Natural Language Processing
Course Summary - Master's Degree

Carlos Alberto Botina Carpio
Universidad Internacional de la Rioja
carlos.botina621@comunidadunir.net

November 16, 2025

Abstract

This document contains a summary of the Natural Language Processing course syllabus for the Master's Degree. It includes a summary of the main topics covered during the sessions, as well as additional explanations and extensions of the concepts and techniques referenced in class. The purpose of this document is to serve as study material and reference for the course contents.

Notice that this document is fully customized to the author's needs, placing greater emphasis on topics that the author struggles with or has not yet mastered, while covering more briefly those topics that are already well understood by the author.

Contents

Lecture 002

1 Characteristics of Text as a Data Source

In the Big Data era, large volumes of heterogeneous data are generated (images, texts, IoT device data, etc.). To work effectively with a specific data source, it is essential to understand its characteristics. This applies to texts in NLP.

1.1 Compositionality Principle

Language is **compositional** and can be treated discretely. The overall meaning of a text (e.g., a sentence) can be composed from the meaning of its elements (e.g., words).

Definition 1.1 (Compositionality Principle)**.** The representation of a text's meaning can be obtained from the meaning of its constituent elements.

Example:

- "He aprobado el examen" (I passed the exam) combines the subject "He aprobado" with the predicate "el examen"
- "He aprobado el examen tras mucho estudio" extends the previous meaning by adding "tras mucho estudio"

Compositionality also applies at the word level (morphology). For example, "médicos" (doctors) combines the root "médico" with "s" to indicate plural.

Limitations: Compositionality alone is insufficient when dealing with:

- **Lexical ambiguity**: Words with multiple meanings (e.g., "banco" can mean bank, bench, or school of fish)
- **Idiomatic expressions**: Phrases whose meaning cannot be derived from individual words

1.2 Distributional Perspective

The distributional perspective addresses compositionality limitations by constructing meaning from context.

Definition 1.2 (Distributional Perspective)**.** The meaning of text elements (e.g., words) can be obtained in an unsupervised manner based on the context that typically surrounds them.

Example: The meaning of "banco" is disambiguated by surrounding words:

- "... fue al banco a contratar una hipoteca" $\rightarrow$ financial institution
- "... en el mar pude ver un banco de peces" $\rightarrow$ school of fish
- "... ellos se sentaron en un banco de madera" $\rightarrow$ bench

This approach is **unsupervised**—no manual annotation is required, only other texts to infer meaning from context.

1.3 Sequential Aspect

Texts have a **sequential nature** that must be considered. Compositionality and distributional perspective alone are insufficient when word order matters.

Definition 1.3 (Sequential Aspect)**.** The sequentiality (bidirectional) of text must often be considered when constructing meaning.

Examples:

- "Quiero información de hipotecas, pero no seguros" vs. "Quiero información de seguros, pero no hipotecas"
  - Same words, different meanings due to word order
- "Al campo no he ido, he ido a la playa" vs. "A la playa no he ido, he ido al campo"
  - Information appears *after* the negated element (bidirectional)

1.4 Syntactic Ambiguity

Some cases require considering **syntactic relationships** between words, not just compositionality, distribution, and sequence.

Example: "Pedro ve un cuadro de su madre" (Pedro sees a painting of his mother)

- Ambiguity: Does the painting belong to the mother, or does it depict the mother?
- Requires syntactic analysis to resolve

Definition 1.4 (Syntactic Ambiguity)**.** A situation where the same sentence can have different interpretations based on syntactic structure.

2 Morphology

According to the Royal Spanish Academy (RAE), **morphology** is "the part of grammar that studies the structure of words and their constituent elements".

Morphology studies how words decompose into indivisible parts, each with meaning. These minimal units are called **morphemes**.

Definition 2.1 (Morpheme)**.** The minimal unit that composes a word and is capable of expressing meaning.

Example: The word "mujeres" (women) contains two morphemes:

- "mujer" (woman) — the root
- "es" — indicates plural

2.1 Grammatical Morphemes and Lexemes

A **grammatical morpheme** has grammatical meaning, such as:

- Gender (masculine or feminine)
- Number (singular or plural)
- Person (e.g., third person singular)
- Mood and tense (e.g., indicative mood, future tense)

In contrast, a **lexeme** is the morpheme that forms the root or invariant part of the word, providing the main lexical meaning.

