NLTK

Natural Language Processing

NLP

Natural Language Processing (NLP) – обработка естественного языка – подраздел информатики и ИИ, посвященный тому, как компьютеры анализируют естественные (человеческие) языки.

Область применения

- ➔ Распознавания речи
- ➔ Обобщение документов
- ➔ Машинный перевод
- ➔ Выявления спама
- ➔ Предиктивного ввода текста

NLTK

- ➔ Ведущая платформа для создания NLP-программ на Python.
- ➔ Легкие в использовании интерфейсы для многих языковых корпусов*
- ➔ Библиотеки для обработки текстов, для классификации, токенизации, стемминга, разметки, фильтрации и семантических рассуждений

★ Языковой корпус – подобранная и обработанная по определенным правилам совокупность текстов, используемых в качестве базы для исследования языка

Установка

1) Устанавливаем библиотеку с помощью pip

```
pip install nltk
```

2) Как обычно импортируем библиотеку

```
import nltk
```

3) Загружаем нужные модели и корпуса текстов

```
nltk.download()
```

Основы

Токенизация по предложениям

Токенизация по предложениям – это процесс разделения письменного языка на предложения-компоненты.

➔ Точка – хороший разделитель предложений
➔ Что делать с сокращениями? (i.e., e.g., p.m.)

Пример

```
text = (
    "Backgammon is one of the oldest known board games. Its history can be traced "
    "back nearly 5,000 years to archeological discoveries in the Middle East. It is "
    "a two player game where each player has fifteen checkers which move between "
    "twenty-four points according to the roll of two dice."
)

sentences = nltk.sent_tokenize(text)

print(sentences)
```

```
['Backgammon is one of the oldest known board games.', 'Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.', 'It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.']
```

Токенизация по словам

Токенизация по словам – это процесс разделения предложений на слова-компоненты.

➔ Пробел – хороший разделитель слов
➔ Что делать с составными существительными? (bus stop, battle royal)

Пример

```
sentence = "Backgammon is one of the oldest known board games."

words = nltk.word_tokenize(sentence)

print(words)
```

```
['Backgammon', 'is', 'one', 'of', 'the', 'oldest', 'known',
'board', 'games', '.']
```

Лемматизация и стемминг

Стемминг – процесс нахождения основы слова (неизменяемая часть слова, которая выражает его лексическое значение).

Лемматизация – это более тонкий процесс, который использует словарь и морфологический анализ, чтобы в итоге привести слово к его канонической форме – лемме.

Примеры

- ➔ Слово **good** – это лемма для слова *better*. Стеммер не увидит эту связь, так как здесь нужно сверяться со словарем.
- ➔ Слово **play** – это базовая форма слова *playing*. Тут справятся и стемминг, и лемматизация.
- ➔ Слово **meeting** может быть как нормальной формой существительного, так и формой глагола *to meet*, в зависимости от контекста. В отличие от стемминга, лемматизация попробует выбрать правильную лемму, опираясь на контекст.

Примеры

```
from nltk.stem import PorterStemmer, WordNetLemmatizer

ps = PorterStemmer()
wnl = WordNetLemmatizer()

sentence = "Programmer program with programming languages"
words = nltk.word_tokenize(sentence)

for word in words:
   print(f"{word}: {ps.stem(word)}, {wnl.lemmatize(word)}")
```

Programmer: programm, Programmer
program: program, program
with: with, with

programming: program, programming
languages: languag, language

Стоп-слова

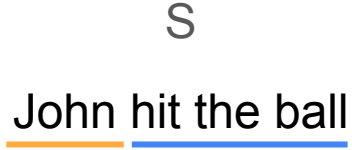
- ➔ **Стоп-слова** – это слова, которые выкидываются из текста до/после обработки текста.
- ➔ Нерелевантны, поэтому создают шум при машинном обучении.
- ➔ **Примеры**: артикли, междометия, союзы, цифры и т.д.
- ➔ Не существует универсального списка стоп-слов.

