Syntax analysis

Contents

Syntax analysis

- Syntactic analysis or parsing or syntax analysis is the third phase of NLP.
- Because it helps us understand the roles played by different words in a body of text. The words themselves are not enough.
- ***Innocent peacefully children sleep little*** **vs**
- ***Innocent little children sleep peacefully***
- There is some evidence that human understanding of language is, in part, based on structural analysis.

What is a language?

- To do syntactic analysis, we build a **parser**,
- i.e., a software systems that checks whether the rules are followed and that provides an analysis based on the grammar.

Why syntax analysis

- determiner noun verb determiner noun, as in
- **The cat ate the fish.**
- det noun prep adj noun verb prep det noun, as in
- **The dog with one eye ran from the cat.**
- But none of these helps us determine the relationships between the words.

Syntax analysis

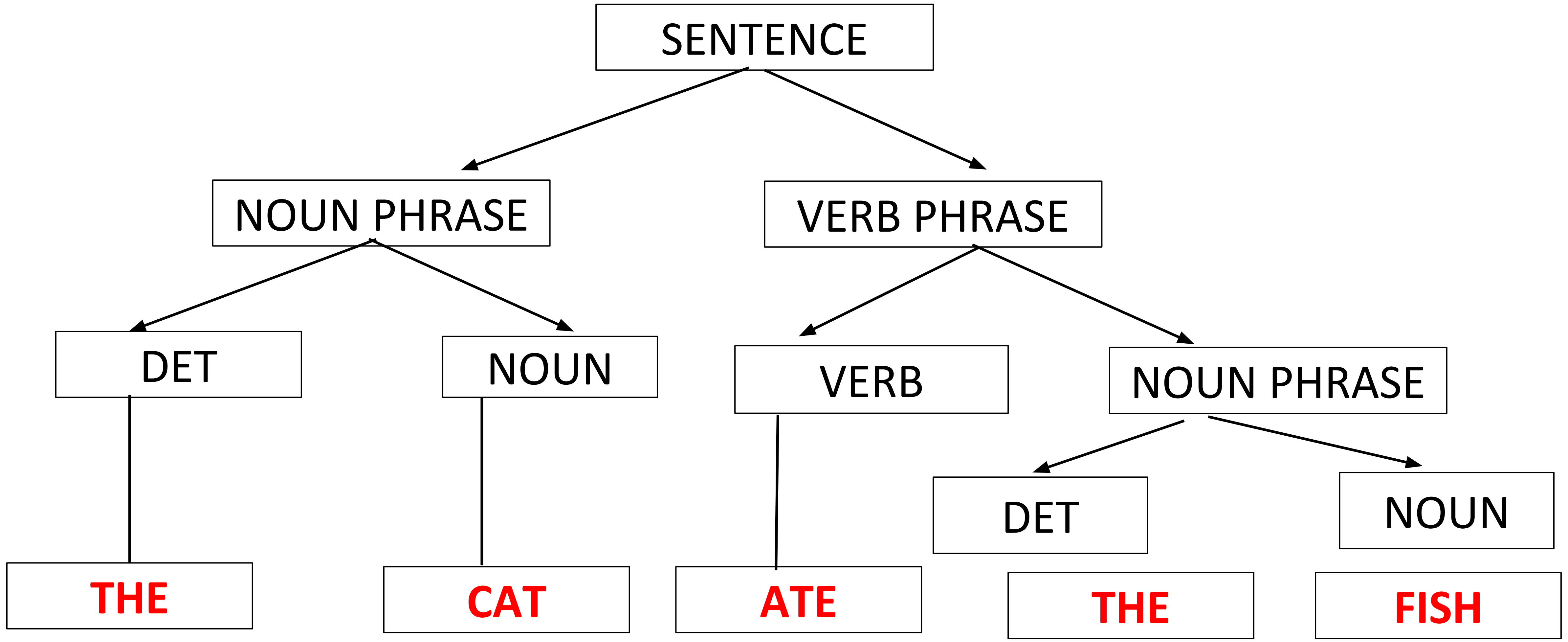
- We want a grammar to give us more structural information. So, for example, some better grammar rules for sentences might be:
- Sentence -> NP VP
- (i.e., a sentence can be formed from a noun phrase followed by a verb phrase).

