ACADEMIC YEAR 2026/27

Text Mining & Search [DS] / Information Retrieval [CS]

Text Processing

TEXT MINING & SEARCH

Prof. Marco Viviani

University of Milano-Bicocca

Department of Informatics, Systems and Communication

INFORMATION RETRIEVAL

Dr. Georgios Peikos

University of Milano-Bicocca

Department of Informatics, Systems and Communication

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Ikr3 Lab — information and knowledge representation, retrieval, and reasoning LAB

Text Processing

Introduction

Introduction – What is Text Processing?

Basic text processing is often the first step* in any text mining application and consists of several layers of simple processing such as tokenization, lemmatization, normalization, or more advanced….[2, Gabe, 2017].

*That used to be the case, but things have since changed.

Text processing is a computational discipline concerned with the manipulation, analysis, transformation, and rendering of textual data using programs and algorithms.

It covers operations performed on raw character sequences, un/structured documents, and human-language content, ranging from low-level byte handling and encoding conversion to high-level semantic interpretation. [IEEE, 2026].

[1] https://technav.ieee.org/topic/text-processing/
[2] Gabe Ignatow and Rada Mihalcea. “An Introduction to Text Mining”. Apple Books, 2017.

Introduction – So, what can be described as Text Processing?

- Text processing is the use of computational methods to prepare, represent, transform, and analyze natural language text.
  - Such computational methods can be Rule-based methods, Statistical methods, ML methods, and Neural methods.
- These methods support applications such as information retrieval, text classification, information extraction, translation, and summarization.
- Text processing converts raw text into units that a computational system can use.
- The appropriate processing steps depend on the language, task, and the model that will be used for the downstream task and analysis.

Introduction – How do we do Text Processing?

- Define the goal. Identify what the system should do, such as retrieve documents, classify text, or generate summaries. [Projects]
- Collect and extract text. Obtain text from sources such as web pages or PDFs. [Today's Lesson]
- Prepare the text (text preprocessing). Apply cleaning, normalization, and tokenization according to the task and model. [Today's Lesson]
- Represent the text. Convert it into a suitable form, such as word counts, TF–IDF vectors, token IDs, or embeddings. [Next Lessons in PART I]
- Apply algorithms or models. Use rules, statistical methods, or learned models to analyze or transform the text into insights and achieve your goal. [Part II of the Courses]
- Evaluate and refine the pipeline. Check whether the outputs meet the goal and adjust the processing steps when needed. [Part II]

Introduction – Is Text Processing important?

- It improves textual quality, as cleaning can remove formatting noise, irrelevant content (e.g. ads in web pages), and fix extraction errors.
- It reduces unnecessary textual variation, normalization can help a system treat equivalent forms consistently, or reduce the space needed to storage the information.
- It is necessary to prepares text for algorithms and models, i.e. create numerical representations of texts (computers do not process text).
- It affects the information available to the system! Removing punctuation, capitalization, or words can discard useful meaning.
    - ➢ So, preprocessing must match the task (and is your job to decide how it should be done correctly).
- In 2026, text (pre)processing remains important, but its role has changed. New models have different requirements for text processing!

Introduction – Texts we produce in our everyday lives

Digital & Media Content

- Emails
- Insurance claims
- News articles
- Web pages
- Patent portfolios
- User generated content in Social media
  (course on Social Media Analytics)

Enterprise & Specialized

- Customer complaint letters
- Contracts
- Transcripts of phone calls with customers
- Technical documents
- Scientific papers
- Health related information

Introduction – What are the units of textual information?

- Characters are individual letters, digits, punctuation marks, and other symbols.
- Morphemes are the smallest units that carry meaning, such as un-, help, and -ful in unhelpful.
- Words are linguistic units such as “coffee” and “mouse”.
  - Tokens are the units a computer system uses to process text. A tokenizer produces these units, which may be words, subwords, characters, or punctuation.
- Phrases are groups of words that function together, such as a relevant document.
- Sentences are grammatical units that express a statement, question, or command.
- Paragraphs and passages are groups of sentences that develop a topic or provide a segment of text.
- Documents are complete textual items, such as articles, emails, or reports.
- Corpora or collections are sets of documents used for analysis, training, or information retrieval.

