Natural Language Processing

Christian Rossi

Academic Year 2024-2025

Abstract

This course introduces students to the challenges and methodologies related to the analysis and production of natural language sentences, both written and spoken. The course explores the current role of stochastic models and Deep Learning, as well as new opportunities to combine traditional, formally based models with stochastic models. Topics covered include morphology, syntax, semantics, pragmatics, voice, prosody, discourse, dialogue, and sentiment analysis.

The course includes practical exercises where students can test the models and techniques presented in the lectures. Applications explored during the course will include: human-machine and human-human interaction analysis based on language; linguistic and prosodic analysis and generation for rehabilitation; pattern recognition and research for sentiment analysis in critical interactions; complexity analysis in text and overall spoken communication; and the development of user profiles that account for expression preferences in forensic, educational, and clinical contexts.

Contents

CHAPTER 1

Introduction

1.1 Natural language

The origins of spoken language are widely debated. Estimates range from as early as 2.5 million years ago to as recent as 60,000 years ago, depending on how one defines human language.

The development of written language, however, is more clearly documented. The first known writing systems emerged in Mesopotamia (modern-day Iraq) around 3500 BCE. Initially, these were simple pictograms representing objects, but over time, they evolved into abstract symbols representing sounds, paving the way for more complex communication.

The characteristics of human language are as follows:

- *Compositional*: language allows us to form sentences with subjects, verbs, and objects, providing an infinite capacity for expressing new ideas.
- *Referential*: we can describe objects, their locations, and their actions with precision.
- *Temporal*: language enables us to convey time, distinguishing between past, present, and future events.
- *Diverse*: thousands of languages are spoken worldwide, each with unique structures and expressions.

1.1.1 Natural Language Processing

One reason to care about Natural Language Processing is the sheer volume of human knowledge now available in machine-readable text. With the rise of conversational agents, human-computer interactions increasingly rely on language understanding. Furthermore, much of our daily communication is now mediated by digital platforms, making Natural Language Processing more relevant than ever.

However, Natural Language Processing is a challenging field. Human language is highly expressive, allowing people to articulate virtually anything—including ambiguous or nonsensical statements. Resolving this ambiguity is one of the core difficulties in computational linguistics. Moreover, meaning can be influenced by pronunciation, emphasis, and context, making interpretation even more complex. Fortunately, language is often redundant, allowing for error correction and inference even when mistakes occur.

1.1.2 History

The field of Natural Language Processing has its roots in linguistics, computer science, speech recognition, and psychology. Over time, it has evolved through various paradigms, driven by advancements in formal language theory, probabilistic models, and Machine Learning.

During World War II, early work in Natural Language Processing was influenced by information theory, probabilistic algorithms for speech, and the development of finite state automata.

Between 1957 and 1970, two primary approaches emerged. The symbolic approach, based on formal language theory and AI logic theories, focused on rule-based processing. Meanwhile, the stochastic approach leveraged Bayesian methods, leading to the development of early Optical Character Recognition systems.

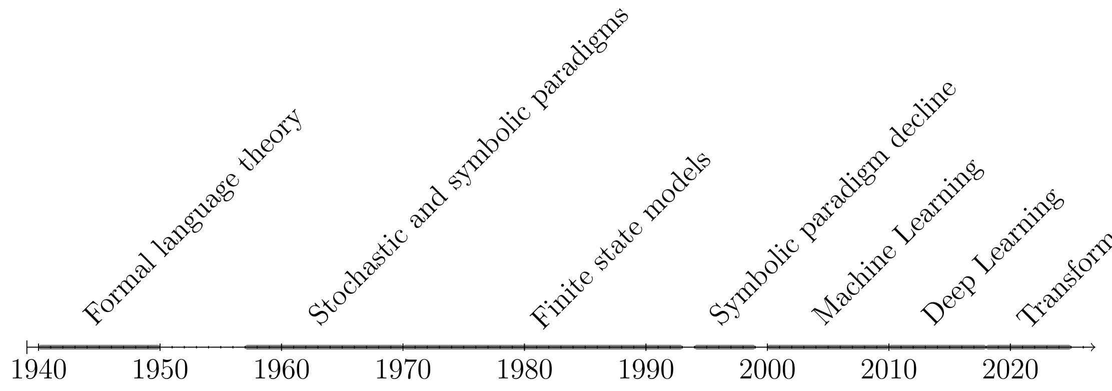
From 1970 to 1993, the focus shifted toward empirical methods and finite-state models. Researchers worked on understanding semantics, discourse modeling, and structural analysis of language.

