INTRODUCTION TO DATA SCIENCE

Natural language processing

Maciej Świtała, PhD

Autumn 2025

UNIVERSITY OF WARSAW | FACULTY OF ECONOMIC SCIENCES

Introductory matters

Brief review of basic concepts

- **Natural Language Processing (NLP)** is a field that provides the methods and tools for analysing, **understanding**, and **generating** human language.
- **Text mining** is the process of extracting information and patterns automatically from large text collections; it can be seen as an **application of NLP**.
- Text mining covers the following subfields:
  1. **information extraction**, including, i.a., pattern matching, named entity recognition, and text summarisation,
  2. **sentiment analysis**, including text scoring and text classification (semi-supervised and supervised learning),
  3. **topic modelling**, i.e., simply put, clustering texts (unsupervised learning).
- The algorithms used for the tasks above have historically evolved from simple statistics to machine-learning algorithms, including modern NLP solutions, i.e., **Large Language Models (LLMs)**.

Specificities of textual data processing

- Textual data is typically **unstructured**, which means it must first be cleaned, normalised, and transformed into a structured form before meaningful analysis can take place.
- Since computers operate on numerical representations, text must be **converted into numbers** using methods such as tokenisation, vectorisation, or modern embedding techniques.
- Text analysis is often **exploratory in nature**, involving the discovery of patterns, themes, and relationships that may not be known in advance.
- Text mining **supplements other analytical methods** by providing insights derived from large collections of unstructured text, enriching traditional quantitative or qualitative analyses.

Textual data cleaning

Cleaning textual data usually involves, among other things:

- removing newline characters,
- removing numbers,
- removing special characters,
- converting all letters to lowercase,
- removing one-letter words.

Cleaning textual data - example

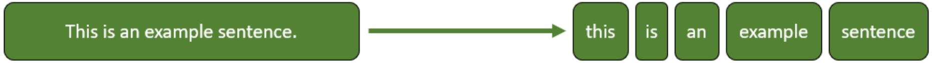
- "7 out of 10 penguins responded that they would prefer living in a warmer place than Antarctica!!!11%^"
- "out of penguins responded that they would prefer living in warmer place than antarctica"

Cleaning textual data - possible problems

- Is removing all the numbers from the corpus always a good idea?
    - **Example #1** - proper names containing numbers, e.g. "doc2vec" will be made "docvec", "windows98" will be made "windows".
    - **Example #2** - numbers in phrases can convey important meaning, e.g. "25 years imprisonment" and "2 years imprisonment" will be made "years imprisonment".
- Is converting all letters to lowercase always a good idea?
    - **Example #3** - conversion to lowercase can create homonyms, e.g., Odra" (en. Oder) is a river in Poland whereas "odra" (en. measles) is an illness, converting both to lowercase makes an algorithms to treat them as the same token.
- Is removing single-letter words always a good idea?
    - **Example #4** - abbreviations: Polish "sp. z o.o.", which means "LLC" (Limited Liability Company), after removing punctuation makes "sp z oo", then, removing all single-letter words makes "sp oo", when considering single words, one considers "sp" and "oo" further, which can make it difficult in interpretation.

Tokenisation

- Tokenisation is the process of **converting strings into lists of terms**.
- In other words, it involves **breaking text into smaller units**.
- It is a standard operation that must be performed before any textual analysis.
- As a result, it is well supported in most programming languages.

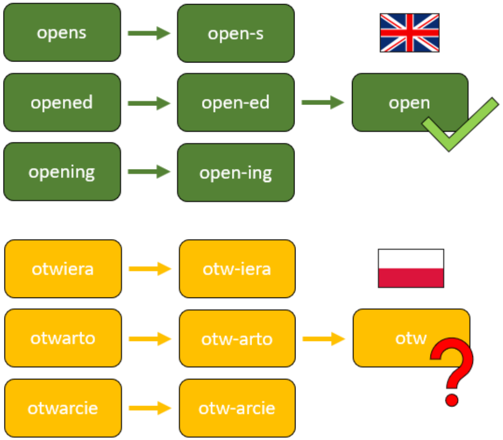


Tokenisation - example

- "out of penguins responded that they would prefer living in warmer place than antarctica"
- out, of, penguins, responded, that, they, would, prefer, living, in, warmer, place, than, antarctica

Stemming

- Stemming is the process of **reducing a term to its root form** by removing prefixes and suffixes.

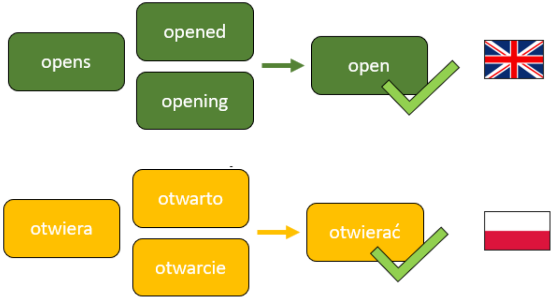


Stemming - example

- out, of, penguins, responded, that, they, would, prefer, living, in, warmer, place, than, antarctica
- out, of, penguin, respond, that, they, would, prefer, liv, in, warm, place, than, antarctic

Lemmatisation

- Lemmatisation is the process of **converting words to their base** or dictionary form (lemma).



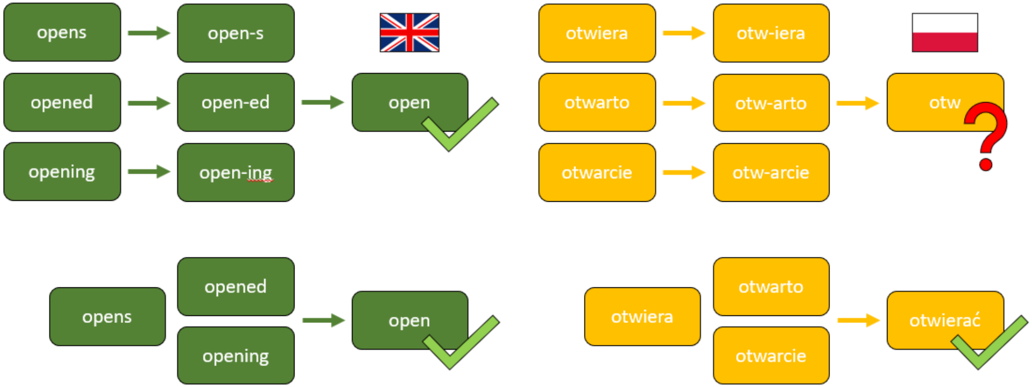
Lemmatisation - example

- out, of, penguins, responded, that, they, would, prefer, living, in, warmer, place, than, antarctica
- out, of, penguin, respond, that, they, would, prefer, live, in, warm, place, than, antarctica

Stemming vs. lemmatisation

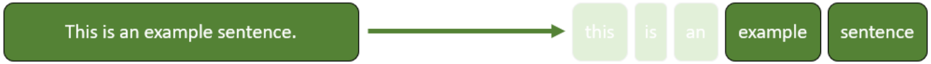
- Stemming and lemmatisation usually produce **different results** - for example, "living" may become "liv" through stemming, and "live" through lemmatisation.
- For English terms, **stemming is often sufficient**, which is why it is widely used.
- However, **for languages like Polish, stemming is often ineffective** - for instance, both "oko" (eye) and "oczy" (eyes) may be stemmed to just "o".
- On the other hand, stemming handles neologisms more easily than lemmatisation - for example, "tiktokisation" can be reduced to "tiktok" via stemming, while lemmatisation may not recognise the word at all.
- In general, stemming appears more aggresive and less precise.

