Natural Language Processing (NLP)

Do Research field – enormously.
Gold Standard Algorithms -> English.
Natural Language Processing algorithms are primarily tested in English.

English syntax is very simple -> linguists maintain that a simple syntax indicates an evolved language.
French is an example of an unevolved language.
72% -> indicates a very good result (success).
We are comparing a syntactic analysis to a human subject.
Morphologically, syntactically processing.

NLP refers to a section of.

Trying to express natural language structures formally.
We capture the difference between linguistics and language processing. This is NOT linguistics. We view language from the point of view of data scientists.
Software has to not only understand ~~long~~ natural language, but also as a reverse process to generate natural language.

Natural Language Generation

List of topics for the Exam.
Lab – applied Python.

NLP – creating a technology that deals with natural language (entirely different from formal languages techniques).

Computational linguistics tries to find models for natural language.
NLP takes place at the following levels: (Syntax is completely different).

NLP Processing Levels:
- morphological
- syntactical
- semantical
- pragmatic level.

Morphological → Input phrase, but not with the sense of a set of sentences! A phrase is any group of words!
We want to determine the part of speech of each word in a phrase.
Ultimate goal is for the software to understand a phrase. Natural language is characterized by ambiguity
(for. ex: bank, banking ops.)
noun verb

Ambiguity is very strong in English. Semantic ambiguity is particularly high in English.
As a noun → 7 meanings. } 12 meanings.
As a verb → 5 meanings }
At the lowest level, we perform morphological analysis → part of speech is v. important, it helps solve ambiguity.
[Tag each word in the input phrase with a part of speech. → software is termed POS-tagger → family of software, Part of Speech taggers, there are several available, the

famous being the Stanford Group one.
(Stanford Part of Speech tagger, famous and very successful)

At the lab, we will work with NLTK, Python, which has its own POS-tagger. It is not clear which one is the best. We do not have serious comparisons yet.

Java programmers always use the Stanford POS-tagger.

Stanford POS-tagger had 98% success, which is extremely high. This is however a field where 100% success cannot be attained. At that point, it was considered that morphological level was solved, and was considered a closed field of NLP.

- Syntactic level → means the structure of the phrase (there are completely different syntactic levels)

Phrase structure. Syntactic level is very much established. Discussion is focused on the type of syntactic analysis.
(key word!)

- Semantic level → is very much alive, even for English, nothing is established.

Semantics → means determining meaning, sense.

At semantic level, I am trying to determine the meaning of an input phrase, context free, regardless of the context it occurs in.

Various types of ambiguities that can occur.

Morphological ambiguity → Rice flies like sand

verb? noun?

Establishing meaning context free.

— At the pragmatic level, we do take content and context into account.

This is an underdeveloped level → many PhD thesis in this field.

We will not get into the pragmatic level at all.

We will focus on the semantic level.

There are 2 completely different theories concerning syntax.

— Syntactic level focuses on the structure of a phrase. Determining the structure of a phrase → the process is called parsing.

⊛ Syntactic analysis algorithms are called parsing algorithms.

Parsing is very well established for English.

Terms for syntactic analysis:

— syntactic phrase (group syntactic)

↳ a group of words which are organized around a word, which is considered the group head. Those words are in cohesion around that group head.

This cohesion is insured by grammar rules.

Various types of syntactic groups:

- noun phrase (NP) - the head is a noun.
- verb phrase (VP) - the head is a verb.
- adjectival phrase (AdjP)
- adverbial phrase (AdvP)
- prepositional phrase (PP) - the head is a preposition.

Determining the structure of a phrase (syntactical analysis) can mean obtaining the concatenated phrases that represent the result of parsing.

If I view the structure of a phrase as a concatenation, I will use in the background a generative grammar, which gives rules.

Rules: ascendence $S \rightarrow NP\ VP$

$VP \rightarrow V\ PP$

A verb phrase → verb ↑ prep. phrase.