By the mid-1990s, symbolic approaches began to decline, and the late 1990s saw a surge in data-driven methods, fueled by the rise of the internet and new application areas.

The 2000s marked a deep integration of Machine Learning into Natural Language Processing. The increasing availability of annotated datasets, collaboration with Machine Learning and high-performance computing communities, and the rise of unsupervised systems solidified empiricism as the dominant paradigm.

From 2010 to 2018, Machine Learning became ubiquitous in Natural Language Processing, with Neural Networks driving major advances in conversational agents, sentiment analysis, and language understanding.

Since 2018, the field has been revolutionized by Transformer architectures. Pretrained language models, such as BERT and GPT, have enabled transfer learning at an unprecedented scale, leading to the rise of massive online language models.



1.2 Text analysis

Before applying any NLP algorithm, it is important to standardize and clean the text. Preprocessing ensures consistency and improves the accuracy of downstream tasks. Common cleaning steps include:

- Before tokenization, removing non-content information such as HTML tags, converting text to lowercase, and eliminating punctuation.
- After tokenization, filtering out stopwords (extremely common words that add little meaning), removing low-frequency words, and applying stemming or lemmatization to reduce vocabulary size.

1.2.1 Text mining

Text may need to be extracted from various sources, each with its own challenges:

- Textual documents, HTML pages, and emails often contain formatting elements that should be removed.
- Binary documents are more complex to process. In PDFs, text may be laid out in multiple columns, requiring careful reconstruction. If all PDFs follow a consistent format, handwritten rules may suffice; otherwise, Machine Learning techniques may be needed.
- Scanned documents require Optical Character Recognition, which relies on Deep Learning to convert images to text. However, Optical Character Recognition is not flawless and can introduce recognition errors.

1.2.2 Characters encoding

When storing and processing text, different character encodings must be considered.

ASCII encoding represents only 128 characters, mapping letters and symbols to numerical values. While sufficient for basic English text, it cannot handle many linguistic symbols.

UTF-8 encoding supports over 149,000 Unicode characters, covering more than 160 languages. Unicode is essential for processing texts that use non-Latin scripts. It also preserves special characters, such as diacritical marks in Italian and English.

1.2.3 Tokens

In many languages, spaces serve as natural boundaries between words, making tokenization straightforward. However, if spaces weren't available, we would need alternative methods to segment text. Since they do exist in most languages, they are commonly used for tokenization.

Despite this, tokenizing text isn't always simple. Hyphenated words can pose challenges, as some languages construct long, compound words that may need to be split for effective processing. In other cases, meaningful units are spread across multiple non-hyphenated words in multiword expressions. Additionally, punctuation cannot always be blindly removed, as certain clitics (words that don't stand alone) depend on them for meaning.

For languages like Chinese, tokenization is even more complex since it does not use spaces to separate words. Deciding what constitutes a word is non-trivial, and a common approach is to treat each character as an individual token. Other languages, such as Thai and Japanese, require sophisticated segmentation techniques beyond simple whitespace or character-based tokenization.

A more advanced method, sub-word tokenization, can be useful for handling long words and capturing morphological patterns within a language. Instead of relying purely on spaces, data-driven techniques determine the optimal way to segment text. This is particularly important for Machine Learning applications, where models benefit from explicit knowledge of a language's structure. A common approach is byte-pair encoding.

In some tasks, text must be split into sentences rather than just words. Sentence segmentation often relies on punctuation marks, which typically indicate sentence boundaries. However, periods are more ambiguous, as they also appear in abbreviations, numbers, and initials. A common approach is to tokenize first and then use rule-based or Machine Learning models to classify periods as either part of a word or a sentence boundary.

1.2.4 Text normalization

In many applications, such as web search, all letters are converted to lowercase. This process significantly reduces the vocabulary size and improves recall by ensuring that variations in capitalization do not affect search results. Since users often type queries in lowercase, this normalization helps retrieve more relevant documents.

For classification tasks, removing case can simplify the learning process by reducing the number of distinct tokens. With fewer parameters to learn, models can generalize better even with limited training data.

However, case folding is not always beneficial. In some contexts, capitalization carries meaningful information. Machine translation and information extraction may also benefit from preserving case distinctions.

