

NLP Brief & Language Syntax

Natural Language

Textual data is unstructured data but it usually belongs to a specific language following specific syntax and semantics. Any piece of text data, a simple word, sentence or a document relates back to some natural language most of the time. In this section, we will be looking at the definition of natural language, the philosophy of language, language acquisition and the usage of language.

What is Natural Language?

To understand text analytics and natural language processing, we need to understand what makes a language "natural". In simple terms, a natural language is a language developed and evolved by humans through natural use and communication rather than constructing and creating the language artificially like a computer programming language.

Various human languages like English, Japanese or even Sanskrit can be called as natural languages. Natural languages can be communicated in different forms including speech, writing or even signs. There has been a lot of interest in trying to understand the origins, nature and philosophy of language. We will discuss it briefly in the following section.

Linguistics

We have seen what natural language means, how language is learnt and used and the origins of language acquisition. In fact a lot of these things are actually formally researched and studied in linguistics by researchers and scholars called linguists. Formally, linguistics is defined as the scientific study of language including form and syntax of language, meaning and semantics depicted by the usage of language and context of use. The origins of linguistics can be dated back to the 4th century BCE, when Indian scholar and linguist Panini formalized the Sanskrit language description. The term linguistics was first defined to indicate the scientific study of languages in 1847 approximately before which the term philology was used to indicate the same.

Language Syntax and Structure

We already know what language syntax and structure indicate. Syntax and structure usually go hand in hand where a set of specific rules, conventions and principles usually govern the way words are combined into phrases, phrases get combined into clauses and clauses get combined into sentences. We will be talking specifically about the English language syntax and structure in this section since in this book we will be dealing with textual data which belongs to the English language. However a lot of these concepts can be extended to other languages too. Knowledge about the structure and syntax of language is helpful in many areas like text processing, annotation and parsing for further operations like text classification or summarization.

In English, usually words combine together to form other constituent units. These constituents include words, phrases, clauses and sentences. All these constituents exist together in any message and are related to each other in a hierarchical structure. Moreover, a sentence is a structured format of representing a collection of words provided they follow certain syntactic rules like grammar. Let's take a sample sentence, "The brown fox is quick and he is jumping over the lazy dog". Following snippet shows us how the sentence looks like in Python.

```
sentence = "The brown fox is quick and he is jumping over the lazy dog"
sentence
```

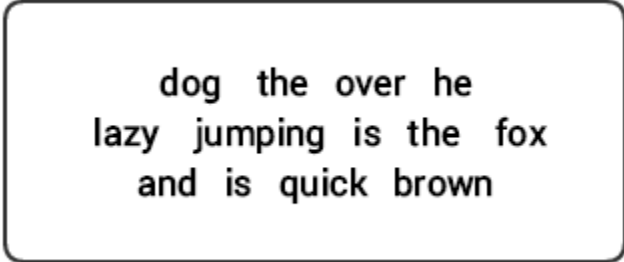
```
'The brown fox is quick and he is jumping over the lazy dog'
```

Grammar and ordering of words definitely gives meaning to a sentence. What if we jumbled up the words? Would it still make sense?

```
words = sentence.split()
np.random.shuffle(words)
print(words)
```

```
['quick', 'is', 'fox', 'brown', 'The', 'and', 'the', 'is', 'he', 'dog', 'lazy',
'jumping', 'over']
```

This un-ordered bunch of words as depicted in the previous output and also represented in Figure 1-3 is definitely hard to make sense of, isn't it?



dog the over he
lazy jumping is the fox
and is quick brown

Figure 1-3. A collection of words without any relation or structure

From the collection of words in Figure 1-3, it is very difficult to ascertain what it might be trying to convey or mean. Indeed, languages are not just comprised of a bag or bunch of unstructured words. Sentences with proper syntax not only help us give proper structure and relate words together but also help them convey meaning based on order or position of the words. Considering our previous hierarchy of *sentence* → *clause* → *phrase* → *word*, we can construct the following hierarchical sentence tree using shallow parsing, a technique often used for finding out the constituents in a sentence.

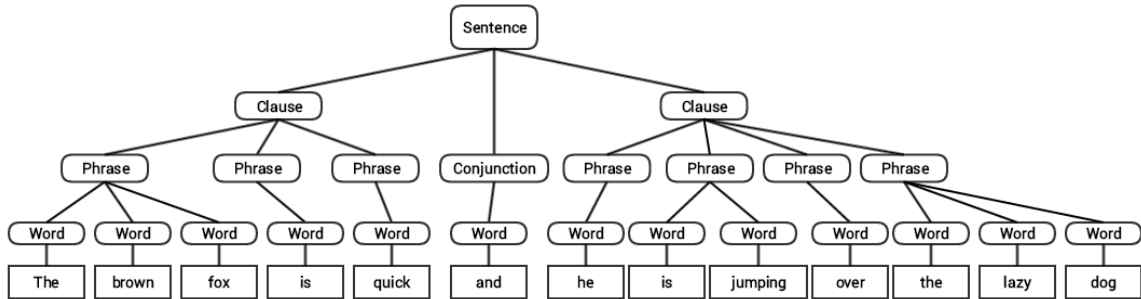


Figure 1-4. Structured sentence following the hierarchical syntax

From the above hierarchical tree, we get the following sentence, "The brown fox is quick and he is jumping over the lazy dog" with a sentence of structure and meaning. We can see that the leaf nodes of the tree consist of words which are the smallest unit here and combination of words form phrases which in turn form clauses. Clauses are connected together through various filler terms or words like conjunctions and they form the final sentence. In the following section, we will look at each of these constituents in further detail and understand how to analyze them and find out what the major syntactic categories are.