Stemming vs. lemmatisation



Removing stopwords

- Some words carry **no meaningful information** and **should be removed** from the analysed corpus.
- Typically, predefined stopword lists are used for this purpose.
- In practice, it is also common to remove additional tokens based on expert judgment, effectively treating them as stopwords.



Removing stopwords - example

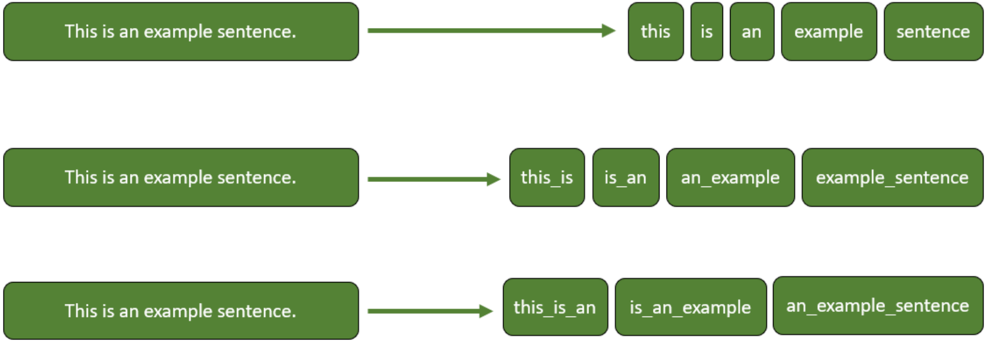
- out, of, penguin, respond, that, they, would, prefer, liv, in, warm, place, than, antarctic
- out, penguin, respond, prefer, liv, warm, place, antarctic

Removing additional tokens treated as stopwords - example

- Should "SSO" (acronym for the Polish "sędzia sądu rejonowego", equivalent to "judge of the regional court") be treated as a stopword?
- Should "Maciej Świtała" be treated as a stopphrase?
- Should "XX" be treated as a stopword?
- Should "model" be treated as a stopword?
- Should "ob" be treated as a stopword?

N-grams

- **N**-grams are **sequences of n words that occur one after another** (usually excluding stopwords) in a text.
- Examples include: **uni**grams (single tokens), **bi**grams (pairs of consecutive tokens), **tri**grams (sequences of three consecutive tokens), and so on.



N-grams - example

- out, penguin, respond, prefer, liv, warm, place, antarctic
- unigrams: [out, penguin, respond, prefer, liv, warm, place, antarctic]
- bigrams: [out-penguin, penguin-respond, respond-prefer, prefer-liv, liv-warm, warm-place, place-antarctic]
- trigrams: [out-penguin-respond, penguin-respond-prefer, respond-prefer-liv, prefer-liv-warm, liv-warm-place, warm-place-antarctic]

When are n-grams especially useful?

- Legal language - e.g., "gross ingratitude", "shall be punishable by imprisonment"; also includes Latin maxims such as "nemo plus iuris", "lex posterior derogat legi priori", "lex specialis".
- Genetics - e.g., protein or DNA sequencing.
- Computational linguistics - e.g., predicting the next word in a text or communication system.

What are the problems with n-grams?

- N-grams **increase the number of tokens** - the higher the n-gram order, the more tokens need to be processed by the algorithm.
- As for the information carried by n-grams, **many are meaningless**, as any sequence of adjacent words is treated as a valid n-gram, regardless of context.
- These issues can be mitigated by filtering n-grams based on their frequency of occurrence in the corpus (see the upcoming slides).

Should we use all tokens for modelling?

- Usually, a corpus contains multiple tokens that are not included in common stopword dictionaries but are either **very frequent or completely non-informative**.
- Very frequent tokens tend to be **too general or semantically similar** to provide informative content.
- Conversely, extremely rare tokens introduce **unnecessary noise** into the corpus, being **too specific to be meaningfully interpreted**.
- Tokens as described above should generally be removed from the corpus before classic text mining. However, how should thresholds be determined? **Which tokens are "too frequent" and which are "too rare"?**

Should we use all the tokens for modelling? - example

- When analysing scientific papers on machine learning, most documents will include tokens such as "machine", "learning", "model", "predict", and "accuracy". Conversely, tokens like "pokemon" are expected to occur very rarely, if at all.
- When analysing court judgements, almost all documents will contain tokens like "judgement", "court", and "judge", while tokens such as "pokemon" should be virtually absent.

Should we use all the tokens for modelling?

- Again, both "too frequent" and "too rare" tokens should be removed from the corpus before modelling.
- Common approaches include filtering tokens with:
    1. Term Frequency (**TF**),
    2. Document Frequency (**DF**),
    3. and their combination (**TF-IDF**).

Term Frequency (TF)

- Simple assumption: frequent terms are prominent.
- Term Frequency (TF) measures **how often a particular token appears in a specific document**.
- Therefore, TF is considered a **document-specific measure**.
- The TF of token i in document j is defined as:

  $$\mathrm{TF}_{ij} = \frac{f_{ij}}{n_j}$$

  where f_{ij} denotes the number of occurrences of token i in document j, and n_j is the total number of tokens in document j.
- The main limitation of TF is that it highlights the most frequent terms, while we are usually interested in terms that best differentiate between texts. Therefore, using TF alone is not recommended.

Document Frequency (DF)

- Document Frequency (**DF**) measures in **what share of documents a given term appears** (at least once).
- Therefore, DF is called a **term-specific measure**.
- DF for token i is defined as:

$$\mathrm{DF}_i = \frac{N_i}{N}$$

  where N is the total number of documents in the corpus, and N_i is the number of documents in which token i occurs at least once.
- Sometimes, Inverse Document Frequency (**IDF**) is used instead of DF, as the log-scale enables more informative graphical presentation:

$$\mathrm{IDF}_i = \ln \frac{N}{N_i}$$

TF-IDF

- TF-IDF is a relative frequency of a word in a specific document compared to the inverse proportion of that word over the entire document corpus.
- TF-IDF for token i in document j is defined as:

$$\text{TF-IDF}_{ij} = \text{TF}_{ij} \times \text{IDF}_i$$

- TF-IDF is one of the most popular ways for weighting tokens, widely applied in different programming languages.

Most prominent text mining tasks

What is information extraction?

"Information extraction turns the unstructured information embedded in texts into structured data."

Source: Jurafsky, D., & Martin, J. H. (2025). Speech and Language Processing (3rd ed. draft). Stanford University. Retrieved from: web.stanford.edu/ jurafsky/slp3/.

What is information extraction?