Пример

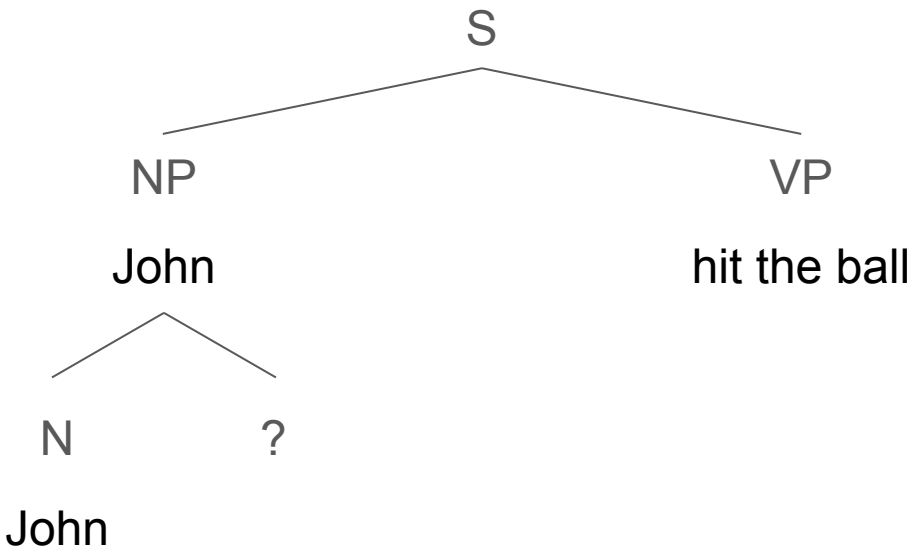
```
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))

sentence = "Backgammon is one of the oldest known board games."

word_tokens = nltk.word_tokenize(sentence)
filtered_sentence = [w for w in word_tokens if w not in stop_words]

print(word_tokens, filtered_sentence)
```

['Backgammon', 'is', 'one', 'of', 'the', 'oldest', 'known', 'board', 'games', '.']
['Backgammon', 'one', 'oldest', 'known', 'board', 'games', '.']

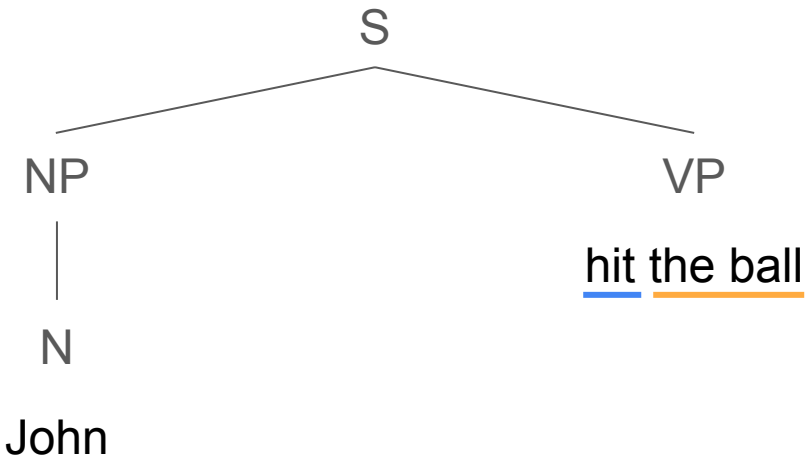
Дерево разбора (parse tree)