Words

Words are the smallest unit in a language which are independent and have a meaning of their own. Although morphemes are the smallest distinctive units, they are not independent like words and a word can be comprised of several morphemes. It is useful to annotate and tag words and analyze them into their parts of speech (POS) and see the major syntactic categories. Here, we will cover the main categories and significance of the various POS tags. Usually, words can fall into one of the following major categories.

- **N(oun)**: This usually denotes words which depict some object or entity which could be living or non-living. Some examples would be *fox*, *dog*, *book* and so on. The POS tag symbol for nouns is **N**.

- **V(erb)**: Verbs are words which are used to describe certain actions, states or occurrences. There are a wide variety of further sub-categories like auxiliary, reflexive, transitive verb and many more. Some typical examples of verbs would be *running*, *jumping*, *read* and *write*. The POS tag symbol for verbs is **V**.
- **Adj(ective)**: Adjectives are words which are used to describe or qualify other words, typically nouns and noun phrases. The phrase "*beautiful flower*" has the noun (N) "*flower*" which is described or qualified using the adjective (ADJ) "*beautiful*". The POS tag symbol for adjectives is **ADJ**.
- **Adv(erb)**: Adverbs usually act as modifiers for other words including nouns, adjectives, verbs or adverbs themselves. The phrase "*very beautiful flower*" has the adverb (ADV) "*very*" which modifies the adjective (ADJ) "*beautiful*" indicating the degree of how beautiful the flower is. The POS tag symbol for adverbs is **ADV**.

Besides these four major categories of parts of speech, there are other categories also which occur frequently in the English language. These include pronouns, prepositions, interjections, conjunctions, determiners and many others. Each POS tag like **nouns (N)** can be further sub-divided into various categories like **Singular Nouns (NN)**, **Singular Proper Nouns (NNP)** and **Plural Nouns (NNS)**. We will be looking at POS tags in further detail in Chapter 3 when we process and parse textual data and implement POS taggers to annotate text. Considering our previous example sentence "The brown fox is quick and he is jumping over the lazy dog" we can leverage nltk or spacy in python to annotate it with POS tags.

```
pos_tags = nltk.pos_tag(sentence.split())
pd.DataFrame(pos_tags).T
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|-----|-------|-----|-----|-------|-----|-----|-----|---------|------|-----|------|-----|
| 0 | The | brown | fox | is | quick | and | he | is | jumping | over | the | lazy | dog |
| 1 | DT | JJ | NN | VBZ | JJ | CC | PRP | VBZ | VBG | IN | DT | JJ | NN |

Figure 1-5. Annotated words with their parts of speech tags using NLTK

You can still leverage spacy to understand the high level semantics for each tag annotation.

```
spacy_pos_tagged = [(word, word.tag_, word.pos_) for word in nlp(sentence)]
pd.DataFrame(spacy_pos_tagged).T
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|-----|-------|------|------|-------|-------|------|------|---------|------|-----|------|------|
| 0 | The | brown | fox | is | quick | and | he | is | jumping | over | the | lazy | dog |
| 1 | DT | JJ | NN | VBZ | JJ | CC | PRP | VBZ | VBG | IN | DT | JJ | NN |
| 2 | DET | ADJ | NOUN | VERB | ADJ | CCONJ | PRON | VERB | VERB | ADP | DET | ADJ | NOUN |

Figure 1-6. Annotated words with their parts of speech tags using spaCy

It is interesting to see that based on the output depicted in Figure 1-6, the tag annotations match in both frameworks and internally they use the Penn Treebank notation for POS tag annotation. Tying this back to our discussion, considering a simple annotation of our sentence using basic POS tags, it would look as depicted in Figure 1-7.

| | | | | | | | | | | | | |
|-----|-------|-----|----|-------|------|------|----|---------|------|-----|------|-----|
| DET | ADJ | N | V | ADJ | CONJ | PRON | V | V | ADV | DET | ADJ | N |
| The | brown | fox | is | quick | and | he | is | jumping | over | the | lazy | dog |

Figure 1-7. Annotated words with their parts of speech tags

From the above example, you might see a few unknown tags. The tag **DET** stands for determiner which is used to depict articles like *a*, *an*, *the* etc. The tag **CONJ** indicates conjunction which is usually used to bind together clauses to form sentences and the **PRON** tag stands for pronoun which represents words which are used to represent or take the place of a noun. The tags, **N**, **V**, **ADJ** and **ADV** are typical open classes and represent words belonging to an open vocabulary. Open classes are word classes which consist of an infinite set of words and commonly accept addition of new words to the vocabulary which are invented by people

Phrases

Words have their own lexical properties like parts of speech which we saw earlier. Using these words, we can order them in ways which give meaning to the words such that each word belongs to a corresponding phrasal category and one of the words is the main or head word. In the hierarchy tree, groups of words make up phrases which form the third level in the syntax tree. By principle, phrases are assumed to have at least two or more words considering the pecking order of *words* ← *phrases* ← *clauses* ← *symbols*. However, a phrase can be a single word or a combination of words based on the syntax and position of the phrase in a clause or sentence. For example, the sentence, *"Dessert was good"* has only three words and each of them roll up to three phrases. The word *"Dessert"* is a noun as well as a noun phrase, *"is"* depicts a verb as well as a verb phrase and *"good"* represents an

adjective as well as an adjective phrase describing the aforementioned dessert. There are five major categories of phrases which are described below.