A **lemma** is the canonical form of a word (e.g., infinitive for verbs, singular for nouns) used as the dictionary entry.

Definition 2.2 (Lemma)**.** The canonical or dictionary form of a word, used as the base form for all its inflected variants.

Example: In "mujeres":

- "mujer" is the lexeme (root with lexical meaning)
- "es" is the grammatical morpheme (expresses plural number)
- "mujer" is also the lemma (canonical form)

The singular form "mujer" contains:

- The lexeme "mujer"
- A **zero morpheme** for singular number

Definition 2.3 (Zero Morpheme)**.** A grammatical morpheme without phonetic realization but with grammatical meaning.

2.2 Example: Complex Word Structure

The word "cantábamos" (we were singing) contains three morphemes:

1. **Lexeme "cant"**: Lexical meaning (the act of producing melodious sounds with the voice)
2. **Grammatical morpheme "aba"**: Indicative mood, past tense
3. **Grammatical morpheme "mos"**: First person plural

The lemma for "cantábamos" is "cantar" (to sing).

2.3 Morphological Analysis Examples

Table 1 shows examples of morphological analysis, decomposing words into their lexemes, grammatical morphemes, and lemmas.

Word	Lexeme	Gram. Morpheme (1)	Gram. Morpheme (2)	Lemma
Cantábais	Cant	ába	Is	Cantar
Casita	Cas	Ita	Zero (SG)	Casa
Perros	Perr	O	S	Perro
Vino	Vino	Zero (M)	Zero (SG)	Vino

Table 1: Examples of morphological analysis in Spanish

2.4 Computational Morphology

Computational morphology aims to automatically recognize the morphemes contained in a word.

Importance in NLP:

- **Search and retrieval**: When searching for "pensar" (to think), also recognize "piénsalo" (think about it), even though they don't share the same character sequence in the root
- **Text generation**: Ensure grammatical agreement (e.g., gender and number between nouns and adjectives, person between verbs and subjects)

2.5 English vs. Spanish Morphological Analysis

Morphological analysis in English (where most computational morphology research is conducted) is simpler than in Spanish because:

- Spanish has many more variations in word endings
- Spanish has more root alterations (stem changes)

3 Text Normalization Techniques

The compositionality principle, while not always sufficient for constructing meaning, is adequate for many text types and use cases. It enables transforming text into input variables for training machine learning models.

For example, in sentiment analysis (classifying text as positive, negative, or neutral), a supervised model requires:

- Annotated texts with sentiment categories
- Input variables extracted from text following the compositionality principle (e.g., which words appear)

Definition 3.1 (Text Normalization Pipeline)**.** The transformation of text into its constituent elements for subsequent processing, following a normalization pipeline with distinct steps.

3.1 Step 1: Tokenization

Tokenization consists of decomposing a text string (e.g., a sentence) into its terms or components (e.g., the words that form it).

Example: For the sentence:

> "Después de estar estudiando 2 horas, he decidido estudiar 2 horas más."

The tokenization result is:

> "Después", "de", "estar", "estudiando", "2", "horas", "he", "decidido", "estudiar", "2", "horas", "más".

This process typically:

1. Removes punctuation marks (commas, periods)
2. Separates text into words based on whitespace
3. Produces an ordered list of words

Definition 3.2 (Tokenization)**.** The process of separating text into smaller parts (tokens) such as words or phrase components.

3.1.1 Observations from Tokenization

After tokenization, several aspects emerge:

- **Repeated words**: Some words appear multiple times (e.g., "horas")
- **Capitalization**: First word may be capitalized (e.g., "Después")
- **Numbers**: Non-word tokens appear (e.g., "2")
- **Word variations**: Similar words with different forms (e.g., "estudiando" vs. "estudiar")

These aspects are addressed in subsequent normalization steps.

3.1.2 Limitations and Challenges

Punctuation handling: Simple punctuation removal is insufficient for:

- Abbreviations: "Dr.", "o'clock"
- Possessives: "doctor's" (apostrophe 's)
- Dates: "2022-06-01"
- Times: "08:30"

Solution: Differentiate between punctuation types using:

- Rules or heuristics
- Databases for standardizing terms

Proper nouns and compound words: Multi-word expressions should remain as single tokens:

- Proper nouns: "Nueva York" (not "Nueva" + "York")
- Compound expressions: "Estado del Arte", "a priori"

Solution: Two-level tokenization:

1. **Low-level**: Initial tokenization (word-level)
2. **High-level**: Semantic tokenization (phrase-level)

High-level tokenization can use:

- Statistical approaches considering word co-occurrence
- Named entity recognition
- Phrase dictionaries

3.1.3 Token Importance

From the compositionality perspective, not all tokens are equally important for a given task.

