

Parsing and syntactic analysis PART I

*Linguistic Resources for Natural Language Processing
LM Language Technologies and Digital Humanities
2024-25*

Cristina Bosco

Overview

- What is syntax?
- Some theoretical frameworks for syntax across tradition and language variation
- Constituency-based formats

Some definition

Syntax (from ancient Greek σύνταξις = 'association', 'organisation in some order') is the study of the order and arrangement of the words into larger units.

Syntax studies the type and structure of sentences, clauses and phrases.

Some definition

Syntax provides us with the means to represent the structure behind the simple stringing together of words, their linear sequence.



To express language, as **written** or **spoken**, we are **bound to the use** of means of expression that enforce **sequentiality**, but we are aware of a more complex organization behind the sequence of words.

Syntactic Parsing (more often called simply **Parsing**) is the task which provides as a result the syntactic analysis of a text.

In computer science the term *parsing* simply indicates the **scanning** of some type of data.

In NLP, the term parsing refers to *syntactic parsing*, a task which consists in **scanning the sentence structure**, that is performing its syntactic analysis.

In NLP, the **analysis** of a text is **organised in a pipeline of several steps**.

Parsing is usually the third step, after tokenisation and PoS tagging (and lemmatisation).

In a **PIPELINE**:

- Each step is applied on a **suitable datum** or set of data: document, word, sentence ...
- Each step provides **novel information** to add to the linguistic data: about the subdivision of linguistic components, morphology, syntax ...
- Each step **benefits from** the analysis made available in the **previous step(s)**: about the subdivision of linguistic components, morphology, syntax ...

Usually the NLP pipelines include parsing as the third step, after segmentation, tokenisation and PoS tagging.

Corpus > Doc₁ ... Doc_n

Segmentation

Sentence > Sent₁ ... Sent_m

Token > Tok₁ ... Tok_k

Token+tag > Tok₁+tag ... Tok_k+tag

Sentence+tags

Usually the NLP pipelines include parsing as the third step, after segmentation, tokenisation and PoS tagging.

Corpus > Doc₁ ... Doc_n

Segmentation

Sentence > Sent₁ ... Sent_m

Tokenization

Token > Tok₁ ... Tok_k

Token+tag > Tok₁+tag ... Tok_k+tag

Sentence+tags

Usually the NLP pipelines include parsing as the third step, after segmentation, tokenisation and PoS tagging.

Corpus > Doc₁ ... Doc_n

Segmentation

Sentence > Sent₁ ... Sent_m

Tokenization

Token > Tok₁ ... Tok_k

PoS Tagging

Token+tag > Tok₁+tag ... Tok_k+tag

Sentence+tags

Usually the NLP pipelines include parsing as the third step, after segmentation, tokenisation and PoS tagging.

Corpus > Doc₁ ... Doc_n

Segmentation

Sentence > Sent₁ ... Sent_m

Tokenization

Token > Tok₁ ... Tok_k

PoS Tagging

Token+tag > Tok₁+tag ... Tok_k+tag

Sentence+tags

Parsing

What's the parsing for?

Applying parsing techniques to a text means making explicit all the syntactic knowledge it implicitly contains.

Parsed texts, i.e. corpora annotated for morphology and syntax are called **treebanks**.

They are the linguistic resources used for representing the syntactic level of knowledge and for training and testing parsers.

Syntactic knowledge can be very useful for subsequent automatic and manual analysis. In particular, this knowledge seems to be an important prerequisite of semantic analysis, the one that allows to identify the meaning of the text.

Why is it necessary to know the syntactic structure of a sentence in order to understand its meaning?

Parsing is fundamental to understanding the content of the text because the **meaning is constructed in a compositional way**.

The meaning of a whole sentence can be deduced from the composition of the meanings of the individual words it contains. However, the composition of these meanings must take into account the linear and non-linear order (i.e. the structure) in which the words appear in the sentence.

The exception are the multi-word-expressions, which we will explore separately.

Parsing

When we talk about parsing, we always refer to the **sentence as a whole**.