- **Noun Phrase (NP):** These are phrases where a noun acts as the head word. Noun phrases act as a subject or object to a verb. Usually noun phrases can be a set of words which can be replaced by a pronoun without rendering the sentence or clause syntactically incorrect. Some examples would be, *"dessert"*, *"the lazy dog"*, *"the brown fox"*.
- **Verb Phrase (VP):** These phrases are lexical units which have a verb acting as the head word. Usually there are two forms of verb phrases, one form which has the verb components as well as other entities like nouns adjectives or adverbs which are a part of the object. The verb here is known as a finite verb. It acts as a single unit in the hierarchy tree and can function as the root in a clause. This form is prominent in constituency grammars. The other form is where the finite verb acts as the root of the entire clause and is prominent in dependency grammars. Another derivation of this includes verb phrases strictly consisting of verb components including main, auxiliary, infinitive and participles. The following sentence, *"He has started the engine"* can be used to illustrate the two types of verb phrases which can be formed. They would be *"has started the engine"* and *"has started"* based on the two forms which we discussed above.
- **Adjective Phrase (ADJP):** These are phrases whose head word is an adjective. Their main role is to describe or qualify nouns and pronouns in a sentence and they will be either placed before or after the noun or pronoun. The sentence, *"The cat is too quick"* has an adjective phrase, *"too quick"* qualifying the cat which is a noun phrase.
- **Adverb Phrase (ADVP):** These phrases act like an adverb since the adverb acts as the head word in the phrase. Adverb phrases are used as modifiers for nouns, verbs or adverbs themselves by providing further details to describe or qualify them. Considering the sentence, *"The train should be at the station pretty soon"*, the adjective phrase is *"pretty soon"* since it describes when the train would be arriving.
- **Prepositional Phrase (PP):** These phrases usually contain a preposition as the head word and other lexical components like nouns, pronouns etc. It acts like an adjective or adverb describing other words or phrases. The sentence *"Going up the stairs"* has a prepositional phrase, *"up"* describing the direction of the stairs.

These five major syntactic categories of phrases can be generated from words using several rules some of which we discussed above like utilizing syntax and grammars of different types. We will be exploring some of the popular grammars in a later section. Shallow parsing is a popular natural

language processing technique to extract these constituents including POS tags as well as phrases from a sentence. For our sentence, *"The brown fox is quick and he is jumping over the lazy dog"*, one way of representing it using shallow parsing is to have seven phrases as depicted in Figure 1-8.

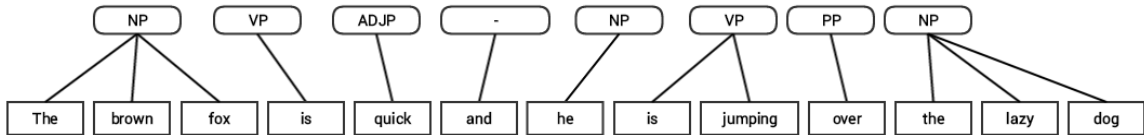


Figure 1-8. Annotated phrases with their tags in shallow parsing

The phrase tags fall into the categories we discussed earlier, however the word *"and"* is a conjunction and is usually used to combine clauses together. Is there a better way to do this? Probably! You can define your own rules for phrases and then enable shallow parsing or chunking using a lookup based parser similar to nltk's `RegexpParser` which is a grammar based chunk parser, which uses a set of regular expression patterns (defined grammar rules) to specify the behavior of the parser. Following code shows it in action for our sentence!

```

grammar = '''
    NP: {<DT>?<JJ>?<NN.*>}
    ADJP: {<JJ>}
    ADVP: {<RB.*>}
    PP: {<IN>}
    VP: {<MD>?<VB.*>+}
'''

pos_tagged_sent = nltk.pos_tag(sentence.split())
rp = nltk.RegexpParser(grammar)
shallow_parsed_sent = rp.parse(pos_tagged_sent)
print(shallow_parsed_sent)

(S
  (NP The/DT brown/JJ fox/NN)
  (VP is/VBZ)
  (ADJP quick/JJ)
  and/CC
  he/PRP
  (VP is/VBZ jumping/VBG)
  (PP over/IN)
  (NP the/DT lazy/JJ dog/NN))

# visualize shallow parse tree
shallow_parsed_sent

```

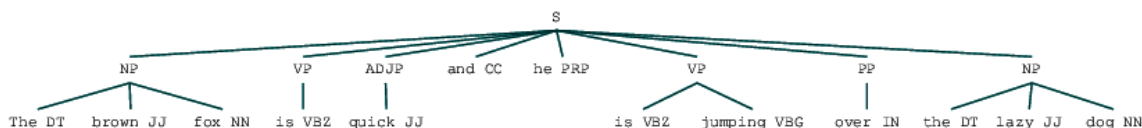


Figure 1-9. Annotated phrases with their tags in shallow parsing using NLTK

Based on the output in Figure 1-9, we can see the power of a simple rule-based chunker in identifying major phrases and structure in our sentence. In the following section, we will be looking at clauses, their main categories and some conventions and syntactic rules for extracting clauses from sentences.

Clauses

By nature, clauses can act as independent sentences or several clauses can be combined together to form a sentence. Clauses are a group of words with some relation between them and usually contain a subject and a predicate. Sometimes the subject may not be present and the predicate usually has a verb phrase or a verb with an object. By default you can classify clauses into two distinct categories, the main clause and the subordinate clause. The main clause is also known as an independent clause because it can form a sentence by itself and act as both a sentence as well as a clause. The subordinate or dependent clause cannot exist just by itself and depends on the main clause for its meaning. They are usually joined with other clauses using dependent words like subordinating conjunctions.