- Information extraction is often associated with a set of operations performed on text, since this is the **text that appears the most common example of unstructured information**.
- Information extraction is eventually, **in a strict sense**, an natural language processing field focused on automatically **converting unstructured text into structured information**.
- Instead of just reading text, information extraction methods **identify who, what, where, when, and how**, and represents them as **entities, relations, events, or database entries**.

Information extraction procedure

- Information extraction procedure varies depending on the type of the algorithm applied.
- Still, it can be stated that each extration:
  1. begins with **input preprocessing**,
  2. continues with a **method-specific extraction** procedure,
  3. ends with **output postprocessing**.

Core information extraction tasks

1. **Named entity recognition** - detecting persons, organisations, products, dates, locations.
2. **Relation extraction** - finding semantic relations.
3. **Event extraction** - detecting events and arguments.
4. **Template filling** - populating predefined fields.
5. **Coreference resolution** - linking mentions of same entity.
6. **Entity linking** - mapping entities to knowledge base IDs.

Named entity recognition - example

- Named entity recognition example - **detect and classify** named entities from the following sentences:
    - *Elon Musk founded SpaceX in 2002 in California.*
    - *Microsoft released Windows 11 in October 2021.*
- The **detected** named entities are:
    - first text → [Elon Musk, SpaceX, 2002, California],
    - second text → [Microsoft, Windows11, October 2021].
- Then, we deal with **classification**:
    - persons: Elon Musk, -,
    - organisations: SpaceX, Microsoft,
    - products: -, Windows11,
    - dates: 2002, October 2021,
    - locations: California, -.

Relation extraction - example

- Relation extraction example - **find semantic relations** in the following sentences:
    - *Marie Curie worked at University of Paris.*
    - *Apple produces iPhone.*
    - *Warsaw is the capital of Poland.*
- The semantic **relations** are:
    - first text $\rightarrow$ (works for),
    - second text $\rightarrow$ (produces),
    - third text $\rightarrow$ (capital of).

Event extraction - example

- Event extraction example - **detect events details** from the following texts:
  - *On May 5, 2020, an explosion occurred in Beirut.*
  - *Lionel Messi scored two goals against Real Madrid.*
  - *Tesla announced the launch of Cybertruck*.
- The **events** are:
  - first text → [who: -; what: explosion; where: Beirut; when: May 5, 2020],
  - second text → [who: Lionel Messi; what: two goals; where: -; when: -],
  - third text → [who: Tesla; what: launch of Cybertruck; where: -; when: -].

Template filling - example

- Template filling example - **populate predefined fields** with extracted information.
- Let us define the **fields** to be populated:
  - for CV parsing: [name: , education: , experience:],
  - for invoice parsing: [invoice no.: , date: , amount:],
  - for sports articles parsing: [team A: , team B: , score:].
- Extract **information** from document and populate the fields:
  - CV parsing: [name: John Smith], [education: MIT], [experience: engineer at Google],
  - invoice: [invoice no.: INV/2021/05/123], [date: 12.05.2021], [amount: $1230.50],
  - sports article: [team A: FC Barcelona], [team B: Real Madrid], [score: 2:1].

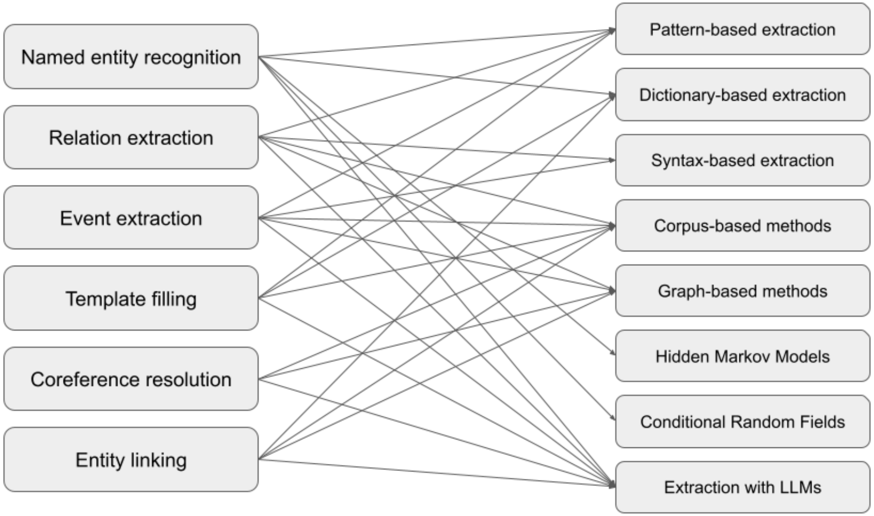
Coreference resolution - example

- Coreference resolution example - link mentions that **refer to the same entity** within the following texts:
    - *Angela Merkel was Germany's chancellor. She ruled for 16 years.*
    - *Amazon was founded by Bezos. The company started with books.*
    - *iPhone 15 was unveiled. The new model will go on sale next month.*
- Here, **coreferences** are:
    - for the first text → [she - Angela Merkel],
    - for the second text → [company - Amazon],
    - for the third text → [new model - iPhone 15].

Entity linking - example

- Entetity linking example - **map** the following entity mentions **to knowledge base IDs**:
  - *Paris*,
  - *Jaguar*,
  - *Python*.
- Next, we assign **knowledge base IDs** to each entity:
  - *Paris* → [Paris, France (Q90)] vs. [Paris, Texas (Q755479)],
  - *Jaguar* → [Jaguar Cars (Q187143)],
  - *Python* → [Programming language (Q28865)] vs. [Snake family (Q33573)].

Information extraction tasks - methods connections



Information extraction methods

- **Pattern-based extraction:**
  - uses hand-crafted patterns or rules (e.g., regular expressions),
  - captures fixed text structures, such as "X was born in Y",
  - intuition: if text follows a predictable template, a pattern can extract information directly.
- **Dictionary-based extraction:**
  - relies on predefined lexicons of entities or terms,
  - example: a dictionary of company names, diseases, or places,
  - intuition: if a word matches the dictionary, it is recognized as relevant.
- **Syntax-based extraction:**
  - uses syntactic structures from parsing (e.g., subject-verb-object relations),
  - intuition: grammatical dependencies reveal relationships between words,
  - example: "Alice founded OpenAI" $\rightarrow$ founder(Alice, OpenAI).

Information extraction methods - cont'd

- **Corpus-based methods:**
  - exploit statistical co-occurrence across large text corpora,
  - intuition: entities and relations that frequently appear together are likely connected,
  - often used in unsupervised or semi-supervised extraction.
- **Graph-based methods:**
  - represent entities and relations as nodes and edges,
  - apply algorithms like clustering, ranking, or propagation on graphs,
  - intuition: information is structured as a network; connections reveal knowledge.

Information extraction methods - cont'd

- **Hidden Markov Models (HMMs):**
  - sequence models with hidden states generating observable words,
  - useful for tagging tasks such as named entity recognition,
  - intuition: text is produced by an underlying sequence of categories.
- **Conditional Random Fields (CRFs):**
  - discriminative sequence models that condition on the whole input,
  - capture dependencies between labels (e.g., "B-PER" followed by "I-PER"),
  - intuition: labels in a sequence are interdependent, not independent.
- **Extraction with Large Language Models (LLMs):**
  - **transformers** (e.g., BERT, GPT) can perform extraction with minimal supervision,
  - intuition: pre-trained models capture world knowledge and linguistic patterns,
  - flexible and adaptable across tasks without heavy feature engineering.