All analyses of the individual word were carried out before parsing, during tokenization, lemmatization, PoS tagging (and NER).

During parsing, **the focus is on the connections that hold the words together** within the sentence.

The sentence is the main unit of analysis considered during parsing, and all other objects in the sentence are subordinate to it.

An analysis or more analyses?

In all phases of language analysis several alternatives are possible for what concerns the conceptual model and the subsequently used format:

- in tokenization we can choose whether to divide or keep together some components of a word (generating tokens)
- in PoS tagging we can use different tag sets and specify in a more or less detailed way morphological information related to a word

An analysis or more analyses?

In parsing the analysis becomes more complex and the number of alternative formats we can apply for representing syntactic knowledge increases enormously. There are **many theoretical approaches to syntax** and therefore many ways of describing a language from a syntactic point of view.

The choice of a particular type of format, e.g. for a treebank, becomes even more important because it has consequences on:

- which knowledge the parsing phase allows to extract from the text
- which algorithm we have to build and how difficult, long and complex syntactic analysis can become

An analysis or more analyses?

The choice of how to analyze the sentence from the syntactic point of view is strongly conditioned by:

- the language in question
- the purpose for which we are analyzing the sentence
- considerations about the complexity that the computer has to face and the time that it could take to do it.

Parser input

As in the case of tokenization and PoS tagging, in parsing also we need to know very well the input and the expected output.

In the parsing input, thanks to the previous analysis steps, all the grammatical information is already represented in explicit form.

In practice the sentence is organized into individual elements (tokens) and each of them has associated a* description of its morphology (lemma, tags and morphological features).

*Unfortunately sometimes more than one description, when the word is ambiguous.

Formalising syntax

Efforts to **formalise** human language for NLP have led to a **variety of theoretical frameworks** inspired by different languages and with different facets of syntax in mind.

Chomsky's theory has (at all times) attracted a variety of linguists, but probably never a majority, as there have always been competing theories:

Generative Semantics, Cognitive Grammar, Relational Grammar, Lexical Functional Grammar, Generalized Phrase Structure Grammar, Combinatory Categorical Grammar... just citing a few of them!

Formalising syntax

In early days, NLP focused almost exclusively on **English**. For this language formalisations and frameworks based on the notion of **phrase** (in its various declensions and in particular as described in Chomsky's theory) were particularly appropriate.

Today's NLP focuses instead on virtually **all existing languages**, in monolingual or multilingual settings, leading to the development of a variety of non-constituency based formats more appropriate for languages other than English. Among them a particularly relevant role has played the resurgence of syntactic formalisms based on the notion of **dependency**.

Formalising syntax

The development of formats for the morphological and syntactic annotation of corpora significantly benefited from the experience gained in the formalisation of languages.

Various **annotation schemes** have indeed been developed for treebanks, based on different **theoretical syntactic frameworks**.

An analysis or more analyses?

Two groups of formalisms (= formal descriptions) have proved particularly interesting and have been widely used in NLP:

- **constituency** > Constituent structure = phrase structure = syntagmatic structure = constituency grammar = formalism based on constituents
- **dependency** > dependency structure = dependency grammar = dependency based formalism

Constituency-based syntax

The best known and used model of formal grammar is the Context-Free Grammar (CFG).

First proposed by Chomsky in 1956, CFG is the classic way of describing the syntactic structure of the sentence through phrases, as a **phrase structure**, according to a **constituency-based** approach.

It underpins almost all parsing systems built until the 1990s.

Constituency-based syntax

Psycholinguistic experiments have shown that the notion of phrase is primitive

Within the sentence we perceive phrases spontaneously, having an innate knowledge a priori about them.

There are different kinds of phrases, each centred on a grammatical category: Noun Phrase, Preposition Phrase, Verb Phrase, Adjective Phrase, Adverb Phrase and Sentence.