Since we are talking a lot about syntactic properties of language, clauses can be sub-divided into several categories based on syntax. They are explained in detail as follows.

- **Declarative:** These clauses usually occur quite frequently and they denote statements which do not have any specific tone associated with it. These are just standard statements which are declared with a neutral tone, which could be factual or non-factual. An example would be, *"Grass is green"*.
- **Imperative:** These clauses are usually in the form of a request, command, rule or advice. The tone in this case would be a person issuing an order to one or more people to carry out an order, request or even an instruction. An example would be, *"Please do not talk in class"*.

- **Relative:** The simplest interpretation of relative clauses is that they are subordinate clauses and hence dependent on another part of the sentence which usually contains a word, phrase or even a clause. This element usually acts as the antecedent to one of the words from the relative clause and relates to it. A simple example would be the following sentence, *"John just mentioned that he wanted a soda"* having the antecedent proper noun, *"John"* which was referred to in the relative clause, *"he wanted a soda"*.
- **Interrogative:** These clauses usually are in the form of questions. The type of these questions can be either affirmative or negative. Some example would be, *"Did you get my mail?"* and *"Didn't you go to school?"*
- **Exclamative:** These clauses are used to express shock, surprise or even complements. All these expressions fall under exclamations and these clauses often end with an exclamation mark. An example would be, *"What an amazing race!"*

Usually most clauses will be expressed in one of the above mentioned syntactic forms however this list of clause categories is not an exhaustive list and it can be further categorized into several other forms. Considering our example sentence, *"The brown fox is quick and he is jumping over the lazy dog"* if you remember the syntax tree, the co-ordinating conjunction (and) divides the sentence into two clauses. They are, *"The brown fox is quick"* and *"he is jumping over the lazy dog"*. Can you guess what categories they might fall into? (Hint: Look back at the definitions of Declarative and Relative Clauses).

Grammar

Grammar helps in enabling both syntax and structure in language. It primarily consists of a set of rules which are used in determining how to position words, phrases and clauses when constructing sentences for any natural language. Grammar is not only restricted to the written word but also when used verbally. These rules can be specific to a region, language, dialect or be somewhat universal like the Subject-Verb-Object (SVO) model. Origins of grammar have a rich history starting with Sanskrit in India. In the West, the study of grammar originated with the Greeks and the earliest work was the *Art of Grammar*, written by Dionysius Thrax. Latin grammar models were developed from the Greek models and gradually across several ages, grammars for various languages started being created. It was only in the 18th century that grammar was considered as a serious candidate for being a field under linguistics.

Grammars have been developed over the course of time and they have kept evolving leading to the birth of newer types of grammars and various older grammars slowly lost prominence. Hence grammar is not just a fixed set of rules but also its evolution based on the usage of language over the course of time amongst humans. Considering the English language as before, there are several ways in which grammars can be classified. We will first talk about two broad

classes into which most of the popular grammatical frameworks can be grouped and then we will further explore how these grammars represent language. Grammar can be sub-divided into two main classes based on their representations for linguistic syntax and structure. They are as follows.

- **Dependency Grammars**
- **Constituency Grammars**

Dependency Grammars: These grammars are a class of grammars which specifically does not focus on constituents (unlike constituency grammars) like words, phrases and clauses but gives more emphasis on words. Dependencies in this context are labeled word-word relations or links which are usually asymmetrical

Dependency grammars always have a one-to-one relationship correspondence for each word in the sentence. There are two aspects to this grammar representation. One is the syntax or structure of the sentence and the other is the semantics obtained from the relationships denoted between the words. Each directed edge represents a specific type of meaningful relationship (also known as syntactic function) and we can annotate our sentence further showing the specific dependency relationship types between the words. The same is depicted in Figure 1-11 considering our sentence, *"The brown fox is quick and he is jumping over the lazy dog"*.