Applications of information extraction

- Automatic extraction metrics in financial reports.
- Capturing disease symptom.
- Crisis monitring in social media.
- Deriving gene relations.
- Detecting and classifying events from financal news text.
- Extracting clauses, terms, and obligations from legal documents.
- Extracting drug interactions.
- Knowledge base and knowledge graphs population.
- Social media opinion mining.
- Trend detection in social media.

What is sentiment analysis?

"Sentiment analysis (also known as sentiment classification or opinion mining - M.Ś.) *is the process of* ***gathering and analyzing opinions****, thoughts, and impressions regarding various topics, products, subjects, and services"."*

Source: Wankhade, M., Rao, A.C.S. & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. Artificial Intelligence Review 55, 5731-5780, DOI: 10.1007/s10462-022-10144-1.

What is sentiment analysis?

- Sentiment analysis **aims to detect sentiment polarity or specific emotions**, typically expressed in natural language.
- Technically, sentiment analysis involves **using information, primarily from texts, to obtain their specific characteristics**.
  1. scoring texts based on a **selected dictionary** (lexicon-based methods),
  2. scoring texts based on a **self-developed, domain-specific dictionary** (corpus-based methods),
  3. **making predictions** of text characteristics utilising **features constructed from their content**,
  4. scoring texts based on **large language models** (transformer-based methods).

Sentiment analysis methods scalability

- Restricting the discussion solely to ”sentiment” analysis may undervalue the broader scope of research in this field.
- The term ”sentiment” traditionally refers to ”opinions, thoughts, and impressions”, its conceptual scope should not be unnecessarily broadened. Nevertheless, **sentiment analysis methods can be effectively applied to any form of text analysis focused on predicting specific textual attributes**.
- Thus, while sentiment analysis is primarily concerned with ”sentiment”, its **methods are highly adaptable and scalable**.

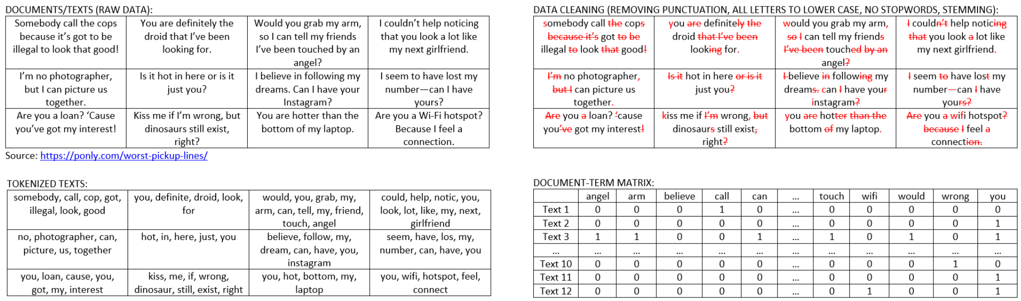
Sentiment analysis procedure

1. Aim of the study.
2. Data collection, usually through the web scraping techniques.
3. Textual data preprocessing (if required to use a method): cleaning, tokenization, removing stopwords, stemming, lemmatization, frequency-based filtering, n-grams inclusion.
4. **Sentiment analysis**, with hyperparameters optimisation (if possible when using a method).
5. Drawing conclusions.

Sentiment analysis input

- Sentiment analysis input varies depending on method to be used:
  - in lexicon- and transformer-based methods, the input is usually **raw text**,
  - in corpus-based methods and supervised sentiment analysis, the input is usually some variation of a **Document-Term Matrix** (DTM).
- In DTM:
  - **row index stands for the analysed texts**,
  - **column index represents unique tokens** (words, pairs of words, etc.),
  - **each cell includes counts of tokens in individual documents** or, as typically in sentiment analysis, their transformations.

Textual data preprocessing: from raw text to DTM



DOCUMENTS/TEXTS (RAW DATA):

Somebody call the cops because it's got to be illegal to look that good!	You are definitely the droid that I've been looking for.	Would you grab my arm, so I can tell my friends I've been touched by an angel?	I couldn't help noticing that you look a lot like my next girlfriend.
I'm no photographer, but I can picture us together.	Is it hot in here or is it just you?	I believe in following my dreams. Can I have your Instagram?	I seem to have lost my number—can I have yours?
Are you a loan? 'Cause you've got my interest!	Kiss me if I'm wrong, but dinosaurs still exist, right?	You are hotter than the bottom of my laptop.	Are you a Wi-Fi hotspot? Because I feel a connection.

Source: https://ponly.com/worst-pickup-lines/

DATA CLEANING (REMOVING PUNCTUATION, ALL LETTERS TO LOWER CASE, NO STOPWORDS, STEMMING):

somebody call ~~the~~ cop~~s~~ ~~because it's~~ got ~~to be~~ illegal ~~to~~ look ~~that~~ good~~!~~	you ~~are~~ definite~~ly the~~ droid ~~that I've been~~ look~~ing~~ for.	would you grab my arm, ~~so I~~ can tell my friend~~s~~ ~~I've been~~ touch~~ed by an~~ angel~~?~~	~~I~~ could~~n't~~ help notic~~ing~~ ~~that~~ you look ~~a~~ lot like my next girlfriend.
~~I'm~~ no photographer, ~~but I~~ can picture us together.	~~Is it~~ hot in here ~~or is it~~ just you~~?~~	~~I~~ believe ~~in~~ follow~~ing~~ my dream~~s.~~ can ~~I~~ have you~~r~~ instagram~~?~~	~~I~~ seem ~~to~~ have los~~t~~ my number—can ~~I~~ have you~~rs?~~
~~Are~~ you ~~a~~ loan? ~~'~~cause you~~'ve~~ got my interest~~!~~	kiss me if ~~I'm~~ wrong, ~~but~~ dinosaur~~s~~ still exist~~,~~ right~~?~~	you ~~are~~ hot~~ter than the~~ bottom ~~of~~ my laptop.	~~Are~~ you ~~a~~ wifi hotspot~~?~~ ~~because I~~ feel ~~a~~ connect~~ion.~~

TOKENIZED TEXTS:

somebody, call, cop, got, illegal, look, good	you, definite, droid, look, for	would, you, grab, my, arm, can, tell, my, friend, touch, angel	could, help, notic, you, look, lot, like, my, next, girlfriend
no, photographer, can, picture, us, together	hot, in, here, just, you	believe, follow, my, dream, can, have, you, instagram	seem, have, los, my, number, can, have, you
you, loan, cause, you, got, my, interest	kiss, me, if, wrong, dinosaur, still, exist, right	you, hot, bottom, my, laptop	you, wifi, hotspot, feel, connect

DOCUMENT-TERM MATRIX:

	angel	arm	believe	call	can	...	touch	wifi	would	wrong	you
Text 1	0	0	0	1	0	...	0	0	0	0	0
Text 2	0	0	0	0	0	...	0	0	0	0	1
Text 3	1	1	0	0	1	...	1	0	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...
Text 10	0	0	0	0	0	...	0	0	0	1	0
Text 11	0	0	0	0	0	...	0	0	0	0	1
Text 12	0	0	0	0	0	...	0	1	0	0	1

Source: own elaboration.