Units of Textual Information

Characters & Morphemes

Characters & Morphemes – Encoding

- Character encoding defines how characters are represented as bytes that computers can store and transmit.
  - For examples of character level encoding, check this link.
- Different encodings emerged to support different languages, writing systems, and computer environments.
  - UTF-8 is the dominant encoding today. W3Techs reports that 99.1% of websites with a known encoding use UTF-8, as of September 2026.
    - UTF-8 preserves ASCII byte values. Basic Latin letters, digits, and common punctuation require one byte each.
  - Other encodings include UTF-16, ISO-8859-1, Windows-1252, and regional encodings such as Shift JIS.

Characters & Morphemes – How are characters represented?

Character	Encoding	Number of bytes	Bytes in hexadecimal	Bytes in binary
A	ASCII / UTF-8	1	41	01000001
é	ISO-8859-1	1	E9	11101001
é	UTF-8	2	C3 A9	11000011 10101001
Ω	UTF-8	2	CE A9	11001110 10101001
😊	UTF-8	4	F0 9F 98 8A	11110000 10011111 10011000 10001010

Characters & Morphemes – Encoding

- The same character can have different byte representations in different encodings.
- In UTF-8, different characters can require different numbers of bytes.
- Therefore, the number of characters is not necessarily the number of bytes.

Characters & Morphemes – What happens when we use the wrong encoding?

- Datasets collected from different sources may use different encodings. So, always pay attention to it.
- Incorrect decoding can corrupt names, words, and symbols before an NLP or IR system processes them.
- This can affect tokenization, entity recognition, classification, and document matching.
- Decode each source correctly, then save the combined dataset in a consistent encoding, usually UTF-8.
- Avoid deleting or replacing invalid bytes. Information may be lost.

Units of Textual Information

Words and Tokens

Words and Tokens – Words and their information value

- Words contribute different kinds of information, including topics, entities, actions, and grammatical relationships.
- Some words help distinguish a document's topic, such as diabetes, election, or inflation.
- Common words, such as the, a, and of, often provide less information about the topic.
- A word's importance depends on its **context** and the **task**.

Which words help you identify the topic of this sentence?
The patient received treatment for diabetes.

Which word changes the meaning most?
The patient did not receive treatment for diabetes.

Words and Tokens – Stopwords

- Stopwords are words that a system chooses to exclude from a particular analysis or representation.
- They often include frequent function words, such as the, a, and, and of.
- Removing them can reduce the size of some text representations.
- There is no universal stopword list. The choice depends on the language, domain, and task.
- Removing words can also discard useful information, including negation and relationships.

Would you remove the same words for topic analysis and sentiment analysis?
Who would like to give an example?

Words and Tokens – Other properties of words that matter in TM & Search

Frequency & Specificity

system vs *chemotherapy*

Frequent words distinguish topics less effectively. Rare words can be informative, but can also be errors.

Ambiguity

bank in finance or geography

The same word can express completely different meanings depending on context.

Synonymy

car and *automobile*

Different distinct words can express highly similar meanings.

Variation in Word Forms

connect, connected, connecting

Related forms may need to be associated while preserving grammatical info.

Domain Dependence

positive in review or test

Word interpretation heavily depends on the target subject and context.

Multiword Meaning

hot dog, New York

Meaning may belong to a group of words rather than an individual word.

Units of Textual Information

Sentences, Phrases, Paragraphs

Sentences, Phrases, Paragraphs – Words and their information value

- Phrases express concepts and local relationships. In *"severe side effects"*, the word severe qualifies side effects. Processing the words separately can lose this relationship.
- Sentences express relationships between entities, actions, and events. *"The drug treats the disease"* and *"The disease resists the drug"* contain similar words but express different relationships.
- Paragraphs connect statements into a broader idea. Information in one sentence may explain, qualify, or contradict information in another.
- The unit we analyze determines how much context is available. Smaller units provide focused information. Larger units can resolve ambiguity but may contain several ideas.
  - In Information Retrieval we work on word, sentence, passage or paragraph (chunk), and document levels aiming to retrieve relevant information to a user's query.
  - Text Mining tasks require different levels of analysis, such as phrases for extracting concepts, sentences for extracting relationships, and documents for topic classification.

Units of Textual Information

Documents

Documents – Overview

- A document is an item treated as a unit by a system, such as a news article, email, medical report, web page, or book.
- A document can contain text, has a specific structure, contain other media, and be accompanied by **metadata**.
  - Structure includes titles, headings, paragraphs, lists, and tables. It helps organize and interpret the content.
- Presentation concerns how the content appears, including fonts, colours, and page layout. Presentation can also communicate structure.