Syntax analysis

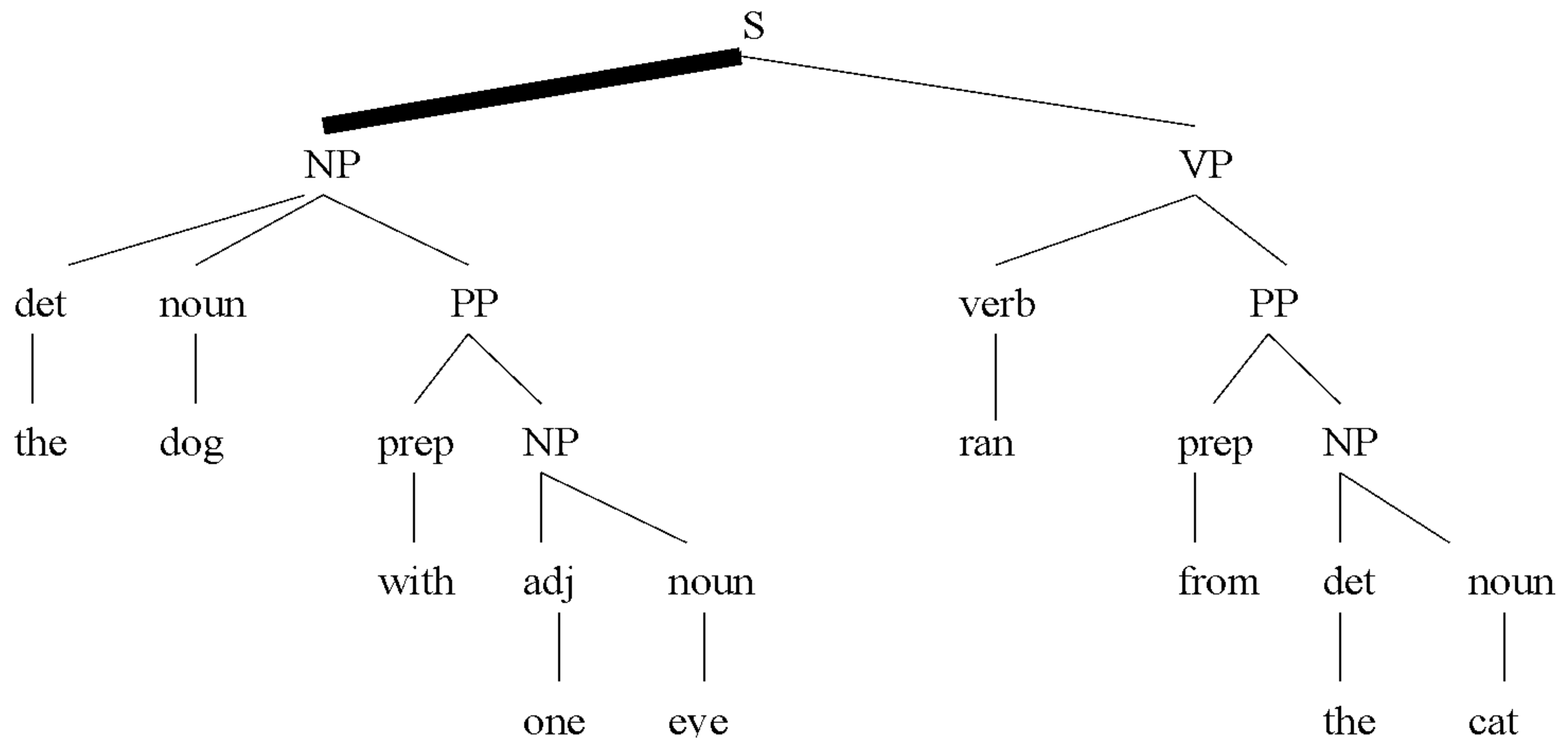
- NP -> noun (a noun phrase can be just a single noun)
- NP -> det noun (a noun phrase can be a determiner followed by a noun)
- VP -> verb (a verb phrase can be just a single verb)
- VP -> verb NP (a verb phrase can be a verb followed by a noun phrase)