$PP \rightarrow PP\ NP$

These are recursive structure rules, typical for generative grammars (Chomsky).

If we change the type of grammar then the syntactic analysis changes → the output changes.

Bibliografie → pentru analiză sintactică.

1. F. Hristea – Introducere în procesarea limbajului natural, Editura Universității din București, 2010.
   ↳ partea de gramatici generative este identică în cele 2 cărți recomandate. (capitolul 3, intitulat "Gramatici")

2. Aspecte ale reprezentării cunoștințelor, cu aplicații în procesarea limbajului natural.
   Ed. Univ. Buc., 2011, capitolul 3 – "Gramatici generative pt. procesarea limbajului natural".

Aceste cărți sunt în parte introductive.

- Analiza sintactică bazată pe constituenți: Constituentul este un grup sintactic.
- Analiza sintactică de dependență este complet diferită.

Tipurile de analiză sintactică → depind de tipul de aplicație.

Ultimate goal – make it easier for the software to understand natural language.

Interdisciplinarity.

This type of syntactic analysis uses generative grammars, which are tools of knowledge representation. A grammar

Generative grammars → set of phrase-structure rules, which are recursive rules based on concatenation (created by Chomsky)

VP → V NP This is a phrase-structure rule.

For ex, a verb phrase can be created with just one verb VP → V.

NP → N P Adj
NP → N Adj } → phrase structure rules based on concatenation, and they are recursive.

They can be represented as a tree.

There is a symbol designating the entire input phrase.

↳ [S → NP VP] main structure rule.

These rules can refer both to syntactic rules and individual words (eg adjectives, lexical symbols).

NP → N Adj.

N ↑ Noun Adj ↑ Adjective

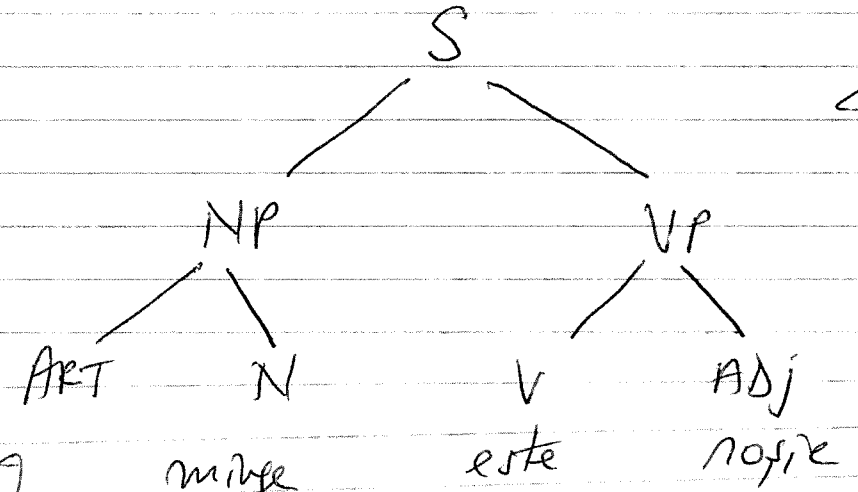
Individual words → content words (can't be in content)

↳ { nouns
adjectives
verbs
adverbs

Articles, conjunctions, prepositions are words without content. Typically, we will instruct the software to eliminate words without content.

The operation is called parsing, the output of the parses is the syntactic structure of the phrase, represented by its constituents.

Assuming we have the following inputs:
"O mingе este noapte".



← Structure tree (which is the output of the parser).

S [np [(art, o), (n, mingе)], vp [...]]

↳ list structure that represents the tree.

This represents a syntactical analysis of this phrase.

The main difficulty in natural language is ambiguity.

Ambiguity can be of different types.