Documents – Characteristics (Text, Structure, Media, Metadata)



Documents – Metadata

- Metadata describes a document and helps systems organize, filter, retrieve, and interpret it.
  - **Descriptive metadata** includes the title, author, publication date, language, and document type.
- **Semantic metadata** describes what the document concerns, using topics, categories, or controlled keywords associated with a document.
- Metadata may be supplied by authors or publishers, or generated automatically by another system.
- Metadata can be missing, inconsistent, or incorrect. Its reliability must be considered.

Is the publication date of a document part of the article's content, its metadata, or both?

Documents – From files to processable input

- Documents arrive in different formats, such as TXT, HTML, DOCX, PDF, and email messages.
- Systems must extract their content and convert it into a representation suitable for processing.
  - Extraction should preserve useful structure, including headings, table relationships, and reading order.
- Scanned documents require optical character recognition (OCR) to recover text from images.*
- Extraction errors can affect every subsequent step, including classification, information extraction, retrieval, and summarization.

If we extract every word from a table but lose its rows and columns, have we preserved its meaning?

Documents – File Format and Typical Errors

Source	Typical problem	Processing decision
HTML web page	Menus and advertisements mix with the article	Extract main content and headings
Digital PDF	Columns or tables appear in the wrong order	Check reading order and structure
Scanned page	The page contains an image of text	Use OCR and **inspect errors**
DOCX or messages	Formatting, quotes, and repeated signatures	Retain useful boundaries and fields

Units of Textual Information

Document Collection (Corpus)

Document Collections (Corpora) – Overview

- A document collection, or corpus, is a set of documents brought together because they share some common characteristics. For instance, are the sources doctors read to treat patients.
  - Organizations accumulate collections through their activities. A hospital, for example, produces medical reports, clinical notes, and discharge summaries.
- Collections can be created by gathering existing files, exporting databases, crawling web pages, or selecting documents for a particular study.
  - Text Mining uses collections to identify patterns, extract information, and compare documents.
  - Search uses collections to identify documents or passages that satisfy a user's information need.
- A collection's language, domain, time period, and coverage determine what information a system can access.

Document Collections (Corpora) – From Document Collections to Vocabulary

- A collection contains many documents, but their texts often share the same words.
- To describe its size, we can count both all token occurrences and the distinct tokens that form its vocabulary.
  - N is the total number of tokens, including repetitions.
  - V is the collection's vocabulary. |V| is the number of distinct tokens.
- Adding documents increases the token count, but does not necessarily introduce new vocabulary.

Collection	Number of tokens N	Vocabulary size |V|
cats chase mice	3	3
Add document: cats chase birds	6	4
Add document: birds chase cats	9	4

Document Collections (Corpora) – Vocabulary Growth Heaps' Law

- As a collection grows, new documents introduce new vocabulary, but many tokens repeat words already present.
- Heaps' law estimates how many distinct tokens we expect in a collection, given its total number of token occurrences.

$$|V| = kN^{\beta}$$

  - N is the total number of token occurrences, including repetitions.
  - |V| is the estimated number of distinct tokens.
  - k controls the scale of the estimate.
  - β controls how quickly the vocabulary grows, where $0 < \beta < 1$:
    - $\beta = 0.5$ means vocabulary grows proportionally to the square root of the token count.
- We estimate k and β based on subsets of a collection.
- For Reuters-RCV1, the fitted parameters were $k \approx 44$ and $\beta \approx 0.49$. At approximately 1 million tokens, the law predicted 38,323 distinct terms, compared with 38,365 actually observed.

Text Processing

Word Tokenization - Tokens

Word Tokenization – What Is Word Tokenization?

Core Principles

- **Word-level units:** Divides input text into individual **tokens**, often separating punctuation into discrete elements.
- **Boundary identification:** Tokenizers rely on language-specific rules, heuristics, or **machine learning** to detect boundaries.
- **Whitespace baseline:** Splitting on whitespace serves as a naive starting point for space-delimited languages.
- **Limitations:** Spaces do not capture all word boundaries, and punctuation doesn't always signal a split.

Example: Space-Based Tokenization

Original Input Sentence:

"L'università di Milano è aperta."