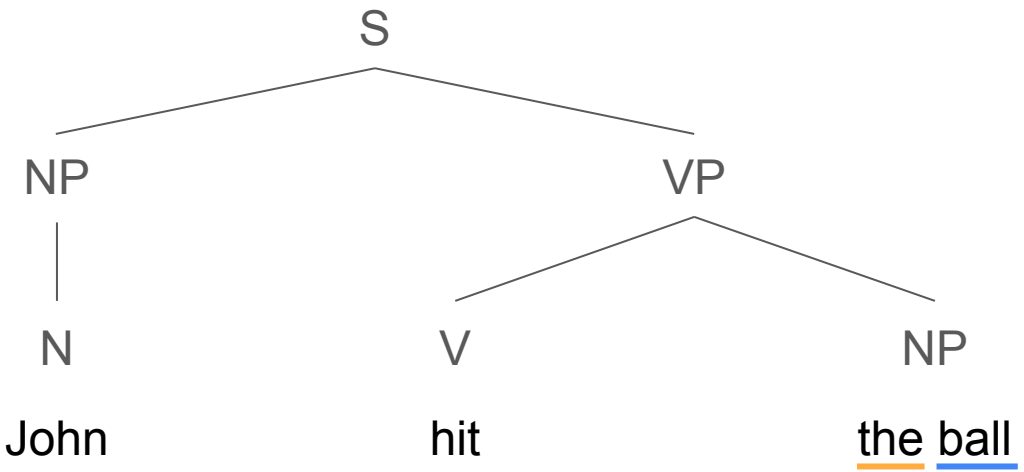
Дерево разбора (parse tree)



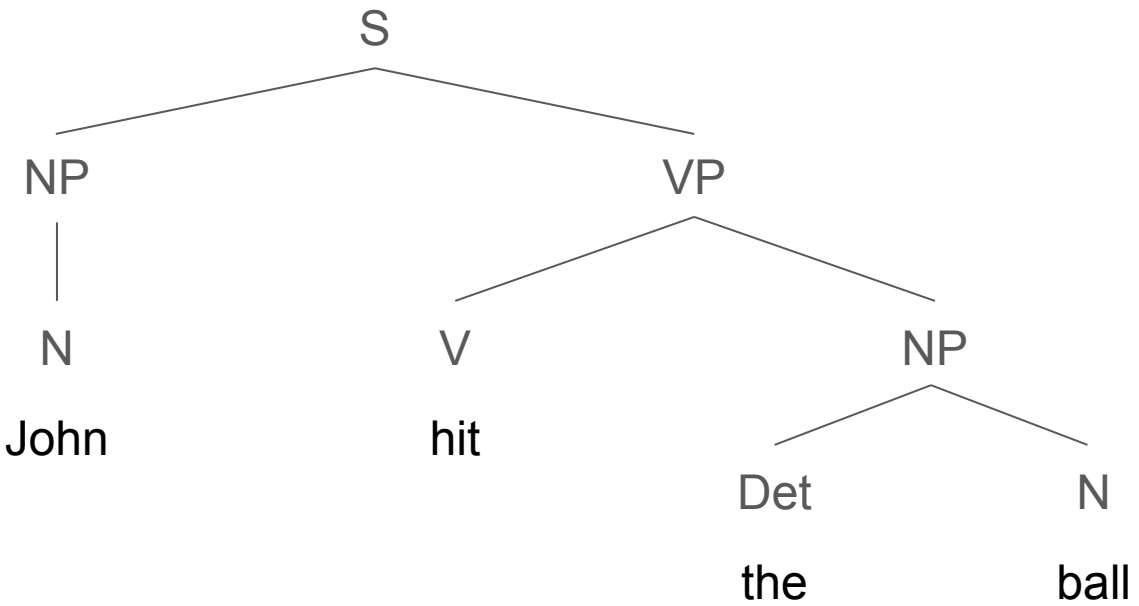
Дерево разбора (parse tree)



Дерево разбора (parse tree)