Constituency-based syntax

A phrase (also called constituent) **is a group of words that behaves as if it were a single unit in the sentence.**

We can infer from certain behaviors that a certain group of words is a **constituent** applying 3 main **criteria**:

1. the group as a whole can be found in **similar contexts**
2. the group as a whole can be moved as if it were an **indivisible block**
3. the group as a whole can be **replaced** exclusively with a constituent of the same type.

Testing NPs according to the criteria

Example: a sentence with marked NPs

[the village band]₁ played under [a very
large tree]₂ while [a big cat]₃ slept on
[the window]₄

Nominal phrase = NP

Testing NPs according to the criteria:

STEP-1: We substitute NP₁ with NP₂ and viceversa. Not considering semantics, the structure is still correct.

[a very large tree]₂ played under [the
village band]₁ while [a big cat]₃ slept
on [the window]₄

1. the NPs were moved in **similar contexts**
2. the NPs were moved as **indivisible blocks**
3. the NPs were **replaced** with a constituent of the same type.

Testing NPs according to the criteria:

STEP-2: We substitute NP₂ with a PP.
The structure is ungrammatical.

[the village band]₁ played under

[in that summer]₂ while [a big cat]₃

slept on [the window]₄

1. the NPs cannot be moved in **different contexts**
2. the NPs cannot be moved as **divisible blocks**
3. **the NPs cannot be replaced** with a constituent of different type

Testing PPs according to the criteria:


Example: a sentence with marked PPs

the village band played [under a very
large tree]₁ while a big cat slept [on
the window]₂

Prepositional phrase = PP

Testing PPs according to the criteria:

STEP-1: We substitute PP₁ with PP₂ and viceversa. Not considering semantics, the structure is still correct.

the village band played [on the window]₁ while a big cat slept

[under a very large tree]₂

1. the PPs were moved in **similar contexts**
2. the PPs were moved as **indivisible blocks**
3. the PPs were **replaced** with a constituent of the same type.

Phrase structure

The main types of phrase are:

- S = sentence
- noun phrase = NP
- prepositional phrase = PP
- verbal phrase = VP
- adjectival phrase = AdjP
- adverbial phrase = AdvP
- ...

Phrase structure

Phrases can be nested inside each other:
the phrase structure consists precisely in hierarchically organized phrases.

We usually indicate them with the acronym of their name in English: NP = Noun Phras, VP = Verb Phrase, PP = Prepositional Phrase:

S = The cat of George run

S = [the cat of George]NP [run]VP

S = [the cat [of George]PP]NP [run]VP

Phrase structure

Knowledge of grammatical categories of words is fundamental in the construction of phrases, since each type of phrase contains a word that characterizes it in a peculiar way: name for NP, verb for VP, preposition for PP, etc.

Now it is further clear why Part of Speech tagging applies before parsing.

Phrase structure

To describe the constituent structure of a sentence there are various formalisms.

We refer in general to the Context Free Grammar (CFG), without going into the details of the particular names used for phrases or see how specific phenomena are treated.

Generative grammar, X-bar theory, generative-transformational grammar, ... are all specific grammatical formalisms that pivot on the notion of constituent (or phrase).

The Penn treebank format is an application of this kind of formalisms.

Phrase structure

A **constituency grammar** is a way of formally describing the constituent structure and consists of:

- a set of rules (also called **productions**) that express the way language symbols can be grouped and ordered
- a **lexicon** of words and symbols.

Example: **Grammar-1**

given the following set of **productions**:

NP → Article Noun

NP → Proper Noun

NP → Article Adjective Noun

and the **lexicon**:

Noun = cat, dog, bee

Article = the, a

Adjective = good, bad, nice

Proper Noun = Mary, John

the following **NPs** can be generated and are grammatical:

a cat, a good dog, Mary, John, the bee, the nice bee, the bad dog

...

A grammar (= a set of productions with a lexicon) can predict the phrases that can be generated as grammatical.

All the phrases that cannot be generated must be considered as ungrammatical for that grammar.