Resulting Tokens (Split on spaces):

```
[L'università] [di] [Milano] [è] [aperta.]
```

Tokenization Challenges & Decisions

- **Apostrophes & Clitics:** Should L'università remain one token or split into L' + università?
- **Trailing Punctuation:** Should the final period in aperta. be isolated as a separate token?

Word Tokenization – Apostrophes, Hyphens, and Multiword Expressions

Example	Decision
l'acqua	Keep one token or separate l' and acqua?
j'aime	Keep one token or separate j' and aime?
Finland's capital	Keep Finland's or separate Finland and 's?
O'Connor	Would splitting at the apostrophe break a name?
Hewlett-Packard	Preserve the company name or split at the hyphen?
state-of-the-art	Preserve the compound or identify its individual words?
New York	Two word tokens, but one named entity?

Word Tokenization – Numbers and Special Expressions

Example	Information to preserve
Ph.D.	The abbreviation
€45,55	The amount, decimal separator, and currency
01/02/2026	The date components. Its interpretation also depends on locale.
+39 02 6448 1234	A telephone number containing spaces
B-52	An identifier whose hyphen matters
rossi@example.com	An email address
https://example.org/news	A URL
#machinelearning	A hashtag and its text

- Numbers and symbols can carry useful information. Used to be neglected in indexing.
- Tokenization should preserve the components needed by the application.

Word Tokenization – Word Boundaries Across Languages

Language	Example	Challenge
Italian	l’università	An apostrophe connects an elided article to a noun.
German	Lebensversicherung — “life insurance”	One written word contains meaningful components. Compound splitting is an additional processing choice.
Chinese	我喜欢北京。— “I like Beijing.”	Spaces do not mark word boundaries.
Japanese	私は本を読む。— “I read a book.”	Different scripts occur together, without spaces between words.
Thai	ฉันรักแมว — “I love cats.”	Word boundaries must be identified without spaces.

Chinese: 我 · 喜欢 · 北京 · 。-
Japanese: 私 · は · 本 · を · 読む · 。
Thai: ฉัน · รัก · แมว

In such cases, a tokenizer uses dictionaries or learned models to identify word boundaries because spaces generally do not separate individual words.

Word Tokenization – Today

- Word tokenization remains useful for word frequency analysis, linguistic annotation, information extraction, and lexical search.
- Nowadays exist libraries provide language-specific tokenizers. We can use existing tools rather than build every rule ourselves.
- Customization still matters. Medical terms, product codes, abbreviations, and informal text may require additional rules.
- There is no universal tokenizer. The appropriate boundaries depend on the language, collection, and task. Tools such as spaCy support language-specific rules and exceptions.
- Modern systems do not all use word-level tokens. Word tokenization remains one approach within the broader range of tokenization methods. Let's see what they use!
  - Here is a resource to start learning more and more about tokenization today.

Text Processing

Subword Tokenization and its Methods

Subword Tokenization – Learning Units from Data

- Rather than relying only on:
  - Whitespace segmentation, which keeps words as separate units.
- Use a text collection to **learn** which pieces of texts should become tokens.
  - Frequent text sequences can become tokens, including whole words or parts of words.
  - Other words can be represented by combining smaller tokens.
  - This approach is called subword tokenization.
- Subword tokenization balances vocabulary size and sequence length. Words absent from the vocabulary can often be represented using known pieces.
- It is widely used in Transformer-based language models.

Subword Tokenization – Methods

Main method	Basic idea
BPE — Byte Pair Encoding	Repeatedly merges frequent adjacent units.
WordPiece	Learns a vocabulary of pieces and segments words using those pieces. Used in BERT.
Unigram	Assigns probabilities to pieces and selects a likely segmentation.

Subword Tokenization – How Does BPE Work?

- First, we learn the tokenizer from a text collection.
  - Start with small units, such as characters. This is a training set of the data collection.
  - Count how frequently adjacent pairs occur.
  - Merge the most frequent pair into a new token.
  - Repeat until the target vocabulary size is reached.

BPE example — count adjacent pairs

Training item unit Word	Item's Occurrences in training dataset	Initial units created
play	5	p / l / a / y
playing	3	p / l / a / y / i / n / g
played	2	p / l / a / y / e / d
player	2	p / l / a / y / e / r

The sequence p / l / a / y appears in all four words and occurs 12 times. BPE can therefore learn it through repeated merges.