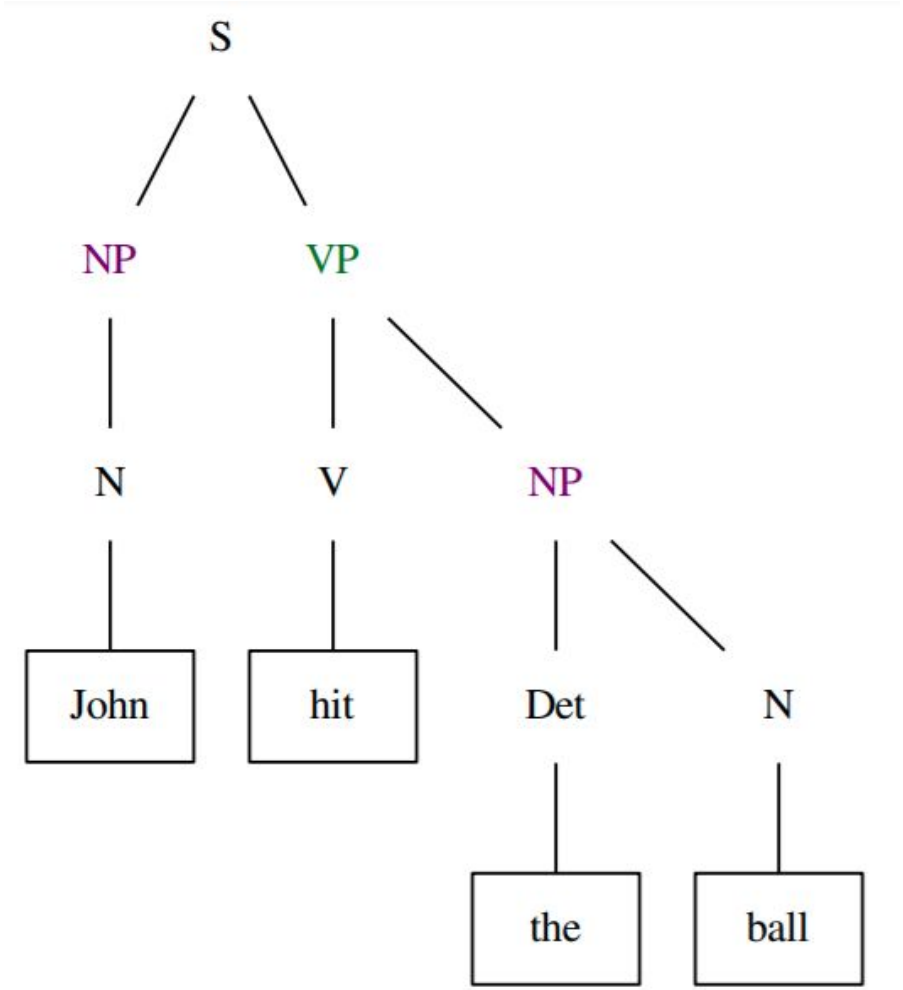
Дерево разбора (parse tree)