Example: *a good nice dog* is not grammatical for **Grammar-1** because it does not exist a sequence of rules of this grammar which allows the generation of *a good nice dog*

Phrase structure

Whenever we want to describe in a FORMAL way a natural language and describe it from the syntactic point of view with a constituent formalism, we need a grammar that includes rules of production and lexical components.

So we have to ask ourselves:

What are the rules by which I can describe this natural language?

What are the words (components or lexical symbols) of this natural language?

Phrase structure

What are the rules by which I can describe this natural language?

The answer depends on the language itself, because each language accepts certain syntactic structures and not others.

For example, in Italian the adjective may precede or follow the noun with which it forms the nominal phrase, but in English the adjective can only precede the noun. This means that different rules must be designed for the adjectival phrase in English and Italian respectively.

Phrase structure

What are the rules by which I can describe this natural language?

The answer depends on the language itself, because each language accepts certain syntactic structures and not others.

For example, in Italian the adjective may precede or follow the noun with which it forms the nominal phrase, but in English the adjective can only precede the noun. This means that different rules must be designed for the adjectival phrase in English and Italian respectively.

Language variation!

Phrase structure

What are the words (components or lexical symbols) of this natural language?

Each language has its own specific lexicon, that is, a collection of words that can be used within sentences, while all other words are not allowed.

Phrase structure

What are the words (components or lexical symbols) of this natural language?

Each language has its own lexicon, that is, a collection of words that can be used within sentences, while all other words are not allowed.

Language variation!

Phrase structure

Some rules also allow you to associate grammar and vocabulary, for example:

Article -> a

Article -> the

The symbols of the lexicon are called **terminals** (= words), all other **non-terminals** (= phrases).

This means that a CFG for a certain language is composed of a collection of terminal symbols (which are the words of the language) and rules that put together terminal and non-terminal symbols to build the sentence.

Phrase structure

A CFG for a certain language can be seen in several ways:

as **a tool to generate all the sentences of the language** (given the rules and vocabulary, it can generate all the correct sentences, for this reason it is also called generative grammar)

as **a tool to assign a structure to a given language sentence** (given a sentence, it recognizes the terminal nodes and binds them together according to rules to form phrases, if possible).

Phrase structure

as a tool to recognize the correctness of the structure of a given sentence of the language (given a sentence, it recognizes the terminal nodes and if it can find all the lexical symbols and rules necessary to bind them together, it recognizes the phrase as correct for that language).

Phrase structure

A CFG can also be seen as **the set of sentences it can generate** (for this reason it is also called generative).

All sentences that can be generated using grammar rules and vocabulary are called grammatical, while all sentences that cannot be generated with grammar are called non-grammatical.

Phrase structure

Given the grammar G (rules and vocabulary) and a sentence, a parser try to construct the syntactic structure/s of the sentence.

For example, we can build the structure of the phrase "*the dog sleeps*" as follows