Subword Tokenization – How Does BPE Work?

- First, we learn the tokenizer from a text collection.
  - Start with small units, such as characters. This is a training set of the data collection.
  - Count how frequently adjacent pairs occur.
  - Merge the most frequent pair into a new token.
  - Repeat until the target vocabulary size is reached.
- Illustrative BPE steps
  - Merge p + l → pl, pl is the new token, i.e. the tokenizer will keep these 2 characters together
  - Merge pl + a → pla, pla is the new token, i.e. the tokenizer will keep these 3 characters together
  - etc..
  - With further merges and data, the tokenizer might learn ing, ed, or er.
- Finally, apply this merging rules you learned into new data. So,

Subword Tokenization – Final Considerations

- Subword tokenization learns reusable units from a training collection.
  - Frequent words or sequences may remain complete tokens, while rare or unseen words are divided into smaller units.
- The resulting tokens depend on the training data, target vocabulary size, and tokenization algorithm.
  - Subword tokens do not necessarily correspond to morphemes or other meaningful linguistic units.
- The same text can be divided differently by different tokenizers.
- A pretrained model must use its associated tokenizer, because its token vocabulary and numerical IDs were fixed during training.
  - Numerical IDs are integer numbers assigned to the tokens in a tokenizer's vocabulary. For instance the sequence ing can take id 201
- Subword tokenization is widely used by current Transformer models, while word-level tokenization remains useful in lexical IR and linguistic analysis.

Text Processing

Document Segmentation

Document Segmentation – Why Divide a Document into Chunks?

- Many of the Text Mining and Retrieval applications we will be using in this course, require us to work on document level, e.g. sentences, paragraphs, chunks of texts, or even the whole text.
- A document can contain several topics, sections, and types of information.
- A chunk is a segment of text processed as a unit. It may contain one or several sentences, paragraphs, or sections.
  - Chunks allow systems to retrieve, classify, or analyze specific parts of a document.
  - They also help accommodate model input limits and reduce the amount of text processed at once.
- The goal is to preserve enough context without including too much unrelated information.

Document Segmentation – How Can We Create Chunks?

Approach	How it works	Main limitation
Fixed-size segmentation	Split after a specified number of words or tokens.	May cut through sentences or ideas.
Sentence-based segmentation	Group consecutive complete sentences.	Related information may cross chunk boundaries.
Structure-based segmentation	Use paragraphs, headings, and sections.	Sections vary in length, and extracted structure may be unreliable.
Topic-based segmentation	Identify changes in subject and split around them.	Topic boundaries can be difficult to detect.

Document Segmentation – Why Divide a Document into Chunks?

- Smaller chunks provide focused information, but may lose context.
- Larger chunks preserve more context, but may combine unrelated information and increase processing cost.
- Overlap repeats some text between adjacent chunks to reduce context loss at boundaries. It also increases storage and processing.
- Preserve document identifiers, section headings, and positions so each chunk can be connected to its source.
- Evaluate chunking choices through the downstream task. There is no universally best chunk size.
- Document chunking is particularly useful for RAG applications.

Text Processing

Word Normalization

Word Normalization – Creating a Common Representation

- Normalization maps selected surface forms (e.g. words) to a standard representation.
  - Forms mapped to the same representation are an equivalence input for a system.
- Normalization reduces variation when counting, comparing, or matching terms.
  - For Information Retrieval, word level normalization, contributes to the minimization of the information storage requirements (less terms in a collection's vocabulary, i.e. smaller index size).
- Input equivalence is a text processing decision, that should be done carefully.

Should Apple and apple be considered as equivalent terms by a system?

Word Normalization – Examples

Surface forms	Possible common representation
Search, SEARCH, search	search
U.S.A., USA	USA
e-mail, email	email
colour, color	color

Example	Decision or risk
si / sì	Accent removal merges different Italian meanings.
Müller / Mueller	These can be alternative spellings of a name.
01/02/2026	Could mean 1 February or 2 January.
2026-02-01	An unambiguous standard representation once the date is interpreted.

Word Normalization – Normalization Depends on Language and Context

- Case folding reduces capitalization differences, but can merge forms such as US and us.
- Periods and hyphens may be removed or standardized when they represent writing variants.
- Accents and umlauts require language-specific decisions.
- Date normalization requires interpreting the original format before producing a standard form.
- Language detection can help select rules, but language alone may not resolve regional conventions.