Sentiment analysis components

As for the sentiment analysis per se, it includes **two steps**:

1. subjectivity classification,
2. sentiment classification.

Subjectivity classification

- Subjectivity classification aims to recognize subjective cues and emotional phrases, **distinguishing subjective from objective text**.
- It can be viewed as the task of determining the subjectivity level of a text.
- Words such as "difficult", "expensive", or "great" are typically considered subjective, as they carry clear sentimental or evaluative meaning.
- In fact, it can be regarded as the initial step towards sentiment analysis, as it focuses on filtering out irrelevant textual components from further processing (acting **like an initial feature selection**).

Sentiment classification

- Sentiment classification includes subtasks such as polarity detection, cross-language classification, and cross-domain classification.
- Polarity detection seeks to determine whether a given text excerpt **expresses a "positive" or "negative" sentiment**. Sometimes, identifying "neutral" sentiment is also relevant (unless it is filtered out during subjectivity classification).
- Cross-language and cross-domain classification involve predicting sentiment in a different language or domain than the one used during model training.

Sentiment analysis output

- Sentiment analysis returns **scores for prespecified units of text**, i.e., word, phrase, sentence, text, or aspect.
- The sentiment score to be obtained can be a **continuous, binary, or categorical variable**.
- Sometimes, one can be more interested in **probabilities** of certain binary or categorical variable values.

Typology of sentiment analysis applications

- Aspect analysis.
- Experience analysis.
- Market research and competitor analysis.
- Product analysis.
- Reputation management.
- Sentiments of customer reviews.
- Stock price prediction.
- Trend prediction.

Source: Wankhade, M., Rao, A.C.S. & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. Artificial Intelligence Review 55, 5731-5780, DOI: 10.1007/s10462-022-10144-1.

Typology of sentiment analysis applications

- Box office revenues for movies.
- Detecting hate speech.
- Detecting sensitive content webpages.
- Election results prediction.
- Finding customers attitude and trends.
- Politics and sociology.
- Prediction of sales performance.
- Public opinion polls.
- Recommendation systems.
- Sentiment-oriented question answering systems.
- Stock market prediction.
- User reviews.

Chauhan, P., Sharma, N., & Sikka, G. (2021). The emergence of social media data and sentiment analysis in election prediction. Journal of Ambient Intelligence and Humanized Computing, 12, 2601-2627, DOI: 10.1007/s12652-020-02423-y.

What is topic modelling?

*"Statistical technique for **revealing the underlying semantic structure** in large collection of documents."*

Source: Kherwa, P., & Bansal, P. (2020). Topic modelling: a comprehensive review. EAI Endorsed transactions on scalable information systems, 7(24), DOI: 10.4108/eai.13-7-2018.159623.

What is topic modelling?

- Retrieving the semantic structure of the texts, the purpose of topic modelling is usually to **group the texts with respect to their thematic motifs**.
- Topic modelling is a set of **machine-learning algorithms** based on either:
  1. linear algebra (semantic algorithms),
  2. estimation of parameters of the preassumed generative process (probabilistic algorithms),
  3. text embeddings clustering (transformer-based algorithms).

What is topic modelling?

- Topic modelling is usually perceived as an **unsupervised learning** problem.
- Still, it appears not the same as text clustering as conceptually **each document belongs to all of the topics with different weights (probabilities)**.

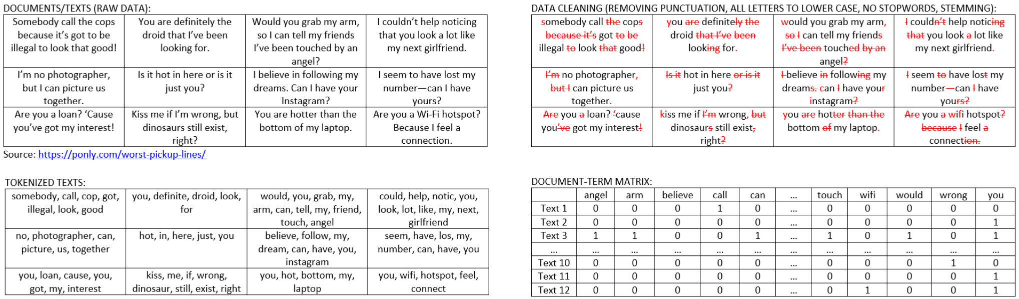
Topic modelling procedure

1. Aim of the study.
2. Data collection.
3. Textual data preprocessing: cleaning, tokenization, removing stopwords, stemming, lemmatization, frequency-based filtering, n-grams inclusion.
4. **Topic modelling, with hyperparameters optimisation**.
5. Drawing conclusions.

Topic model input

- In classic topic models (i.e., semantic and probabilistic algorithms), the input is a **Document-Term Matrix** (DTM).
- In DTM:
  - **row index stands for the analysed texts**,
  - **column index represents unique tokens** (words, pairs of words, etc.),
  - **each cell includes counts of tokens in individual documents** or their transformations.
- For the purpose of obtaining the topics, dimensionality of the DTM is reduced (in semantic algorithms) or its values are utilised to estimate probabilities in the preassumed generative process (in probabilistic algorithms).
- In the transformer-based topic models the input are sentence embeddings, eventually dimensionally reduced and clustered.

Textual data preprocessing: from raw text to DTM



DOCUMENTS/TEXTS (RAW DATA):

Somebody call the cops because it's got to be illegal to look that good!	You are definitely the droid that I've been looking for.	Would you grab my arm, so I can tell my friends I've been touched by an angel?	I couldn't help noticing that you look a lot like my next girlfriend.
I'm no photographer, but I can picture us together.	Is it hot in here or is it just you?	I believe in following my dreams. Can I have your Instagram?	I seem to have lost my number—can I have yours?
Are you a loan? 'Cause you've got my interest!	Kiss me if I'm wrong, but dinosaurs still exist, right?	You are hotter than the bottom of my laptop.	Are you a Wi-Fi hotspot? Because I feel a connection.