Пример

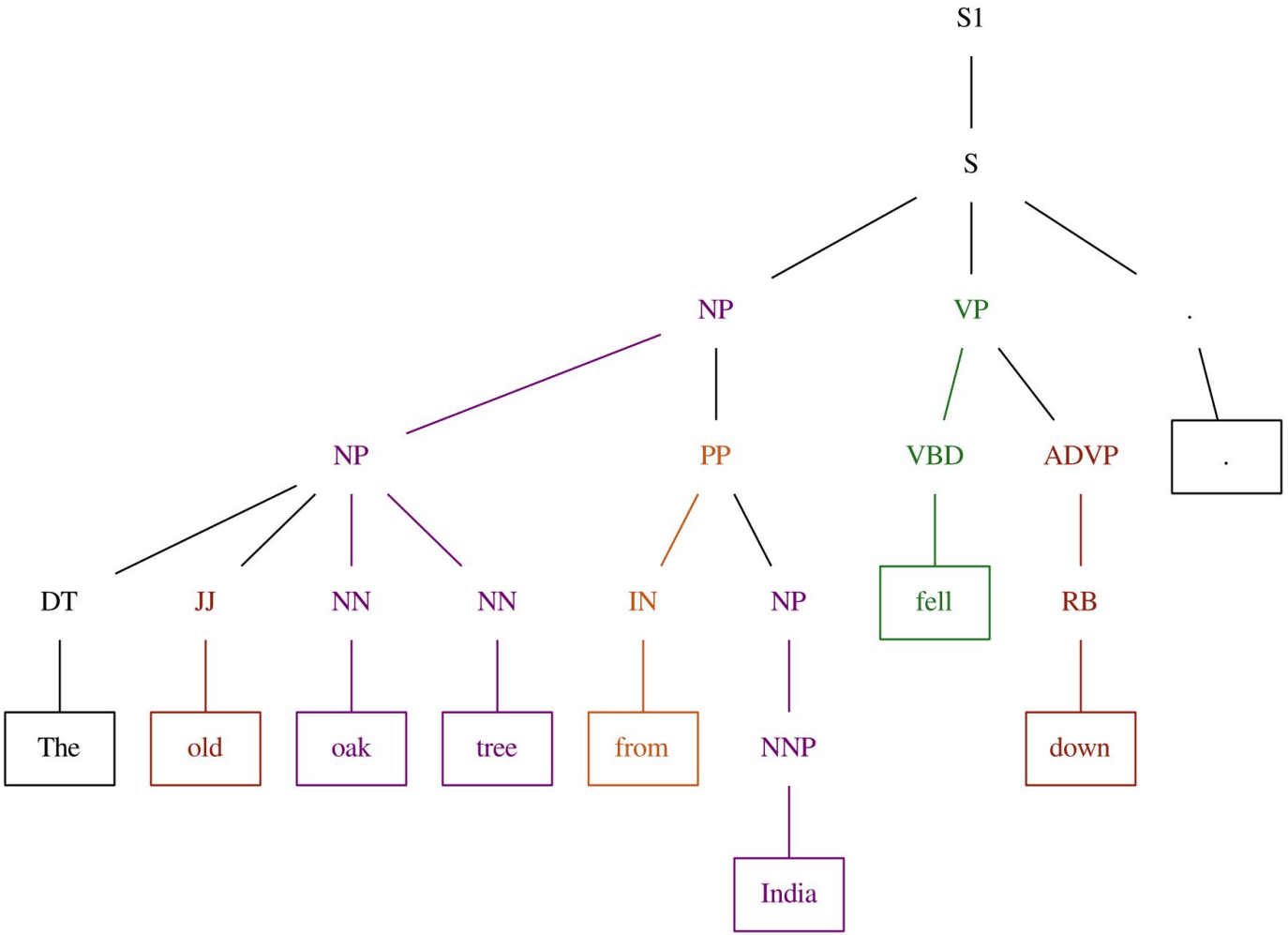
```
from nltk.parse import ChartParser

grammar = nltk.CFG.fromstring("""
   S -> NP VP
   NP -> N | Det N
   VP -> V NP
   N -> 'John' | 'ball'
   V -> 'hit'
   Det -> 'the'
""")

parser = ChartParser(grammar)
trees = parser.parse(nltk.word_tokenize("John hit the ball"))

for tree in trees:
   print(tree)
```

```
(S (NP (N John)) (VP (V hit) (NP (Det the) (N ball))))
```


S
NP
VP
N
V
NP
John
hit
Det
N
the
ball

Пример

```
from nltk.data import find
from nltk.parse import BllipParser

model_dir = find('models/bllip_wsj_no_aux')
parser = BllipParser.from_unified_model_dir(model_dir)

tokens = nltk.word_tokenize("The old oak tree from India fell down.")
trees = parser.parse(tokens)

print(trees.get_parser_best())
```


S1
S
NP
VP
.
NP
PP
VBD
ADVP
.
DT
JJ
NN
NN
IN
NP
fell
RB
The
old
oak
tree
from
NNP
down
India