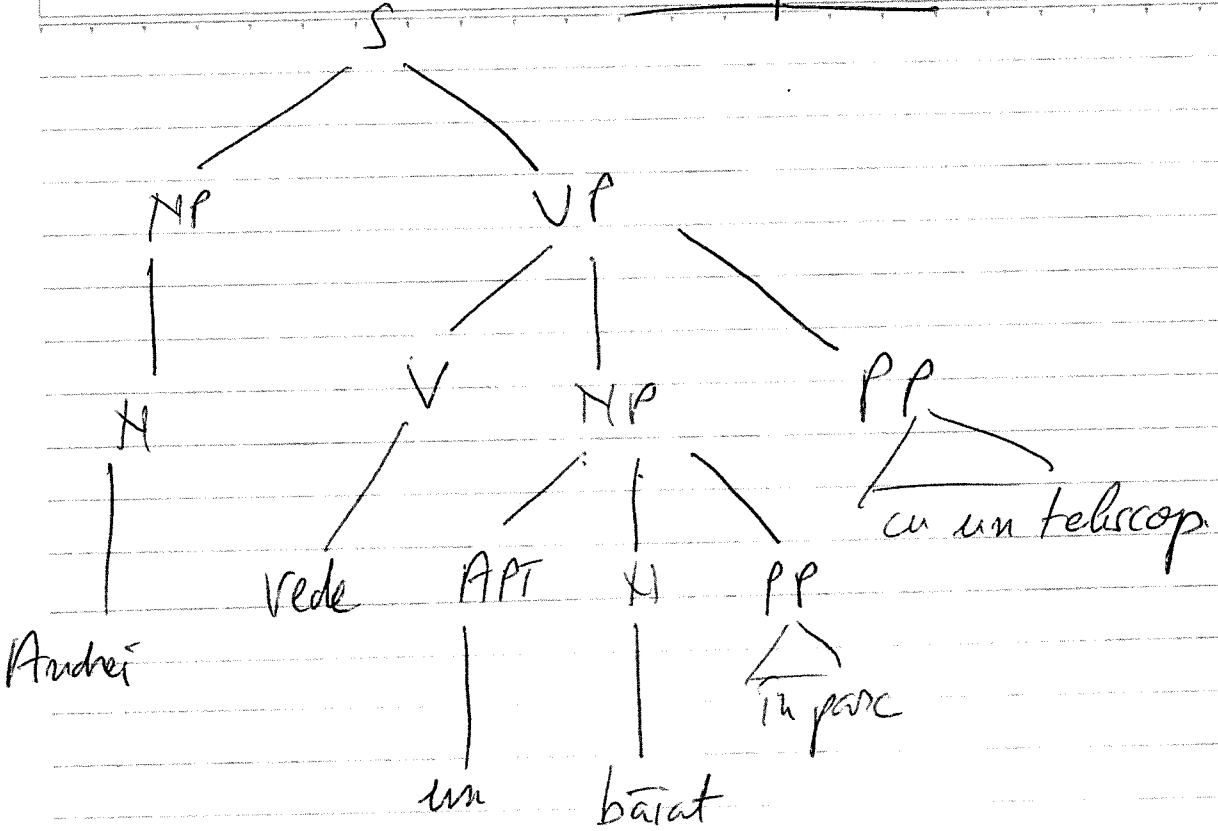
1). Structural ambiguity → this occurs whenever a parser detects different possible structure for the phrase → for each structure there is a different tree. This takes you to the next level.

"Andrei vede un băiat în parc cu un telescop."