Source: https://ponly.com/worst-pickup-lines/

DATA CLEANING (REMOVING PUNCTUATION, ALL LETTERS TO LOWER CASE, NO STOPWORDS, STEMMING):

somebody call ~~the~~ cop~~s~~ ~~because it's~~ got ~~to be~~ illegal ~~to~~ look ~~that~~ good~~!~~	you ~~are~~ definite~~ly the~~ droid ~~that I've been~~ look~~ing~~ for.	would you grab my arm, ~~so I~~ can tell my friend~~s~~ ~~I've been~~ touch~~ed by an~~ angel~~?~~	~~I~~ could~~n't~~ help notic~~ing~~ ~~that~~ you look ~~a~~ lot like my next girlfriend.
~~I'm~~ no photographer, ~~but I~~ can picture us together.	~~Is it~~ hot in here ~~or is it~~ just you~~?~~	~~I~~ believe ~~in~~ follow~~ing~~ my dream~~s.~~ can ~~I~~ have you~~r~~ instagram~~?~~	~~I~~ seem ~~to~~ have los~~t~~ my number—can ~~I~~ have you~~rs?~~
~~Are~~ you ~~a~~ loan? ~~'~~cause you~~'ve~~ got my interest~~!~~	kiss me if ~~I'm~~ wrong, ~~but~~ dinosaur~~s~~ still exist~~,~~ right~~?~~	you ~~are~~ hot~~ter than the~~ bottom ~~of~~ my laptop.	~~Are~~ you ~~a~~ wifi hotspot~~?~~ ~~because I~~ feel ~~a~~ connect~~ion.~~

TOKENIZED TEXTS:

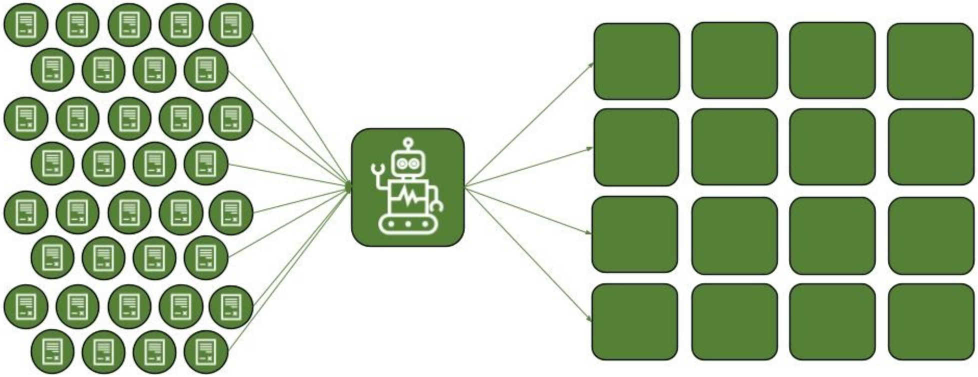
somebody, call, cop, got, illegal, look, good	you, definite, droid, look, for	would, you, grab, my, arm, can, tell, my, friend, touch, angel	could, help, notic, you, look, lot, like, my, next, girlfriend
no, photographer, can, picture, us, together	hot, in, here, just, you	believe, follow, my, dream, can, have, you, instagram	seem, have, los, my, number, can, have, you
you, loan, cause, you, got, my, interest	kiss, me, if, wrong, dinosaur, still, exist, right	you, hot, bottom, my, laptop	you, wifi, hotspot, feel, connect

DOCUMENT-TERM MATRIX:

	angel	arm	believe	call	can	...	touch	wifi	would	wrong	you
Text 1	0	0	0	1	0	...	0	0	0	0	0
Text 2	0	0	0	0	0	...	0	0	0	0	1
Text 3	1	1	0	0	1	...	1	0	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...
Text 10	0	0	0	0	0	...	0	0	0	1	0
Text 11	0	0	0	0	0	...	0	0	0	0	1
Text 12	0	0	0	0	0	...	0	1	0	0	1

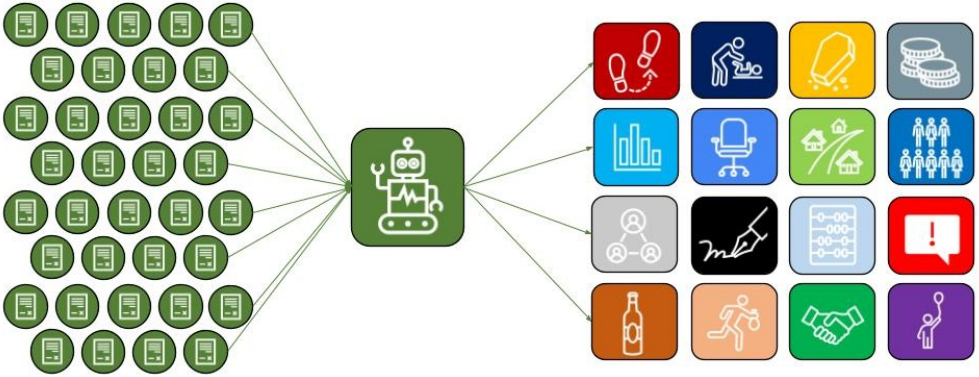
Source: own elaboration.

Topic model



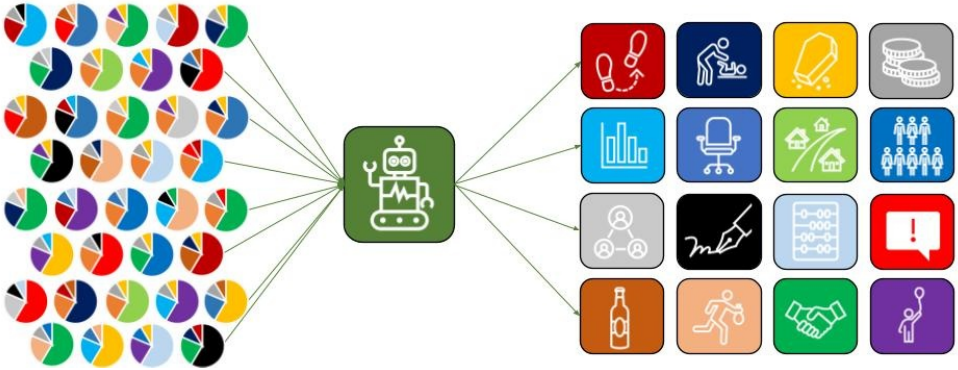
Source: own elaboration.

Topic model



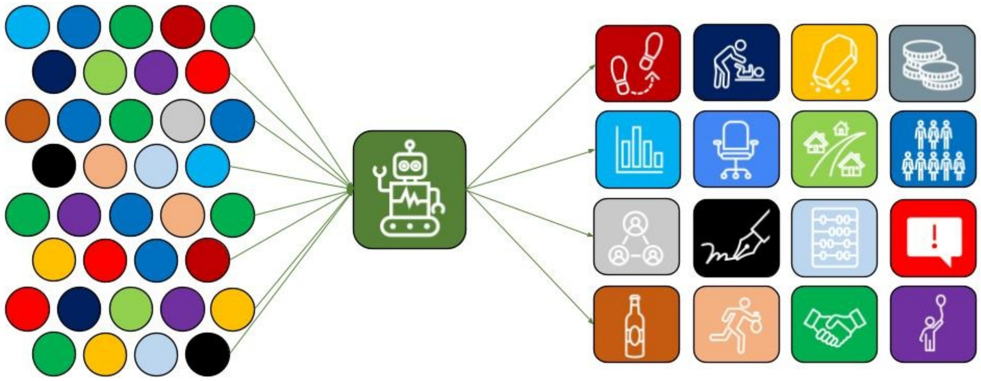
Source: own elaboration.

Topic model



Source: own elaboration.

Topic model



Source: own elaboration.