Example: For sentiment analysis of:

> "Estoy contento de haber comprado este libro."

Words like "de" and "este" provide little information for sentiment classification. This leads to the next normalization step.

3.2 Step 2: Stopword Removal

Stopword removal eliminates tokens that are very common and provide little information, such as articles, conjunctions, prepositions, and sometimes pronouns.

Definition 3.3 (Stopword Removal)**.** A technique used to remove tokens that are very common and provide little information, such as articles, conjunctions, or prepositions when those tokens represent words.

Types of stopwords:

- Articles (e.g., "el", "la", "un", "una")
- Conjunctions (e.g., "y", "o", "pero")
- Prepositions (e.g., "de", "en", "con")
- Pronouns (in some cases)

Detection method: Language-specific dictionaries containing lists of words to remove.

Important considerations:

- Stopword removal is **not always applied**—it depends on the specific use case
- Some NLP tasks may require the information provided by these words
- The decision to remove stopwords should be based on the task requirements

Example: After tokenization and stopword removal, the sentence:

"Estoy contento de haber comprado este libro."

Might become: "contento", "haber", "comprado", "libro" (removing "de" and "este").

3.3 Step 3: Capitalization Treatment

Capitalization treatment handles uppercase letters in tokens appropriately, removing them when both uppercase and lowercase versions refer to the same word.

Definition 3.4 (Capitalization Treatment)**.** The process of appropriately handling capitalization in tokens, removing uppercase letters when both a capitalized token and a lowercase token refer to the same word.

Problem: Without capitalization treatment, the same word with different capitalization is treated as different tokens.
Example: In the sentence:

"Días y días pasaron sin noticias nuevas."

"Días" and "días" would be treated as two distinct tokens, even though they represent the same word.
Solution: Normalize by removing capitalization, so both words are represented by the same token.
Challenge: Capitalization removal is not always appropriate. Consider:

"Su amiga Estrella estaba de vacaciones, y vio una estrella en el cielo."

- "Estrella" (capitalized) refers to a person's name (proper noun)
- "estrella" (lowercase) refers to a common noun (star)

Removing capitalization from "Estrella" would incorrectly represent both as the same token.
Approaches:

1. **Rule-based**: Remove capitalization only from the first word of sentences
2. **Sophisticated solutions**: Use Named Entity Recognition (NER) algorithms to identify tokens referring to persons, organizations, etc., preserving capitalization in those cases

Additional considerations: Other normalization treatments can homogenize token representations. For example, in the first text, "dos" was represented as "2". If both representations coexist in a document ("dos" and "2"), they should be normalized to the same representation.

3.4 Step 4: Special Character Treatment

In some text types (e.g., tweets), non-alphanumeric characters appear (hashtags, @, etc.). The normalization pipeline must handle these characters when representing text as normalized tokens.

Approach: In some cases, these characters can be removed (e.g., removing some exclamation marks), but this depends on:

- The specific use case
- Whether removal causes significant information loss

This step is typically called **special character treatment**.

3.5 Order of Normalization Steps

The order of normalization steps is **not always fixed**. While tokenization is typically the first step, the order of other steps can vary depending on the specific case.

Example: For "Principado de Asturias", capitalization and special character treatment might occur before stopword removal, allowing detection as a named entity (place) and representation as "Principado" "de" "Asturias" before removing the stopword "de".

3.6 Lemmatization and Stemming

After the normalization steps, some use cases include obtaining the **lemma** or **root** of words.

Example: In the text:

> "Ayer jugaron al mismo deporte que han jugado hoy."

Two verbs appear: "jugaron" and "jugado" in different conjugations. Obtaining the lemma or root represents both with the same token.

3.6.1 Lemmatization

Lemmatization represents words by their lemma (canonical form). For "jugaron" and "jugado", the lemma would be "jugar" (to play).

Methods:

- Dictionaries with equivalences between words and lemmas
- Other computational techniques

3.6.2 Stemming

Stemming is different from lemmatization. It obtains the root (lexeme) by removing grammatical morphemes. For "jugaron" and "jugado", the stem would be "jug".

Algorithms: Various algorithms exist, such as the Snowball algorithm (Porter, 2001).