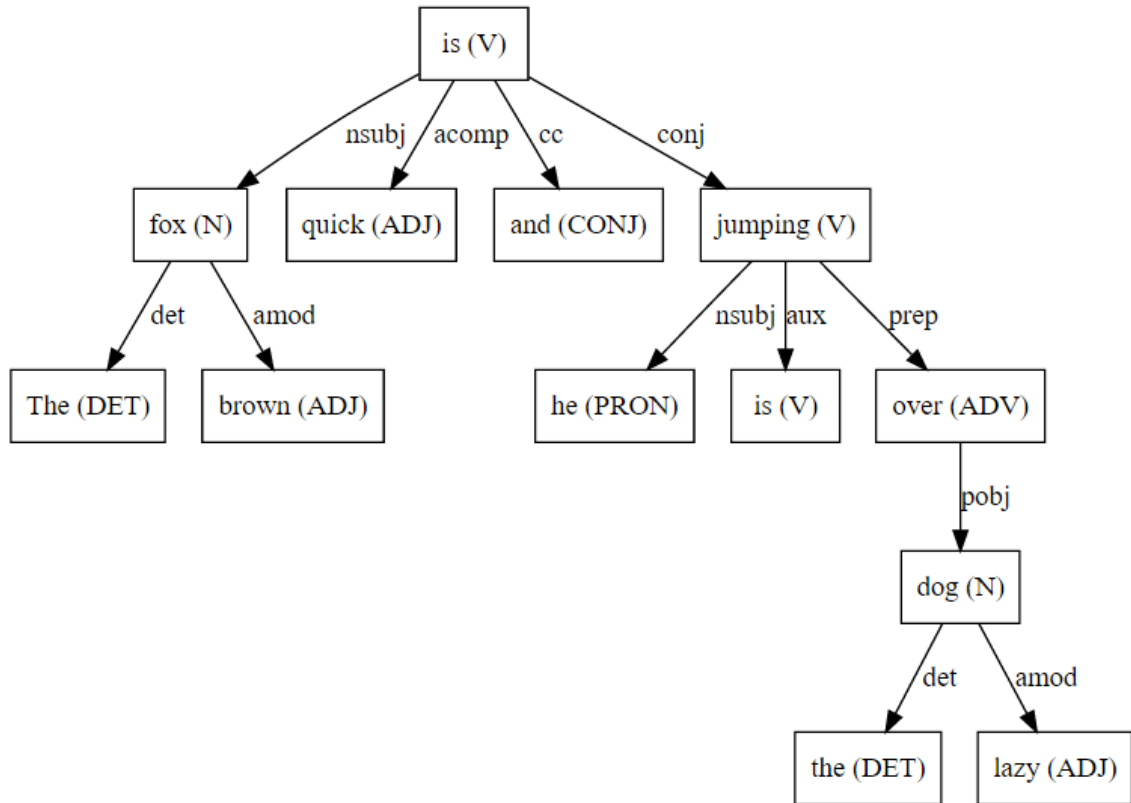


Figure 1-11. Dependency grammar based syntax tree annotated with dependency relationship types

We will look in detail some of the tags used in the dependencies for the sentence in the above figure.

- The dependency tag **det** is pretty intuitive that it denoted the determiner relationship between a nominal head and the determiner. Usually the word with POS tag DET will also have the det relation. Examples include (fox -> the) and (dog -> the)
- The dependency tag **amod** stands for adjectival modifier and stands for any adjective which modifies the meaning of a noun. Examples include (fox -> brown) and (dog -> lazy)
- The dependency tag **nsubj** stands for an entity which acts as a subject or agent in a clause. Example include (is -> fox) and (jumping -> he)

- The dependencies **cc** and **conj** are more with linkages related to words connected by co-ordinating conjunctions. Examples include (is -> and) and (is -> jumping)
- The dependency tag **aux** indicates the auxiliary or secondary verb in the clause. Examples include (jumping -> is)
- The dependency tag **acom** stands for adjective complement and acts as the complement or object to a verb in the sentence. Examples include (is -> quick)
- The dependency tag **prep** denotes a prepositional modifier which usually modifies the meaning of a noun, verb, adjective or even a preposition. Usually this representation is used for prepositions having a noun or noun phrase complement. Examples include (jumping -> over)
- The dependency tag **pobj** is used to denote the object of a preposition. This is usually the head of a noun phrase following a preposition in the sentence. Examples include (over -> dog)

The above tags have been extensively used in our sample sentence for annotating the various dependency relationships among the words. Instead of creating a tree with linear orders, you can also represent it with a normal graph since there is no concept of order of words in dependency grammar. We can leverage spacy to build us this dependency tree\graph for our sample sentence.

```
from spacy import displacy
```

```
displacy.render(nlp(sentence), jupyter=True,
                 options={'distance': 100,
                          'arrow_stroke': 1.5,
                          'arrow_width': 8})
```

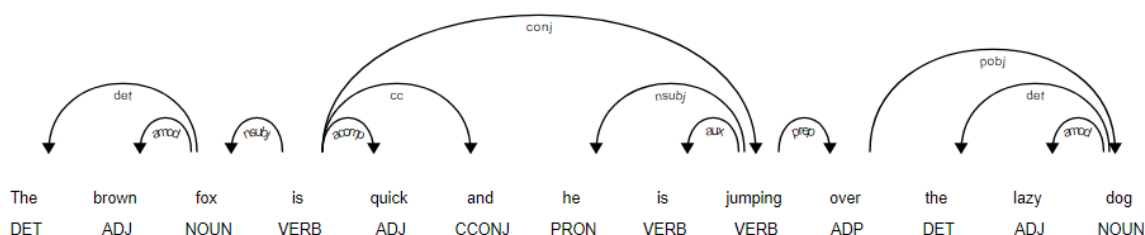


Figure 1-12. Dependency grammar annotated graph for our sample sentence with spacy

Figure 1-12 depicts our sentence annotated with dependency tags which should be clear to you based on our earlier discussion. Next, we will look at the concepts behind constituency grammars and their representation.

Constituency Grammars: These grammars are a class of grammars which are built upon the principle that a sentence can be represented by several constituents derived from it. These grammars can be used to model or represent the internal structure of sentences in terms of a hierarchically ordered structure of its constituents. Each and every word usually belongs to a specific lexical category in the case and forms the head words of different phrases. These phrases are formed based on rules called as phrase structure rules. Hence constituency grammars are also called as "phrase structure grammars".

Phrase structure rules form the core of constituency grammars since they talk about syntax and rules which govern the hierarchy and ordering of the various constituents in the sentences. These rules cater to two things primarily. The generic representation of a phrase structure rule is $S \rightarrow AB$ which depicts that the structure S consists of constituents A and B and the ordering is A followed by B .

There are several phrase structure rules and we will explore them one by one to understand how exactly do we extract and order constituents in a sentence. The most important rule describes how to divide a sentence or a clause. The phrase structure rule denotes a binary division for a sentence or a clause as $S \rightarrow NP VP$ where S is the sentence or clause and it is divided into the subject, denoted by the Noun Phrase (NP) and the predicate, denoted by the Verb Phrase (VP). Of course we can apply further rules to break down each of the constituents further but the top level of the hierarchy always starts usually with a NP and VP. The rule for representing a Noun Phrase is of the form, $NP \rightarrow [DET][ADJ]N [PP]$ where the square brackets [...] denote that it is optional. We look at a few examples which are governed by the aforementioned rules for noun phrases in Figure 1-13.