1) we recognize the terminal symbols (words) of the phrase

The > Article

dog > Noun

sleeps > Verb

Phrase structure

2) we look for the rules that can link together the
recognized terminal symbols

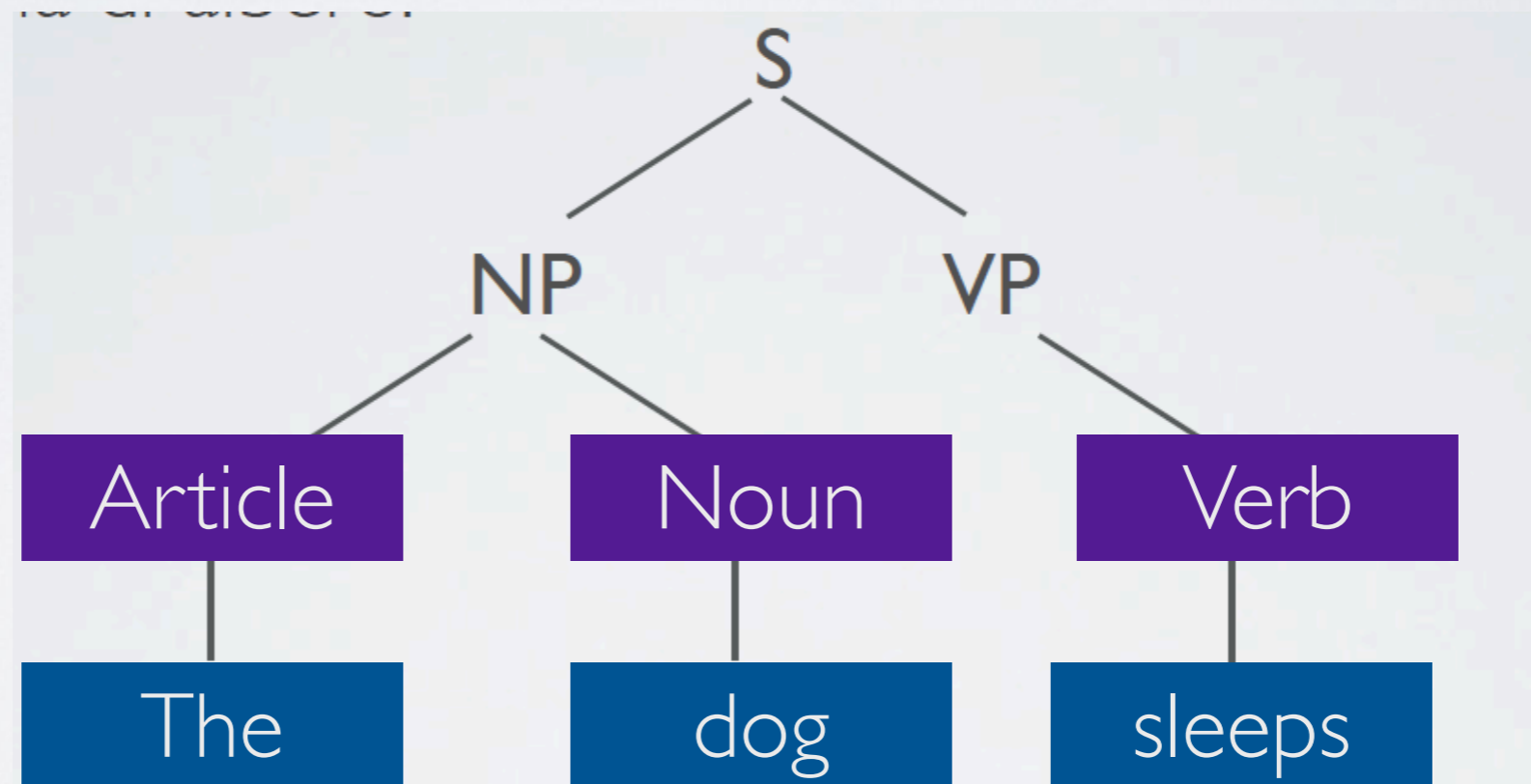
Article + Noun = NP

Verb = VP

NP + VP = S >>>this is the structure of the sentence "*the
dog sleeps*"