Definition 3.5 (Lemmatization and Stemming)**.** Processes that seek to simplify word representations through tokens using their lemmas or lexemes respectively.

3.6.3 Lemmatization vs. Stemming

Problem with stemming: Sometimes two words that should have the same representation do not.

Example:

- "jugaron" and "yo juego" (I play)
  - **Lemmatization**: Both become "jugar"
  - **Stemming**: "jugaron" → "jug", "juego" → "jueg" (different stems)

Important consideration: Lemmatization or stemming is **not always advisable** for all use cases, as it can cause loss of relevant information.

Example: In the text:

> "El número de ingenieras egresadas ha aumentado en los últimos años."

If lemmatization is applied, "ingenieras" and "egresadas" would become "ingeniero" and "egresado", losing important information about gender expressed in the text.

3.7 Final Considerations

These normalization pipelines are useful for preprocessing many texts before using them in various NLP tasks, especially when the source is raw text.

Alternative formats: Other data formats (e.g., XML) include relevant information alongside texts, making some normalization phases unnecessary in many cases.

4 Linguistic Resources

The implementation of different Natural Language Processing tasks requires various linguistic resources. Among the linguistic knowledge bases, the following stand out:

- Dictionaries
- Lexicons
- Thesauri
- Lexical relation databases

Additionally, collections of texts or speech resources called **corpora** (singular: **corpus**) are also important linguistic knowledge bases.

4.1 Dictionary

A **dictionary** is a repertoire where words of a language are grouped according to a specific order, accompanied by their definition or explanation. Dictionaries include a detailed and human-understandable description of the different senses of words.

Example: A dictionary entry for "banco" might look like:

banco *(noun)*
1. A financial institution that accepts deposits and makes loans.
2. A long seat for several people, typically made of wood or stone.
3. A group of fish swimming together.

4.2 Lexicon

A **lexicon** is a dictionary that contains morphological information. This morphological dictionary is a repertoire that collects a list of morphemes and basic information about them.

Example: A lexicon entry might include:

cantábamos
Lexeme: *cant*
Morphemes: *cant* + *aba* + *mos*
Morphological information:

- Root: *cant-* (to sing)
- Tense/Aspect: *-aba-* (imperfect past)
- Person/Number: *-mos* (first person plural)

4.3 Thesaurus

Also called **thesaurus**, **thesauri**, or **tesoro**. It refers to a dictionary that contains a list of word meanings and the relationships between these meanings.

In literature: A thesaurus contains a list of words with their synonyms and antonyms.

In linguistics: A thesaurus gathers knowledge about hypernymy/hyponymy and meronymy/holonymy relationships represented in the thesaurus's hierarchical structure. That is, the thesaurus hierarchy models:

- The **is-a** relationship (hypernymy/hyponymy)
- The **part-whole** relationship (meronymy/holonymy)

Additionally, a thesaurus can also group knowledge about other semantic relationships such as synonymy or antonymy.

Example (Literature): A thesaurus entry for "happy":

happy
Synonyms: joyful, cheerful, content, delighted, pleased
Antonyms: sad, unhappy, miserable, depressed

Example (Linguistics): A hierarchical thesaurus structure:

vehicle (hypernym)
→ **car** (hyponym)
→ **sedan** (hyponym)
→ **SUV** (hyponym)
→ **bicycle** (hyponym)

car (holonym)
→ **wheel** (meronym)
→ **engine** (meronym)
→ **door** (meronym)

Curious Fact: Semantic Relationships

Hypernym and Hyponym (is-a relationship):

- A **hypernym** is a more general term that encompasses more specific terms. Example: "vehicle" is a hypernym of "car".
- A **hyponym** is a more specific term that belongs to a broader category. Example: "car" and "bicycle" are hyponyms of "vehicle".
- Relationship: "A car *is a* vehicle" (hyponym is-a hypernym).

Holonym and Meronym (part-whole relationship):

- A **holonym** is a whole that contains parts. Example: "car" is a holonym of "wheel".
- A **meronym** is a part that belongs to a whole. Example: "wheel", "engine", and "door" are meronyms of "car".
- Relationship: "A wheel *is part of* a car" (meronym part-of holonym).

Mnemonic: Think of hypernym as "super" (above) and hyponym as "sub" (below) in a hierarchy. For holonym/meronym, remember that a "whole" (holonym) contains "parts" (meronyms).