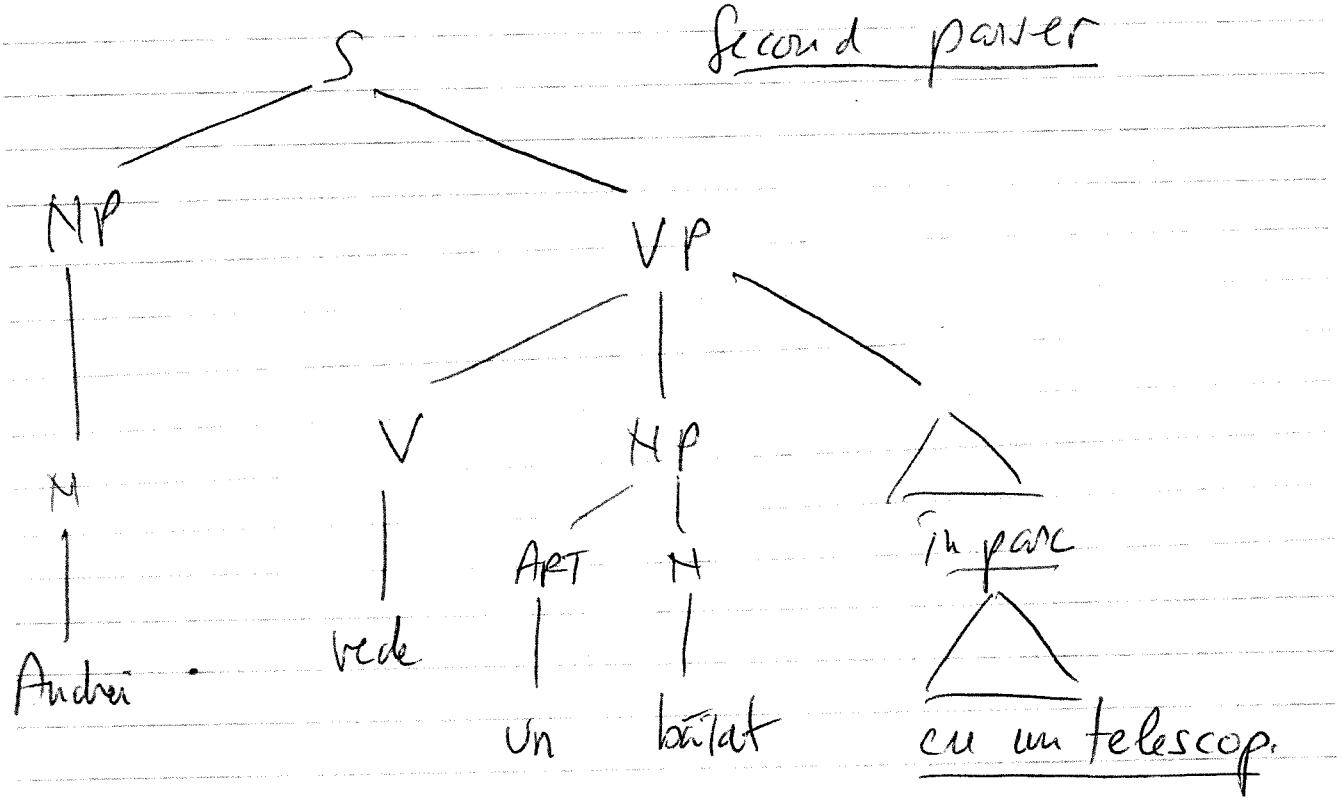
Structural ambiguity leads to semantic ambiguity.

An intelligent algorithm should give me two parsing trees.

First parser.



Second parser



2). Lexical ambiguity → a word can be different parts of speech. (ex: nice flies).

Lexical ambiguity can lead to structural ambiguity.

Sometimes, ambiguity can be resolved.

Ex. Flying planes are dangerous.
Flying planes is dangerous.
↳ the singular gives us the part of speech Verb.

3. Semantic ambiguity, which concerns us the most.

A word has multiple meanings. A word has several meanings.

Ex of a sentence which is not structurally ambiguous, but it is semantically ambiguous:
"Fiecare copil iubește o prăjitură"

Natural Language Processing Laboratory

```
dir(text2)
text2.vocab().keys()
l.sort()
print(l)
[exp(x) for x in list if cond(x)]
```

string.isalnum() → The isalnum() method returns True if all characters in the string are alphanumeric. If not, returns False.
"abc".isalnum()