Phrase structure

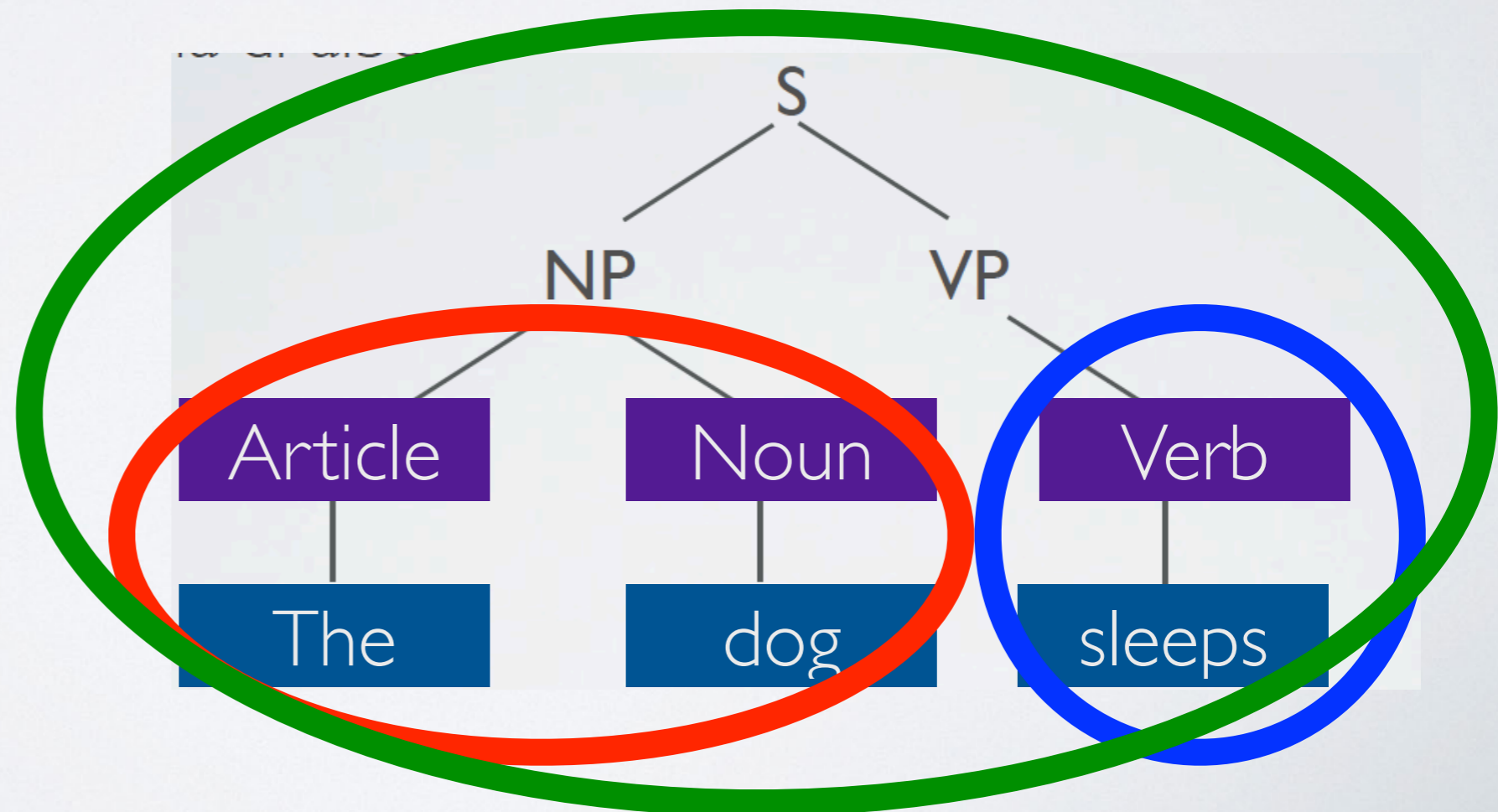
The result of the parsing task is usually provided as a graph called *syntactic tree* or *derivation tree* (it derives from the rules of a given grammar CFG).



Phrase structure

For every sentence a syntactic tree shows the **non linear structure underlying words.**

Words are considered in the exact order in which they occur in the sentence, but organised in a hierarchical structure composed of units and sub-units, words and then phrases nested one inside the other.



Penn Treebank

The Penn Treebank is one of the widely used linguistic resources. It includes a collection of texts in four different formats:

- Raw text
- Tagged with POS using a tagset which was developed as part of the project
- Parsed in constituent structure
- Combined, including both POS tags and constituent structure.

The Penn Treebank project has produced treebanks from the Brown, Switchboard, ATIS and Wall Street Journal corpora of English, as well as treebanks in Arabic and Chinese.

Penn Treebank

The Penn Treebank format for the syntactic level of annotation is a typical example of annotation based on constituency.

As for the PoS tag set, which is a simplified version of the Brown Corpus tag set, in this project, also for the syntactic format the researchers decided to include a limited amount of details.

