables: documents, words and topics.

The ***L***atent ***D***irichlet ***A***nalysis (**LDA**) is the Bayesian version of pLSA. It categorizes documents, treated as Bag of Words, by topic via a generative probabilistic model assuming they are produced from a mixture of topics. Dirichlets distributions is better in finding the assignment of documents to a general topic. In a corpus of 𝑀 documents, it is necessary to discover 𝑘 topics. LDA outputs the topic model and the 𝑀 documents expressed a combination of topics finding the weight of connections between documents and topics and words. The algorithm creates an intermediate layer with topics and finds the weights. In this case documents are no longer connected to words but to topics.

A Dirichlet distribution 𝐷𝑖𝑟(𝑝) is a probability function which gives probabilities for discrete random variables. It includes the concentration parameter 𝑝 that rules the trend of the distribution:

* Sparse, producing a real life distribution: 𝑝 < 1;
* Uniform, producing a random distribution: 𝑝 = 1;
* Concentrated: 𝑝 > 1.

The main difference between LSA and LSA is the assumption of the Dirichlet distribution, while the second does not assume any distribution. LSA works better than pLSA because it can better generalize new documents.

There are different approaches evaluating topic modeling:

* Eye Balling Models, identifying the top 𝑛 words in a documents;
* Intrinsic Evaluation Metrics, using the *Perplexity*, a measure of uncertainty, and the *Coherence*, measuring the semantic similarity between top words within the topic. They help to distinguish topics in an interpretable topics from topics that are artifacts of statistical inference;
* Human Judgments, so what is a topic;
* Extrinsic Evaluation Metrics, so if the model is good at performing task, such as classification.