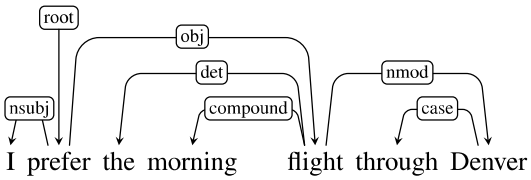
CHAPTER

19 Dependency Parsing

> Tout mot qui fait partie d'une phrase... Entre lui et ses voisins, l'esprit aperçoit des connexions, dont l'ensemble forme la charpente de la phrase.
>
> [Between each word in a sentence and its neighbors, the mind perceives **connections**. These connections together form the scaffolding of the sentence.]
>
> Lucien Tesnière. 1959. Éléments de syntaxe structurale, A.1.§4

dependency grammars

The focus of the last chapter was on context-free grammars and constituent-based representations. Here we present another important family of grammar formalisms called **dependency grammars**. In dependency formalisms, phrasal constituents and phrase-structure rules do not play a direct role. Instead, the syntactic structure of a sentence is described solely in terms of directed binary grammatical relations between the *words*, as in the following dependency parse:



(19.1)

typed dependency

Relations among the words are illustrated above the sentence with directed, labeled arcs from **heads** to **dependents**. We call this a **typed dependency structure** because the labels are drawn from a fixed inventory of grammatical relations. A *root* node explicitly marks the root of the tree, the head of the entire structure.

Figure 19.1 on the next page shows the dependency analysis from Eq. 19.1 but visualized as a tree, alongside its corresponding phrase-structure analysis of the kind given in the prior chapter. Note the absence of nodes corresponding to phrasal constituents or lexical categories in the dependency parse; the internal structure of the dependency parse consists solely of directed relations between words. These head-dependent relationships directly encode important information that is often buried in the more complex phrase-structure parses. For example, the arguments to the verb *prefer* are directly linked to it in the dependency structure, while their connection to the main verb is more distant in the phrase-structure tree. Similarly, *morning* and *Denver*, modifiers of *flight*, are linked to it directly in the dependency structure. This fact that the head-dependent relations are a good proxy for the semantic relationship between predicates and their arguments is an important reason why dependency grammars are currently more common than constituency grammars in natural language processing.

free word order

Another major advantage of dependency grammars is their ability to deal with languages that have a relatively **free word order**. For example, word order in Czech can be much more flexible than in English; a grammatical *object* might occur before or after a *location adverbial*. A phrase-structure grammar would need a separate rule

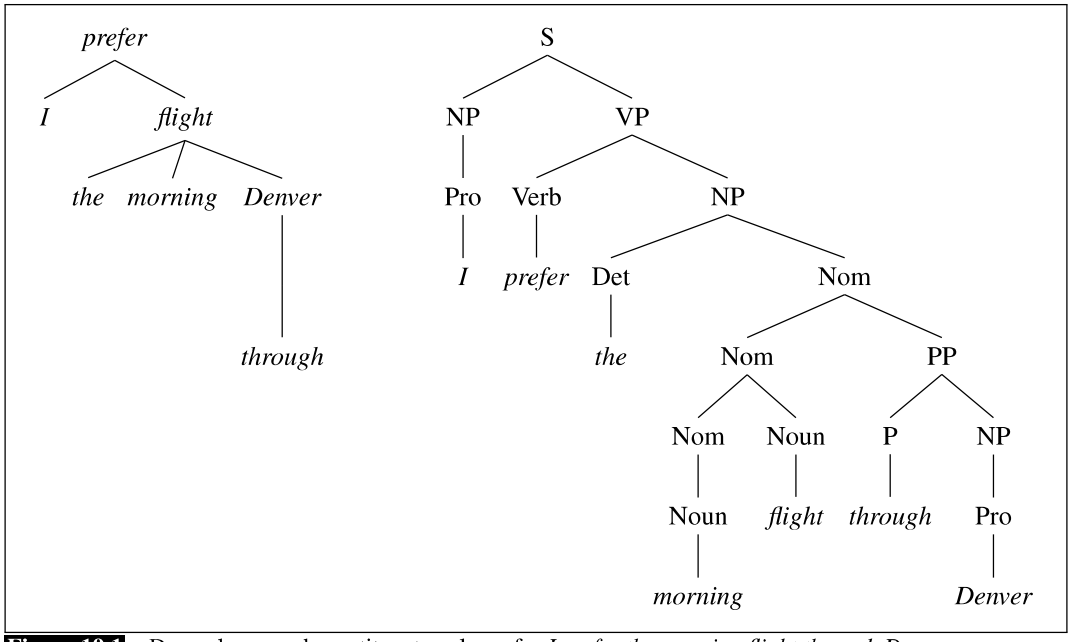


Figure 19.1 Dependency and constituent analyses for *I prefer the morning flight through Denver.*