Syntax analysis

- *The cat ate the fish*
- our grammar could help us determine the following structure, which we call a **parse tree**:



PARSE TREE


S
NP
VP
det
noun
PP
verb
PP
the
dog
prep
NP
ran
prep
NP
with
adj
noun
from
det
noun
one
eye
the
cat

Components of a syntactic analysis program

- In order to perform syntactic analysis, we need a **parser** - i.e.,
- a **program** that takes as input a sentence and produces the analysis.
- a **grammar** - i.e., a set of rules that the parser can use.
- a **lexicon** - i.e., a dictionary of legal words and their parts of speech

POS TAGGING

- Tagging is a kind of classification that may be defined as the automatic assignment of description to the tokens.
- In simple words, we can say that POS tagging is a task of labelling each word in a sentence with its appropriate part of speech.
- We already know that parts of speech include **nouns, verb, adverbs, adjectives, pronouns, conjunction and their sub-categories.**

WHY POS Tagging

- Part-of-Speech tagging in itself may not be the solution to any particular NLP problem.
- It is however something that is done as a

pre-requisite to simplify a lot of different problems.

PENN tree bank

- the Penn Treebank **tagset** is based on that of the Brown Corpus.
- The Penn Treebank produced approximately
- 7 million words of part-of-speech tagged text,
- 3 million words of skeletally parsed text,
- over 2 million words of text parsed for predicate argument structure, and
- 1.6 million words of transcribed spoken text annotated for speech disfluencies.

Tag set for English (Penn Tree bank)

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker

11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*

26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

Types of POS taggers

- POS-tagging algorithms fall into two distinctive groups:

- **Rule-Based POS Taggers**
- **Stochastic POS Taggers**

Rule-based POS Tagging

- One of the oldest techniques of tagging is rule-based POS tagging.
- It use contextual information to assign tags to unknown or ambiguous words. Disambiguation is done by analysing the linguistic features of the word, its preceding word, its following word, and other aspects.
- For example,
- if the preceding word is an article, then the word in question must be a noun. This information is coded in the form of rules.

- **Defining a set of rules manually is an extremely cumbersome process and is not scalable at all. So we need some automatic way of doing this.**

Properties of Rule-Based POS Tagging

- Rule-based POS taggers possess the following properties −
- These taggers are knowledge-driven taggers.
- The rules in Rule-based POS tagging are built manually.
- The information is coded in the form of rules.
- We have some limited number of rules approximately around 1000.
- Smoothing and language modelling is defined explicitly in rule-based taggers.

Rule-Based POS Tagging

- Following Rules are applied to identify different Tags
- 1. Noun Identification Rules Rule
- If word is adjective then there is high probability that next word will be noun.
- For Example:- वह एक सच्चा देशभक्त है।
- In above example सच्चा is adjective and देशभक्त is noun.

Rule-Based POS Tagging

- Rule 2: If word is relative pronoun then there is high probability that next word will be noun.

- For Example:- ये वो घर है जिसे राजा ने बनवाया था|

- In above example वो and जिसे is relative pronoun and घर (ghar) and राजा is noun.

Stochastic POS tagging

- The simplest stochastic taggers disambiguate words based solely on the probability that a word occurs with a particular tag.
- In other words, the tag encountered most frequently in the training set with the word is the one assigned to an ambiguous instance of that word.
- The problem with this approach is that while it may yield a valid tag for a given word, it can also yield inadmissible sequences of tags.