Topic model output

- Simply put, topic models return **topics**, i.e., representation of the semantic motifs observed within the analysed corpus (collection of texts).
- Typically, topics are represented by **keywords associated with weights** - the higher the weight, the stronger the association between the keyword and the topic. In probabilistic topic models, these weights are **probabilities that given keywords indicates given topics**.
- Moreover, the output also includes **weights indicating the degree to which a text belongs to a given topic**. Obviously, in probabilistic methods, these weights are probabilities.

Example applications of topic modelling in the literature

- Analysing scientific trends observed in the literature.
- Analysing the judiciary, specifically the case law.
- Assessing interpretability of reviews for recommendation.
- Gene expression profiles, toxicogenomics.
- Intelligent spam filtering.
- Medical studies.
- Research on patent data.
- Social media analysis with the most specific examples of political studies, opinons on Covid-19, renewable energy and finance.
- Social networks analysis.

Large language models

Large language models

Large language models are:

1. **pre-trained** on massive text datasets,
2. complex **neural network** architectures,
3. used for **various natural language processing tasks**.

LLMs are **pre-trained** on massive text datasets

- LLMs are trained on **vast and diverse datasets** comprising text from books, academic articles, news media, encyclopedias, online forums, websites, and social media platforms.
- This diversity of the training corpora is intended to provide the model with a broad and nuanced understanding of human language, **covering multiple domains, registers, and writing styles**.

LLMs are **pre-trained** on massive text datasets

Model	**Year**	**Training Corpus Name(s)**	**Corpus Size**
GPT	2018	BooksCorpus	∼0.7B tokens / ∼7K docs / ∼5GB
BERT	2018	BooksCorpus + English Wikipedia	∼3.3B tokens / ∼13K + 2.5M docs / ∼16GB
GPT-2	2019	WebText (Reddit-linked webpages)	∼10B tokens / ∼8M docs / ∼40GB
RoBERTa	2019	CC-News, OpenWebText, Stories, Books	∼30B tokens / doc count N/D / ∼160GB
GPT-3	2020	Common Crawl, WebText, Books, Wikipedia	∼300B tokens / ∼50M+ docs / ∼570GB
T5	2020	C4 (Colossal Clean Crawled Corpus)	∼250B tokens / ∼365M docs / ∼750GB
GPT-J	2021	The Pile (22 corpora incl. ArXiv, StackExchange, etc.)	∼400B tokens / ∼210M docs / ∼825GB
PaLM	2022	Web docs, books, Wikipedia, code	>780B tokens / doc count N/D / size N/D
LLaMA	2023	Common Crawl, GitHub, Books, ArXiv, Wikipedia	∼1.4T tokens / doc count N/D / size N/D
GPT-4	2023	Proprietary (web, books, code, etc.)	est. ∼2.0T tokens / doc count N/D / size N/D

Source: own compilation based on documentation and publications of various models.

LLMs are complex **neural network** architectures

- LLMs typically contain **millions to trillions of parameters**.
- For example, GPT-3 has 175 billion parameters, and GPT-4 has $\sim$1.7 trillion (est.) (see: next slide).
- These parameters represent the learned weights of the model, enabling it to capture nuanced patterns in language.

LLMs typically contain **millions to trillions of parameters**

Model	**Year**	**Parameters**
GPT	2018	117M
BERT	2018	110M
GPT-2	2019	1.5B
RoBERTa	2019	355M
GPT-3	2020	175B
T5	2020	11B
GPT-J	2021	6B
PaLM	2022	540B
LLaMA	2023	65B
GPT-4	2023	N/D (est. $\sim$1.7T)

Source: own compilation based on documentation and publications of various models.

Transformers

- LLMs are based on specific neural networks, i.e. the **Transformers**, which has become the foundation of modern natural language processing.
- Transformer is built around a **mechanism called self-attention**, allowing models to weigh the importance of different words in a sentence relative to each other.

Transformer architecture types

- There are three main types of Transformer architectures: encoder, decoder, and encoder-decoder.
- The **encoder** processes the input and **generates a text representation**; think of it as reading and understanding text.
- The **decoder generates text token by token**, based on input and context; think of it as writing the next word in a sentence,
- The **encoder-decoder does both, one after another** - first encodes the input and then decodes it to generate the output.

Transformer architecture types

Model	**Year**	**Architecture**	**Creator**
GPT	2018	Decoder	OpenAI
BERT	2018	Encoder	Google
GPT-2	2019	Decoder	OpenAI
RoBERTa	2019	Encoder	Facebook AI (Meta)
GPT-3	2020	Decoder	OpenAI
T5	2020	Encoder-Decoder	Google
GPT-J	2021	Decoder	EleutherAI
PaLM	2022	Decoder	Google
LLaMA	2023	Decoder	Meta
GPT-4	2023	Decoder	OpenAI

Source: own compilation based on documentation and publications of various models.

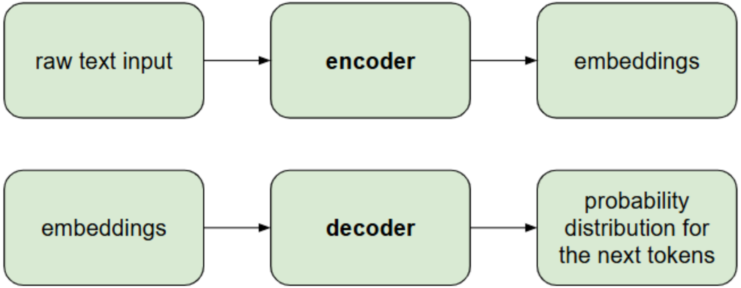
Encoders understand text, enable obtaining **text embeddings**

- Encoders use raw text input.
- During training, encoders learn to represent textual inputs as high-dimensional numerical vectors known as **embeddings**.
- These embeddings encode semantic, syntactic, and contextual information about the text, **capturing latent dimensions** such as sentiment, topic, discourse structure, or even authorial intent.
- Embeddings **can represent the meaning of words, tokens, sentences, or texts**.
- Even though the embeddings are state-of-the-art contextual representations of text, their **interpretability is limited**, especially when their **dimensionality is reduced**.

Decoders **generate text**, token by token

- The decoder does not create embeddings from raw input. Instead, it generates text, token by token, based on the embeddings.
- Taking embeddings as input, decoder computes probability distribution for the next tokens, eventually generating text.
- Optionally, one can construct a framework that generates tokens based on raw text input (encoder-decoder models).

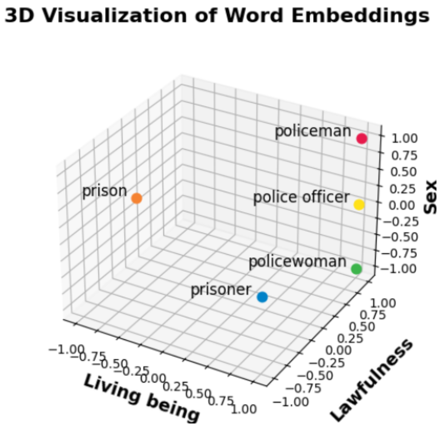
Encoder vs. decoder



Source: own elaboration.