It includes phrases represented with brackets around them.

Combined:

(S

(NP-SBJ

(NP (NNP Pierre)

(NNP Vinken))

(, ,)

(ADJP

(NP (CD 61)

(NNS years))

(JJ old))

(, ,))

(VP (MD will)

(VP (VB join)

(NP (DT the)

(NN board))

(PP-CLR (IN as)

(NP (DT a)

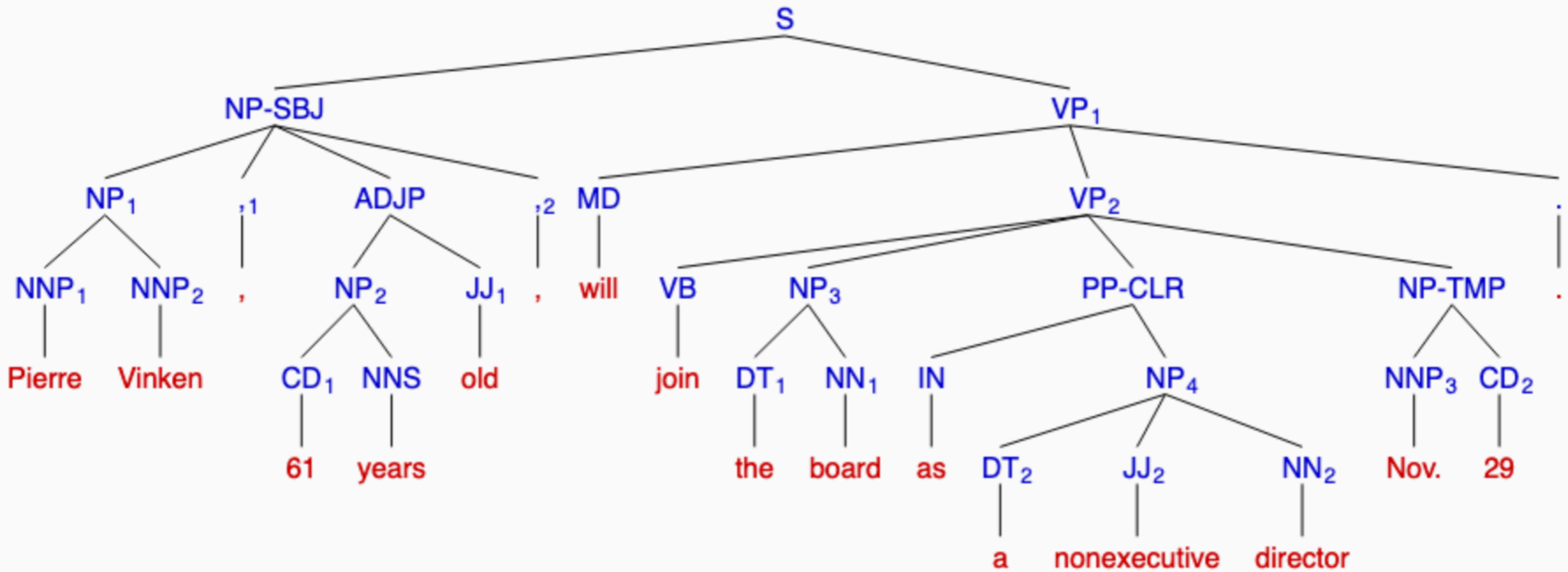
(JJ nonexecutive)

(NN director)))

(NP-TMP (NNP Nov.)

(CD 29))))

(. .)))



RECAP about Constituency structure

- The basic notion in constituency format is **PHRASE**
- Phrases are hierarchically organised
- A phrase is a group focused on a grammatical category that defines its type (noun > NP, verb > VP ...)
- In a constituency tree, phrases are the **non-terminal** nodes, while words are the **terminal** nodes
- A phrase has the same distributional properties as the other phrases of the same type
- The order of the phrases in the sentences is considered as fixed
- **Penn Treebank** format is based on constituency