$P(word|tag)$

Properties of Stochastic POS Tagging

- Stochastic POS taggers possess the following properties −
- This POS tagging is based on the probability of tag occurring.
- It requires training corpus
- There would be no probability for the words that do not exist in the corpus.
- It uses different testing corpus (other than training corpus).
- It is the simplest POS tagging because it chooses most frequent tags associated with a word in training corpus.
-

Stochastic tagger

- **Word Frequency Approach**
- In this approach, the stochastic taggers disambiguate the words based on the probability that a word occurs with a particular tag.
- The tag encountered most frequently with the word in the training set is the one assigned to an ambiguous instance of that word.
- The main issue with this approach is that it may yield inadmissible sequence of tags.

WORD FREQUENCY APPROACH

- Cat$_{NN}$ ate$_{VB}$ the rat$_{NN}$. Cat$_{NN}$ was running$_{VBD}$ behind the tree. Cat$_{NN}$ is seen with the mat$_{NN}$.

- TEST INSTANCE
- Cat$_{NN}$ is sitting on the mat$_{NN}$

Stochastic tagger

- **Tag Sequence Probabilities**
- It is another approach of stochastic tagging, where the tagger calculates the probability of a given sequence of tags occurring.

- It is also called n-gram approach. It is called so because the best tag for a given word is determined by the probability at which it occurs with the n previous tags.
- It is based on mathematical equation taking account the PD of occurrences of one POS followed by another in a sentence sequence.

Stochastic tagger

- Assign each word its most likely POS tag
- If we have tags t1,t2……tk, then we can use
- P(ti|w)= c(w,ti)/ c(w,t1)+c(w,t2)+…………..+c(w,tk)

- C(w,ti)= number of times w/ti appears in the corpus

- **Heat:: noun/89, verb/5**
- P(noun|heat)= 89/ 89+5 = **0.94**
- P(Verb|heat) = 5/ 89+5= 0.05

Issues in tagging

- The main problem with POS tagging is **ambiguity**.
- In English, many common words have multiple meanings and therefore multiple POS.
- The job of a POS tagger is to resolve this ambiguity accurately based on the context of use.
- WORD SENSE DISAMBIGUATION
- She saw a **bear.**
- Your efforts will **bear** fruit.

- **Silver** is too costly
- She gave a **silver** talk

Issues in tagging

- Tagging unknown words
  - All tagging algorithms require dictionary that provides POS of every word
  - But even the largest dictionary cannot contain every possible word
  - Proper nouns are not part of dictionary

Issues in tagging

- To build a tagger, we need some method for guessing the tag of unknown words

- PD of tags over unknown words is very similar to the distribution of tags over words that occurred only once in the training set.
- Such words are more likely to be nouns followed by words.
- Thus the likelihood for an unknown word is determined by the average of the distribution over all singleton words in the training set

Context free grammar

- **A *context-free grammar (CFG)* is a list of rules that define the set of all well-formed sentences in a language.**
- Each rule has a left-hand side, which identifies a syntactic category, and a right-hand side, which defines its alternative component parts, reading from left to right.
- E.g., the rule s --> np vp
- means **that "a sentence is defined as a noun phrase followed by a verb phrase”**

Context free grammar

- Figure 1 shows a simple CFG that describes the sentences from a small subset of English.

Figure 1. A grammar and a parse tree for "the giraffe dreams".

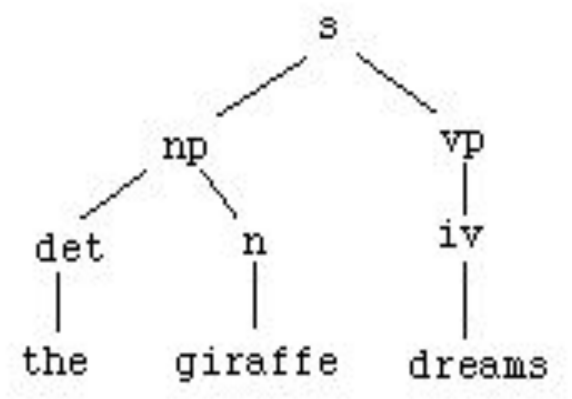
```
s    --> np vp

np   --> det n
vp   --> tv np
     --> iv

det  --> the
     --> a
     --> an

n    --> giraffe
     --> apple

iv   --> dreams
tv   --> eats
     --> dreams
```