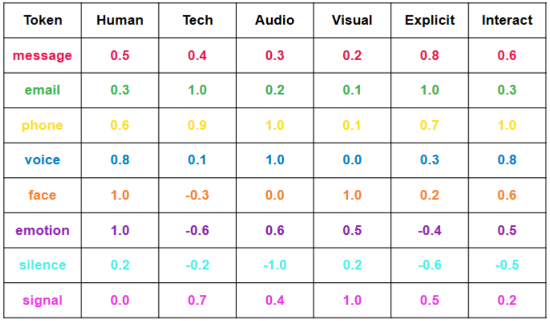
Word embeddings

Token	Living being	Lawfulness	Sex
policeman	1.0	1.0	1.0
policewoman	1.0	1.0	-1.0
police officer	1.0	1.0	0.0
prisoner	1.0	-1.0	0.0
prison	-1.0	0.0	0.0

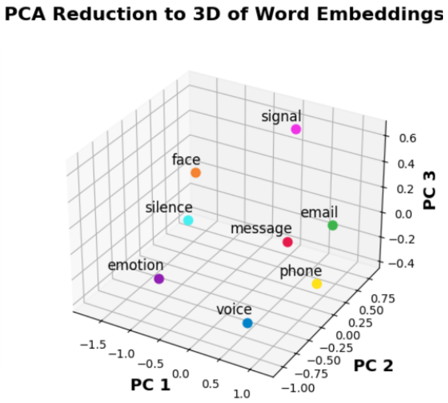


Source: own elaboration.

Word embeddings



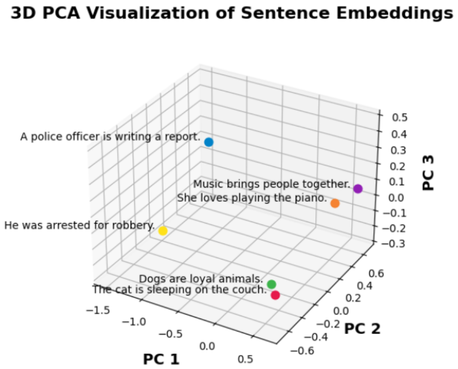
Token	Human	Tech	Audio	Visual	Explicit	Interact
message	0.5	0.4	0.3	0.2	0.8	0.6
email	0.3	1.0	0.2	0.1	1.0	0.3
phone	0.6	0.9	1.0	0.1	0.7	1.0
voice	0.8	0.1	1.0	0.0	0.3	0.8
face	1.0	-0.3	0.0	1.0	0.2	0.6
emotion	1.0	-0.6	0.6	0.5	-0.4	0.5
silence	0.2	-0.2	-1.0	0.2	-0.6	-0.5
signal	0.0	0.7	0.4	1.0	0.5	0.2



Source: own elaboration.

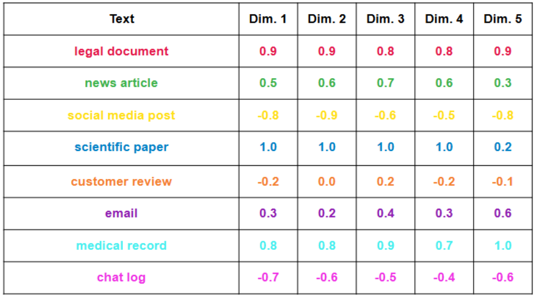
Sentence embeddings

Sentence	Dim. 1	Dim. 2	Dim. 3	Dim. 4	Dim. 5
The cat is sleeping on the couch.	0.9	0.1	0.0	0.7	0.0
Dogs are loyal animals.	1.0	0.0	0.1	0.9	0.0
He was arrested for robbery.	0.0	0.1	0.9	-0.7	1.0
A police officer is writing a report.	0.1	0.0	1.0	0.2	0.3
She loves playing the piano.	0.0	0.9	0.0	0.5	0.0
Music brings people together.	0.1	1.0	0.0	0.9	0.0

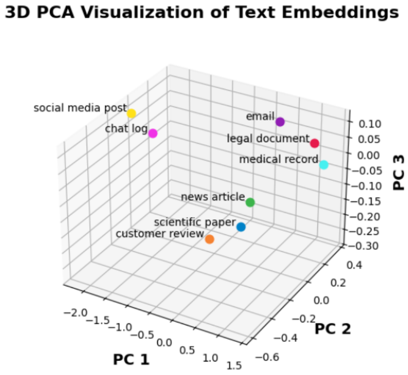


Source: own elaboration.

Text embeddings



Text	Dim. 1	Dim. 2	Dim. 3	Dim. 4	Dim. 5
legal document	0.9	0.9	0.8	0.8	0.9
news article	0.5	0.6	0.7	0.6	0.3
social media post	-0.8	-0.9	-0.6	-0.5	-0.8
scientific paper	1.0	1.0	1.0	1.0	0.2
customer review	-0.2	0.0	0.2	-0.2	-0.1
email	0.3	0.2	0.4	0.3	0.6
medical record	0.8	0.8	0.9	0.7	1.0
chat log	-0.7	-0.6	-0.5	-0.4	-0.6



Source: own elaboration.

Learning paradigm in large language models

LLMs take advantage primarily on the fact that they can be easily used to perform a wide range of natural language processing (NLP) tasks thanks to:

1. **fine-tuning**,
2. **zero-shot learning**,
3. **few-shot learning**.

Fine-tuning

- Fine-tuning is the process of training a pre-trained language model **on a specific dataset** to adapt it to a particular task or domain.
- It **improves accuracy for specific tasks**, customes tone and style. At the same time, however, it is made **less flexible** to new tasks unless retrained.
- Simply put, it works the following way:
  1. Start with a general pre-trained model (e.g., GPT-3).
  2. Provide a large set of labeled examples (hundreds to thousands).
  3. Train the model further on this data.
  4. The model becomes specialized for that task (e.g., legal writing, customer service chat, medical Q&A).

Zero-shot learning

- In zero-shot learning, a language model is given **a task without any examples**. It relies only on a prompt or natural language instruction to perform the task.
- It works well in ad hoc answering and **text classification without labeled data** (e.g., named entity recognition, sentiment analysis).
- However, one cannot be sure about the accuracy, i.e., **no labels imply no error measures**. In practice, when using zero-shot, a labelled test set is considered for the model assessment.

Few-shot learning

- In few-shot learning, the model is given **a small number of examples in the prompt**, so it can better understand the desired format or logic of the task.
- Usually, by seeing similar examples, the model can more easily generalize and produce accurate results.
- Again, one cannot directly measure the accuracy of such an algorithm, unless a labelled test set is included.

Applications of large language models

- Code generation.
- Conversational agents.
- Creative writing assistance.
- Education and tutoring.
- Information extraction.
- Legal support.
- Machine translation.
- Medical support.
- Question answering.
- Sentiment analysis.
- Text classification.
- Text generation.
- Text summarization.
- Topic modelling.

#NLPath

Related courses:

Information extraction - Mondays, 4:45 p.m.
Large language models - Mondays, 6:30 p.m.
Sentiment analysis - Wednesdays, 4:45 p.m.
Topic modelling - Thursdays, 6:30 p.m.

Maciej Świtała, PhD

Thank you for your attention!

Maciej Świtała, PhD
`ms.switala@uw.edu.pl`