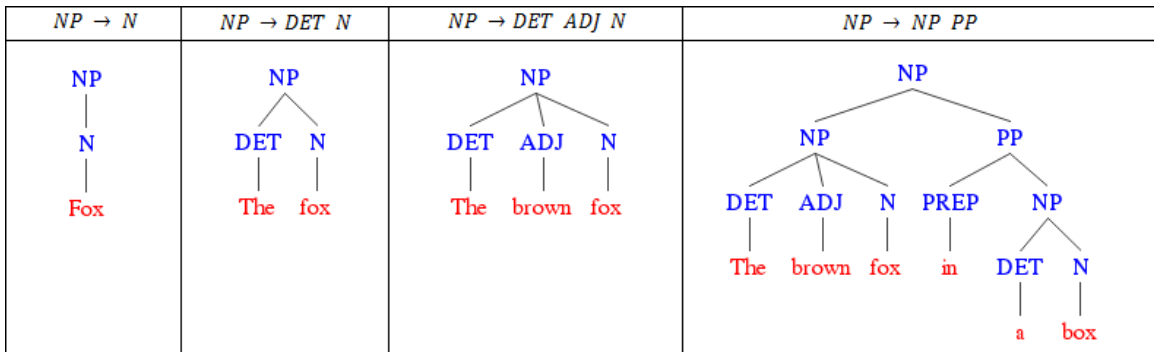


Figure 1-13. Constituency Syntax Trees depicting structuring rules for Noun Phrases

A noun phrase denoted by NP on the left side of the production rule may also appear on the right side of the production rule as depicted in the last example. This is a property called recursion and we will talk about it towards the end of this section. We will now look at rules for representing Verb

Phrases. The rule is of the form $VP \rightarrow V \mid MD [VP][NP][PP][ADJP][ADVP]$ where the head word is usually a **(V)**erb or a modal **(MD)**. A modal is itself an auxiliary verb but we give it a different representation just to distinguish itself from the normal verb. Figure 1-14 depicts a few examples for the different types of verb phrases which can be typically constructed and their representations as syntax trees.

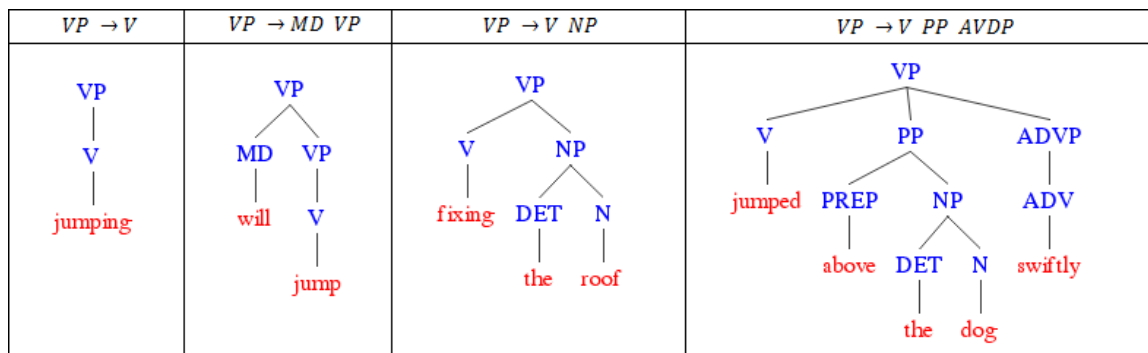


Figure 1-14. Constituency Syntax Trees depicting structuring rules for Verb Phrases

The above syntax trees show us the representations of the various constituents in verb phrases and using the property of recursion, a verb phrase may also contain another verb phrase inside it as we see in the second syntax tree. Let us look at the production rules for representing Prepositional Phrases. The basic rule has the form $PP \rightarrow PREP [NP]$ where **PREP** denotes a preposition which acts as the head word and it is optionally followed by a Noun Phrase **(NP)** often. Figure 1-15 depicts some representations of prepositional phrases and their corresponding syntax trees.

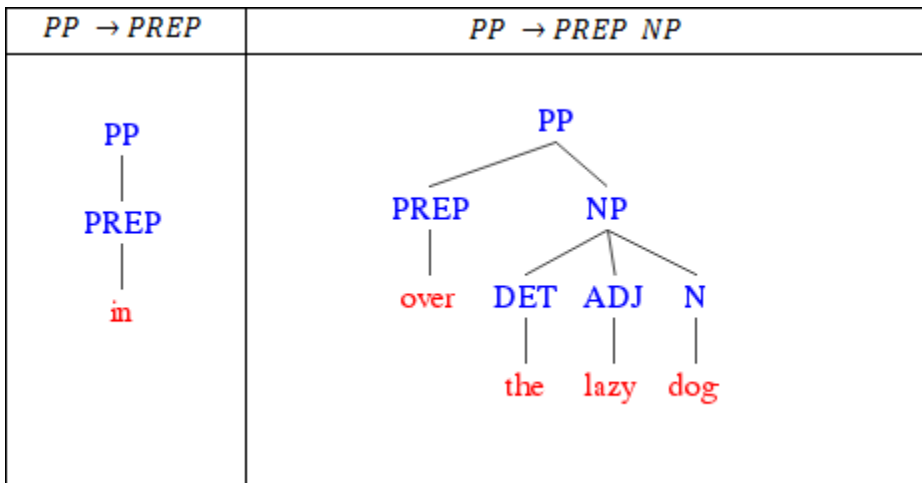


Figure 1-15. Constituency Syntax Trees depicting structuring rules for Prepositional Phrases

The above syntax trees show us some different representations for prepositional phrases. A simple example representation of the sentence, *"The flying monkey in the circus on the trapeze by the river"* is depicted by the constituency parse tree in Figure 1-16.