Word Normalization – Today

- Normalization remains useful for lexical search, corpus analysis, and matching structured information.
  - In Information Retrieval, to perform lexical matching we need to do specific text normalization processes.
- Modern models can handle some variation, but this does not make every normalization step unnecessary.
- For pretrained models, follow their expected preprocessing rather than automatically removing capitalization, accents, or punctuation.
- Preserve distinctions that carry meaning, and retain original text when possible. Transformer-based models today take as input raw text, as they have their internal text normalization and tokenizations mechanisms.
- In short, normalize differences that obstruct the task, while preserving differences that contribute useful information.

Text Processing

Lemmatization and Stemming

Lemmatization and Stemming – Overview

- A word can appear in different forms, such as connect, connected, and connecting.
  - Treating every form separately can hurt the efficiency and effectiveness of our text mining and retrieval tasks.
    - A query containing “connect” could also match a document containing “connected”.
- Stemming and lemmatization map selected word forms to a shared representation.
- They are beneficial for lexical search, topic analysis, and corpus statistics, i.e. when grammatical differences are less important.
- They are optional processing choices.
  - Removing distinctions such as tense may be unsuitable for some tasks.
  - For example, in modern transformer-based models, we do not perform this type of text processing.

Lemmatization – Finding the Dictionary Form

- Lemmatization maps a word to its lemma, or dictionary form.
- It uses dictionaries or linguistic rules.
- It can handle irregular forms that simple suffix removal cannot resolve.

Word in context	Lemma
The children are playing. → children	child
She went home. → went	go
They are studying. → studying	study
I gatti dormono. → gatti	gatto
Noi mangiamo. → mangiamo	mangiare

Stemming – Applying Word-Reduction Rules

- Stemming applies rules to reduce words to a common stem, often by removing or modifying endings.
  - The resulting stem does not have to be a valid dictionary word.
- Stemmers usually process words without sentence context.
  - They can group inflected forms and some derived forms.

Input	Stem
connected, connecting, connection	connect
studies, studying	studi
agreed	agre
traditional	tradit

Lemmatization and Stemming – Methods

- Tools include spaCy's lemmatizer, Stanza's lemmatizer, and NLTK's WordNetLemmatizer.
  - spaCy supports lookup and rule-based approaches, while Stanza also uses a neural model.
- Common stemmers include Porter, Snowball, and Lancaster.
  - Snowball provides language-specific stemmers, including Italian, French, and German.

Lemmatization and Stemming – Which Approach Should We Choose?

- Choose stemming when simple grouping is sufficient, such as in some lexical retrieval pipelines.
- Choose lemmatization when interpretable base forms and grammatical distinctions matter.
- Both can make errors. Stemming may merge unrelated words or fail to merge related forms; lemmatization may select the wrong analysis.
- Apply compatible processing to texts that must be compared, such as documents and queries must be processed the same way!
- The decision is based on the characteristics of the task and its context.
- Commonly, when your text will be processed by a transformer-based model, e.g. LLM, the input is just the original form of the text.

Text Processing

Conclusions and Main Takeaways

Text Processing – Conclusions and Main Takeaways

- Text processing prepares and transforms text into representations that systems can use for search, analysis, and generation.
- The unit of analysis matters. Words, sentences, chunks, and documents provide different amounts of information and context.
- Start by recovering the text accurately from the source document. Preserve the structure and metadata needed to interpret it.
- Tokenization defines the processing units. Their boundaries depend on the language, data, and method.
- Normalization, stemming, and lemmatization reduce variation, but can also remove useful distinctions.
- There is no universal preprocessing pipeline. Choose steps for the task and model, apply them consistently, and evaluate their effect.
- Every processing choice changes the information available to the system. In this course, we will help you learn how to make the right ones!
- For LLMs and Transformer models, use the associated tokenizer and expected input format. Avoid automatically removing punctuation, capitalization, or stopwords that may carry useful information.

Text Processing – Main Takeaway

Read examples of your data manually before choosing processing steps.

Consider the language, context, information needed for the task, and input requirements of the model.

Contacts

Dr. Georgios Peikos

georgios.peikos@unimib.it

Academic Year 2026/27

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Ikr3 Lab information and knowledge representation, retrieval, and reasoning LAB