Beyond case folding, word normalization involves converting words or tokens into a standard format, ensuring consistency in text processing. This step is particularly crucial in applications like web search, where variations in word forms should not hinder retrieval performance.

Stopwords Stopwords are the most frequently occurring words in a language. They typically have extremely high document frequency scores but carry little discriminative power, meaning they do not contribute much to understanding the main topic of a text. Removing stopwords can sometimes improve the performance of retrieval and classification models, mainly by reducing computational and memory overhead. Eliminating common words can also speed up indexing by preventing the creation of excessively long posting lists. However, stopword removal is not always beneficial. In some cases, stopwords play an important role in understanding meaning and context.

1.2.5 Morphology and lemmatization

Morphology, a fundamental concept in linguistics, refers to the analysis of word structure. At its core, it involves breaking words down into their smallest meaningful units, known as morphemes.

Definition (*Morpheme*)**.** A morpheme is the smallest linguistic unit that carries meaning.

A morpheme can be a root (base form) or an affix, which can appear as a prefix, infix, or suffix.

Definition (*Lexeme*)**.** A lexeme is unit of lexical meaning that exists regardless of inflectional endings or variations.

Definition (*Lemma*)**.** A lemma is the canonical form of a lexeme.

Definition (*Lexicon*)**.** A lexicon is the set of all lexemes in a language.

Definition (*Word*)**.** A word is an inflected form of a lexeme.

Lemmatization Lemmatization is the process of reducing words to their lemma, or base form. By normalizing words to a common root, it helps deal with complex morphology, which is essential for many languages with rich inflectional systems.

Stemming Stemming is a simpler approach that removes affixes based on predefined rules, often without considering the actual meaning or structure of the word. Unlike lemmatization, stemming does not require a lexicon.

Porter stemming algorithm (1980) is one of the most widely used stemming algorithms, it applies a set of rewriting rules to reduce words to their stems. While computationally efficient, stemming can introduce errors such as collisions (different words may be reduced to the same stem) and over-stemming (some words may be shortened excessively, losing meaning).

While stemming is computationally cheaper, lemmatization provides more linguistically accurate results, making it preferable for tasks requiring precise language understanding.

1.3 Regular expressions

Text documents are fundamentally just sequences of characters. Regular expressions provide a powerful way to search within these sequences by defining patterns that match specific character sequences. Regular expressions are useful for:

- *Pattern detection*: determine whether a specific pattern exists within a document.
- *Information extraction*: locate and extract relevant information from a document whenever the pattern appears.

Name	**Formula**	**Description**
Exact match	`aaa`	Matches the exact sequence `aaa`
Sequence choice	`(aaa|bbb)`	Matches either `aaa` or `bbb`
Wildcard	`.`	Matches any single character except for a newline
Character choice	`[]`	Matches any one character inside the square brackets
Newline	`\n`	Represents a newline character
Tab	`\t`	Represents a tab character
Whitespace	`\s`	Matches any whitespace character
Non-whitespace	`\S`	Matches any non-whitespace character
Digit	`\d`	Matches any digit (`[0-9]`).
Word character	`\w`	Matches any word character (`[a-zA-Z0-9]`)
Zero or more times	`*`	Matches the preceding character zero or more times
One or more times	`+`	Matches the preceding character one or more times
Zero or one time	`?`	Matches the preceding character zero or one time
Exactly n times	`{n}`	Matches n occurrences of the preceding character
From n to m times	`{n,m}`	Matches between n and m occurrences of the preceding character

1.3.1 Regular expressions in text mining

Regular expressions offer a powerful way to define patterns that can extract specific content from text documents. This allows for highly customizable and efficient text processing.

The advantages of regular expression based text extraction are:

- *Simplicity*: regular expressions are a straightforward way to specify patterns.
- *Precision*: extraction rules can be finely tuned to target specific patterns, which reduces false positives.

The limitations of regular expression based text extraction are:

- *Manual rule creation*: writing extraction rules usually requires manual effort, which can be time-consuming and complex.
- *False positives*: regular expressions can still yield false positives, where the pattern matches unintended content.
- *False negatives*: false negatives occur when the rules are not broad enough to capture all valid cases.
- *Lack of context awareness*: regular expressions typically work on isolated patterns, without understanding the context in which the pattern appears.