for each possible place in the parse tree where such an adverbial phrase could occur. A dependency-based approach can have just one link type representing this particular adverbial relation; dependency grammar approaches can thus abstract away a bit more from word order information.

In the following sections, we'll give an inventory of relations used in dependency parsing, discuss two families of parsing algorithms (transition-based, and graph-based), and discuss evaluation.

19.1 Dependency Relations

grammatical relation

The traditional linguistic notion of **grammatical relation** provides the basis for the binary relations that comprise these dependency structures. The arguments to these relations consist of a **head** and a **dependent**. The head plays the role of the central organizing word, and the dependent as a kind of modifier. The head-dependent relationship is made explicit by directly linking heads to the words that are immediately dependent on them.

head

dependent

grammatical function

In addition to specifying the head-dependent pairs, dependency grammars allow us to classify the kinds of grammatical relations, or **grammatical function** that the dependent plays with respect to its head. These include familiar notions such as *subject*, *direct object* and *indirect object*. In English these notions strongly correlate with, but by no means determine, both position in a sentence and constituent type and are therefore somewhat redundant with the kind of information found in phrase-structure trees. However, in languages with more flexible word order, the information encoded directly in these grammatical relations is critical since phrase-based constituent syntax provides little help.

Linguists have developed taxonomies of relations that go well beyond the familiar notions of subject and object. While there is considerable variation from theory

Clausal Argument Relations	Description
NSUBJ	Nominal subject
OBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

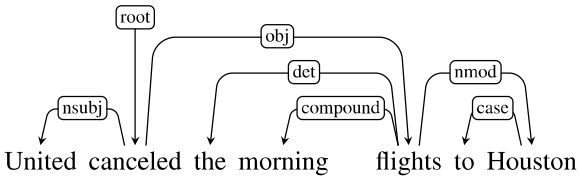
Figure 19.2 Some of the Universal Dependency relations (de Marneffe et al., 2021).

Universal Dependencies

to theory, there is enough commonality that cross-linguistic standards have been developed. The **Universal Dependencies** (UD) project (de Marneffe et al., 2021), an open community effort to annotate dependencies and other aspects of grammar across more than 100 languages, provides an inventory of 37 dependency relations. Fig. 19.2 shows a subset of the UD relations and Fig. 19.3 provides some examples.

The motivation for all of the relations in the Universal Dependency scheme is beyond the scope of this chapter, but the core set of frequently used relations can be broken into two sets: clausal relations that describe syntactic roles with respect to a predicate (often a verb), and modifier relations that categorize the ways that words can modify their heads.

Consider, for example, the following sentence:



(19.2)

Here the clausal relations NSUBJ and OBJ identify the subject and direct object of the predicate *cancel*, while the NMOD, DET, and CASE relations denote modifiers of the nouns *flights* and *Houston*.

19.1.1 Dependency Formalisms

A dependency structure can be represented as a directed graph $G = (V, A)$, consisting of a set of vertices V, and a set of ordered pairs of vertices A, which we'll call arcs.

For the most part we will assume that the set of vertices, V, corresponds exactly to the set of words in a given sentence. However, they might also correspond to punctuation, or when dealing with morphologically complex languages the set of vertices might consist of stems and affixes. The set of arcs, A, captures the head-dependent and grammatical function relationships between the elements in V.

Different grammatical theories or formalisms may place further constraints on these dependency structures. Among the more frequent restrictions are that the structures must be connected, have a designated root node, and be acyclic or planar. Of most relevance to the parsing approaches discussed in this chapter is the common,

Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
OBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
COMPOUND	We took the **morning** *flight*.
NMOD	*flight* to **Houston**.
AMOD	Book the **cheapest** *flight*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which *flight* was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

Figure 19.3 Examples of some Universal Dependency relations.

dependency tree

computationally-motivated, restriction to rooted trees. That is, a **dependency tree** is a directed graph that satisfies the following constraints:

1. There is a single designated root node that has no incoming arcs.
2. With the exception of the root node, each vertex has exactly one incoming arc.
3. There is a unique path from the root node to each vertex in V.