4.4 Lexical Relation Databases

A **lexical relation database** is a generalization of thesauri—a type of dictionary that collects knowledge about any semantic relationship between word senses. These databases contain:

- A set of lemmas
- Each lemma annotated with:
  - The possible set of senses of the word
  - The relationships between those senses

Example: A lexical relation database entry (similar to WordNet structure):

Lemma: *car*
Synset 1: {car, auto, automobile, machine, motorcar}
Sense: "a motor vehicle with four wheels"
Hypernym: vehicle
Hyponyms: sedan, SUV, coupe, convertible
Meronyms: wheel, engine, door, window
Part-of: traffic, fleet

Synset 2: {car, railcar, railway car, railroad car}
Sense: "a wheeled vehicle adapted to the rails of railroad"
Hypernym: wheeled vehicle
Hyponyms: passenger car, freight car, dining car

4.5 Linguistic Corpus

A **linguistic corpus** is a collection of texts representative of a language used for linguistic analysis.

4.5.1 Types of Corpora

Textual corpora: Collections of extracts from books, magazines, newspapers, or any other written source.

Oral corpora: Collections of speech, i.e., audio extracts from sources such as radio or television.

4.5.2 Tagged Corpora

Corpora can be **annotated** or **tagged** so that the words they contain present additional linguistic information, such as:

- Syntactic information
- Semantic information
- Pragmatic information

These are known as **tagged corpora**.

4.5.3 Historical Development

The first corpora available online appeared in the 1960s. One of the pioneers was the **Brown Corpus**, an American English corpus developed in 1963 by Brown University, containing a collection of one million words extracted from 500 texts of different genres: newspapers, novels, non-fiction, academic, etc. (Kucera and Francis, 1967). Initially, the corpus was not tagged, but over the years it was tagged with information about grammatical categories (POS).

Modern corpora: Today's corpora tend to be much more extensive than the Brown Corpus. For example, the **WSJ corpus** (Wall Street Journal), tagged with morphosyntactic information, is a collection of one million words published in Wall Street Journal articles in 1989.

5 NLP Tools and Libraries

Numerous tools and libraries facilitate the development and implementation of different Natural Language Processing tasks.

5.1 Python Libraries

5.1.1 Natural Language Toolkit (NLTK)

NLTK is one of the most widely used tools for implementing NLP applications in Python. It is:

- Available for Windows, Mac OS X, and Linux

- Open source (Apache License Version 2.0)
- Primarily developed for English NLP, but flexible enough for other languages like Spanish

Features:

- Interfaces to more than fifty corpora and lexical resources (e.g., WordNet)
- Libraries for classification, tokenization, stemming, morphosyntactic tagging, syntactic parsing, and semantic analysis
- Can train morphosyntactic taggers in Spanish from tagged corpora
- Allows integration of external taggers for Spanish

5.1.2 spaCy

spaCy is a Python library widely used in both academic and industrial settings.

Features:

- Incorporates various text normalization pipelines (tokenization, lemmatization, etc.)
- Supports more than sixty languages
- Includes recent neural models for NLP tasks (e.g., Named Entity Recognition)
- Includes pre-trained models like BERT
- Open source for commercial use (MIT License)

5.1.3 Gensim (Generate Similar)

Gensim is a Python library for working with NLP.

Features:

- Topic modeling
- Document indexing
- Document search using techniques like vector embeddings

5.2 Java Libraries

5.2.1 LingPipe

LingPipe is a Java library for developing NLP applications.

5.2.2 Apache OpenNLP

Apache OpenNLP is a Java library for NLP tasks, providing tools for various text processing operations.

5.2.3 Stanford NLP

Stanford NLP is a suite of Java libraries for NLP, offering various tools for text analysis and processing.

5.2.4 Stanford CoreNLP

Stanford CoreNLP is a comprehensive Java library for NLP tasks.
Features:

- Incorporates models for NLP tasks in Spanish
- Handles Spanish-specific characteristics:
  - Separation of word contractions (e.g., "del" = "de" + "el") during tokenization
  - Identification of verb tenses and moods (e.g., present indicative) in morphosyntactic tagging
- Implemented in Java but provides command-line access, enabling integration with Python
- Can be used with NLTK in Python code for Spanish morphosyntactic tagging

5.3 Node.js Libraries

5.3.1 Natural

Natural is a library for implementing NLP tasks in Node.js, providing various text processing capabilities for JavaScript applications.