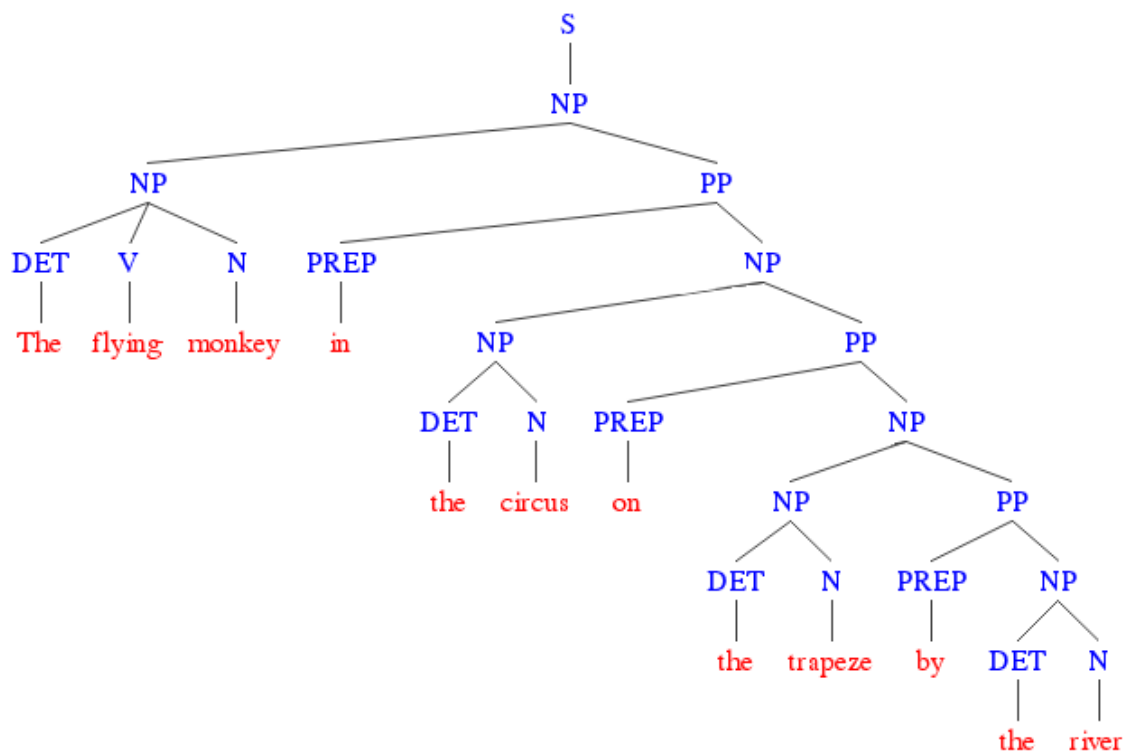


Figure 1-16. *Constituency Syntax Tree depicting recursive properties among constituents*

If you closely observe the above syntax tree closely, you will notice that it is only constituted of noun phrases and prepositional phrases. However due to the inherent recursive property that a prepositional phrase itself can consist of a noun phrase and the noun phrase can consist of a noun phrase as well as a prepositional phrase.

A simple example for a sentence consisting of a noun phrase which by itself, is constructed out of two noun phrases and a conjunction would be, "The brown fox and the lazy dog". This is depicted by the constituency syntax tree showing the adherence to the production rule in Figure 1-17.

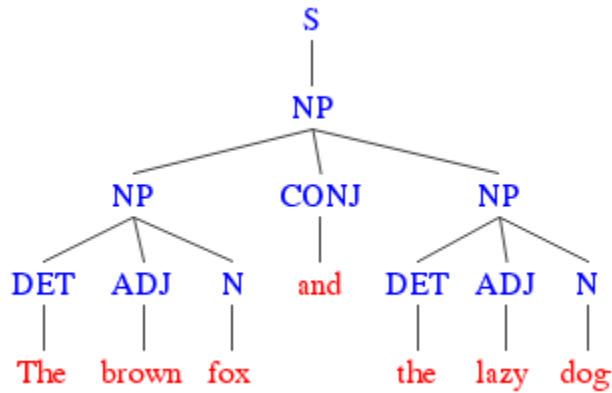


Figure 1-17. Constituency Syntax Tree depicting noun phrases joined by a conjunction

The above figure shows us that the top level noun phrase is the sentence by itself and it has two noun phrases as its constituents which are joined together by a conjunction thus satisfying our aforementioned production rule. What if we wanted to join two sentences or clauses together with a conjunction? We can do that by putting all of these rules and conventions together and generate the constituency based syntax tree for our sample sentence, *"The brown fox is quick and he is jumping over the lazy dog"*. This would give us the syntactic representation of our sentence as depicted in Figure 1-18.