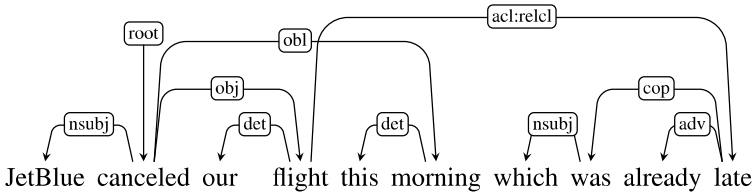
Taken together, these constraints ensure that each word has a single head, that the dependency structure is connected, and that there is a single root node from which one can follow a unique directed path to each of the words in the sentence.

19.1.2 Projectivity

projective

The notion of projectivity imposes an additional constraint that is derived from the order of the words in the input. An arc from a head to a dependent is said to be **projective** if there is a path from the head to every word that lies between the head and the dependent in the sentence. A dependency tree is then said to be projective if all the arcs that make it up are projective. All the dependency trees we've seen thus far have been projective. There are, however, many valid constructions which lead to non-projective trees, particularly in languages with relatively flexible word order.

Consider the following example.



(19.3)

In this example, the arc from *flight* to its modifier *late* is non-projective since there is no path from *flight* to the intervening words *this* and *morning*. As we can see from this diagram, projectivity (and non-projectivity) can be detected in the way we've been drawing our trees. A dependency tree is projective if it can be drawn with no crossing edges. Here there is no way to link *flight* to its dependent *late* without crossing the arc that links *morning* to its head.

Our concern with projectivity arises from two related issues. First, the most widely used English dependency treebanks were automatically derived from phrase-structure treebanks through the use of head-finding rules. The trees generated in such a fashion will always be projective, and hence will be incorrect when non-projective examples like this one are encountered.

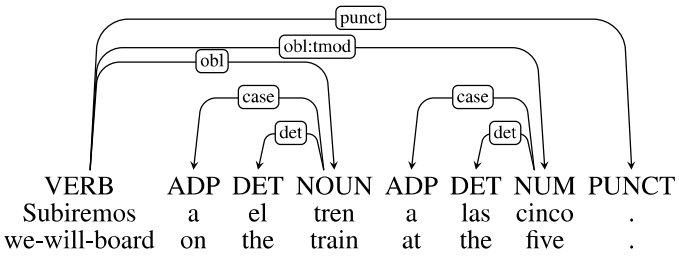
Second, there are computational limitations to the most widely used families of parsing algorithms. The transition-based approaches discussed in Section 19.2 can only produce projective trees, hence any sentences with non-projective structures will necessarily contain some errors. This limitation is one of the motivations for the more flexible graph-based parsing approach described in Section 19.3.

19.1.3 Dependency Treebanks

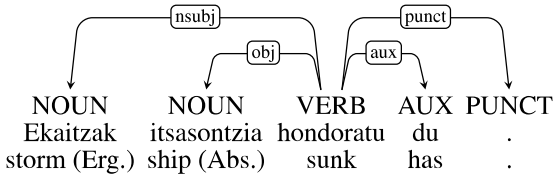
Treebanks play a critical role in the development and evaluation of dependency parsers. They are used for training parsers, they act as the gold labels for evaluating parsers, and they also provide useful information for corpus linguistics studies.

Dependency treebanks are created by having human annotators directly generate dependency structures for a given corpus, or by hand-correcting the output of an automatic parser. A few early treebanks were also based on using a deterministic process to translate existing constituent-based treebanks into dependency trees.

The largest open community project for building dependency trees is the Universal Dependencies project at `https://universaldependencies.org/` introduced above, which currently has almost 200 dependency treebanks in more than 100 languages (de Marneffe et al., 2021). Here are a few UD examples showing dependency trees for sentences in Spanish, Basque, and Mandarin Chinese:



[Spanish] Subiremos al tren a las cinco. "We will be boarding the train at five."(19.4)



[Basque] Ekaitzak itsasontzia hondoratu du. "The storm has sunk the ship."(19.5)