Context free grammar

- A *sentence* in the language defined by a CFG is a series of words that can be derived by systematically applying the rules, beginning with a rule that has s on its left-hand side. A

- *parse* of the sentence is a series of rule applications in which a syntactic category is replaced by the right-hand side of a rule that has that category on its left-hand side, and the final rule application yields the sentence itself

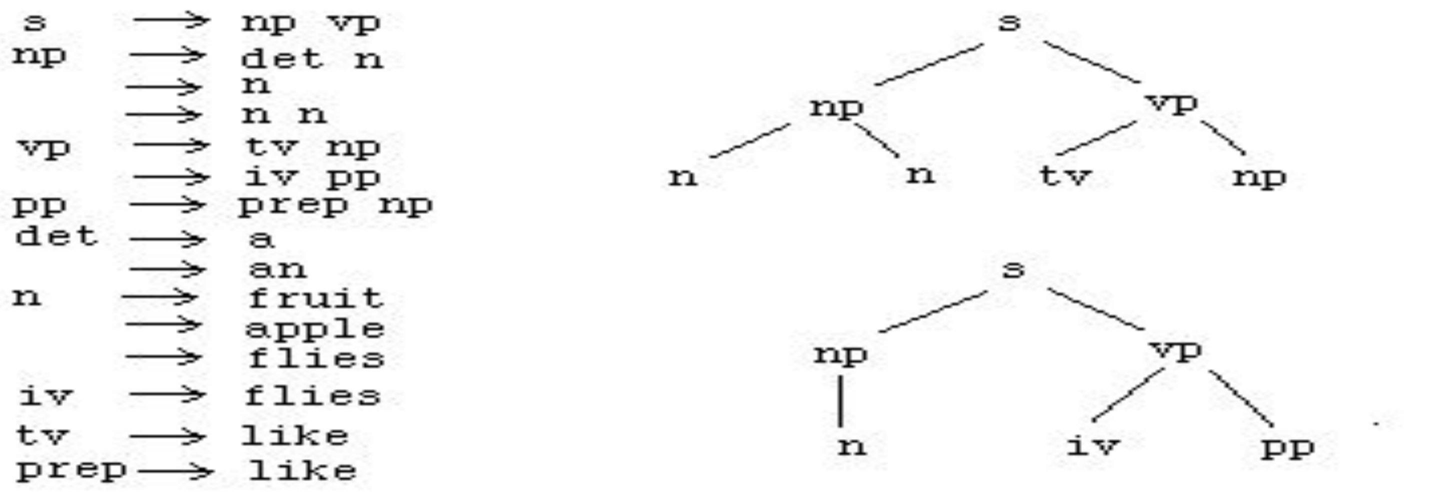
Context free grammar

- a parse of the sentence "the giraffe dreams" is:
- s => np vp
- => det n vp
- => the n vp
- => the giraffe vp
- => the giraffe iv
- => the giraffe dreams

Goals of Linguistic Grammars

- Permit ambiguity - ensure that a sentence has all its possible parses (E.g., "fruit flies like an apple" in Figure 2)

Figure 2. An ambiguous grammar and partial parse trees for "fruit flies like an apple."



```
s    ⟶ np vp
np   ⟶ det n
     ⟶ n
     ⟶ n n
vp   ⟶ tv np
     ⟶ iv pp
pp   ⟶ prep np
det  ⟶ a
     ⟶ an
n    ⟶ fruit
     ⟶ apple
     ⟶ flies
iv   ⟶ flies
tv   ⟶ like
prep ⟶ like
```

Sequence labeling Hidden Markov Model