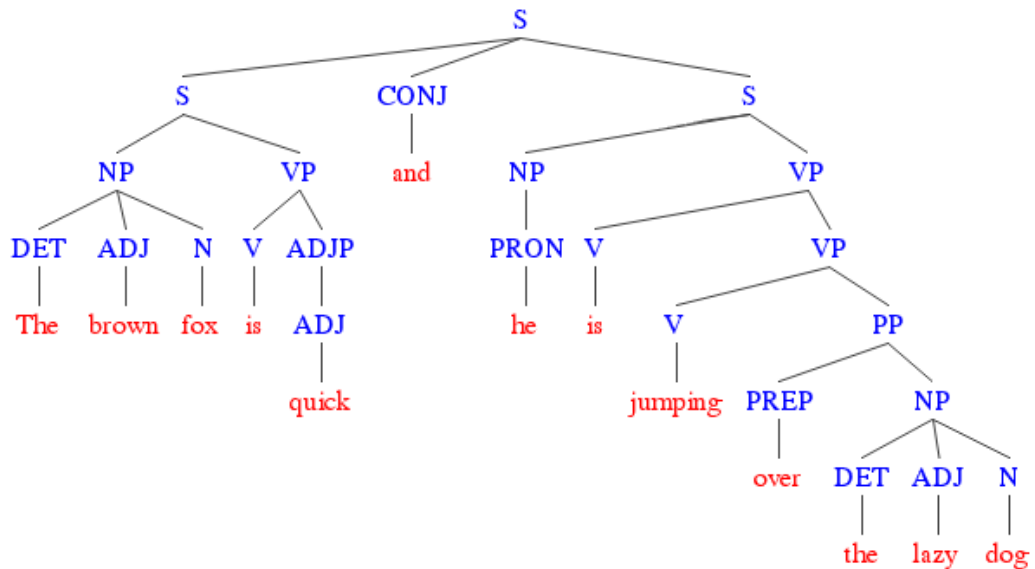


Figure 1-18. *Constituency Syntax Tree for our sample sentence*

From Figure 1-18, you can conclude that our sentence has two main clauses or constituents which we had talked about earlier and they are joined together by a co-ordinating conjunction (and).

We can leverage Stanford's Core NLP based parsers in nltk to perform constituency parsing on our sample sentence.

```
from nltk.parse.stanford import StanfordParser

scp = StanfordParser(path_to_jar='E:/stanford/stanford-parser-full-2015-04-20/stanford-parser.jar',
                     path_to_models_jar='E:/stanford/stanford-parser-full-2015-04-20/stanford-parser-3.5.2-models.jar')

result = list(scp.raw_parse(sentence))
print(result[0])

(ROOT
  (NP
    (S
      (S
        (NP (DT The) (JJ brown) (NN fox))
        (VP (VBZ is) (ADJP (JJ quick))))
      (CC and)
      (S
        (NP (PRP he))
        (VP
          (VBZ is)
          (VP
            (VBG jumping)
            (PP (IN over) (NP (DT the) (JJ lazy) (NN dog))))))))))

# visualize constituency tree
result[0]
```

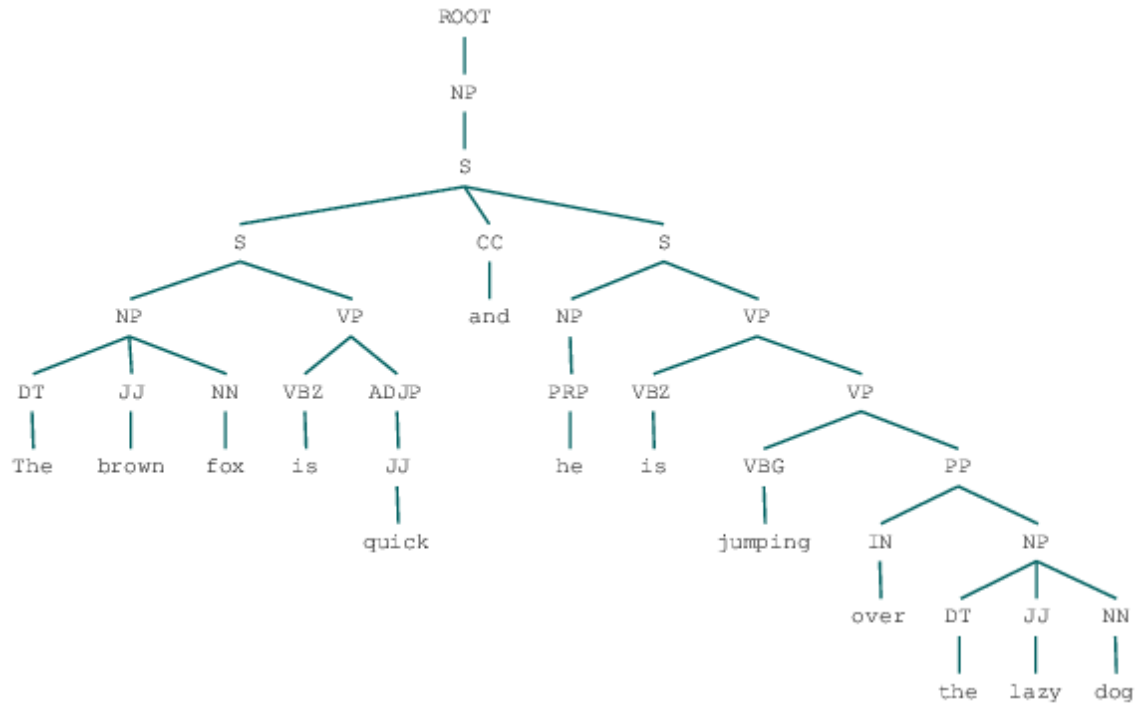


Figure 1-19. *Constituency Syntax Tree for our sample sentence with NLTK and Stanford Core NLP*

We can see that the constituency tree depicted in Figure 1-19 has the same hierarchical structure based on the tree we had talked about in Figure 1-18.