Автоматический синтаксический анализ

Екатерина Владимировна Еникеева

2021

Автоматическая обработка естественного языка, лекция 6

Синтаксический анализ

- соотнесение входной строки (предложения) в заданной (или обученной по корпусу) грамматикой:

- распознавание (recognition) : да/нет – однозначный ответ
- собственно анализ (parsing) : дерево разбора / структура составляющих / последовательность продукций – возможны разные варианты

Оценка качества (составляющие)

Constituent-level precision / recall / F-score — аналогично IR

Верный ответ: совпадение индексов начала/конца составляющей и тега нетерминала

$$\textbf{labeled recall:} = \frac{\text{\# of correct constituents in hypothesis parse of } s}{\text{\# of correct constituents in reference parse of } s}$$

$$\textbf{labeled precision:} = \frac{\text{\# of correct constituents in hypothesis parse of } s}{\text{\# of total constituents in hypothesis parse of } s}$$

Алгоритмы парсинга

- **top-down parsing** – нисходящие алгоритмы разбора (например, *Earley parser*)
- **bottom-up parsing** – восходящие алгоритмы разбора (например, *CYK parser*)

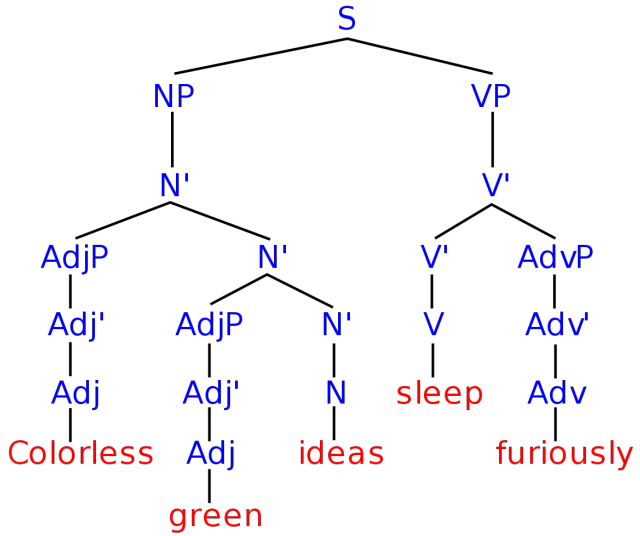
$S \longrightarrow \langle NP \rangle \langle VP \rangle$

$NP \longrightarrow N'$

$VP \longrightarrow V'$

$N' \longrightarrow \langle AdjP \rangle \langle N' \rangle$

…



CYK парсер

Алгоритм Кока-Янгера-Касами / CYK / CKY

S			
NP		VP	
Det	N	V	Adv
the	**cat**	**sleeps**	**quitely**

$$S \rightarrow \langle NP \rangle \langle VP \rangle$$

$$NP \rightarrow Det \langle NP \rangle$$

$$VP \rightarrow \langle VP \rangle Adv$$

$$NP \rightarrow N$$

$$VP \rightarrow V$$

$$N \rightarrow cat$$

$$Det \rightarrow the$$

$$V \rightarrow sleeps$$

$$Adv \rightarrow quitely$$

Парсер Эрли

Earley Parser

Итеративно «*распознает*» правила, храня таблицу соответствующих *состояний* (*dotted rules*):

$S \rightarrow \cdot\, VP$, [0,0] << 0 позиция в списке входных токенов

$NP \rightarrow Det \;\cdot\; Nominal$, [1,2] << NP начинается с 1 токена, точка в позиции 2

$VP \rightarrow Verb\ NP \cdot$, [0,3] << конец парсинга

Парсер Эрли

Процедуры на шаге *k*:

- *Prediction*: раскрываем нетерминалы справа от точки, добавляя новые правила в таблицу

Из $S \rightarrow \cdot\, VP$, [0,0] добавляем $VP \rightarrow \cdots$

- *Scanning*: сопоставляем POS-нетерминалы справа от точки входным токенам; сдвигаем точку, если нашли совпадение

Если есть $Verb \rightarrow book$, то из $VP \rightarrow \cdot\, Verb\ NP$, [0,0] добавляем $VP \rightarrow Verb \cdot NP$, [0,1]

- *Completion*: если точка оказалась в конце правила, ищем по предыдущим состояниям

$NP \rightarrow Det\ Nominal\ \cdot$, [1,3] + $VP \rightarrow Verb \cdot NP$, [0,1] = успех

Парсер Эрли

Подробный пример разбора можно найти в учебнике Jurafsky+Martin: глава 13 в изд. 2

PCFG

Probabilistic Context Free Grammar

- каждое правило сопровождается весом (вероятностью)
- сумма всех вероятностей расширений нетерминалов = 1
- консистентная PCFG – сумма вероятностей всех предложений языка = 1

Вероятностный парсинг

Вероятность разбора T, состоящего из n правил вида $LHS \rightarrow RHS$, для предложения S:

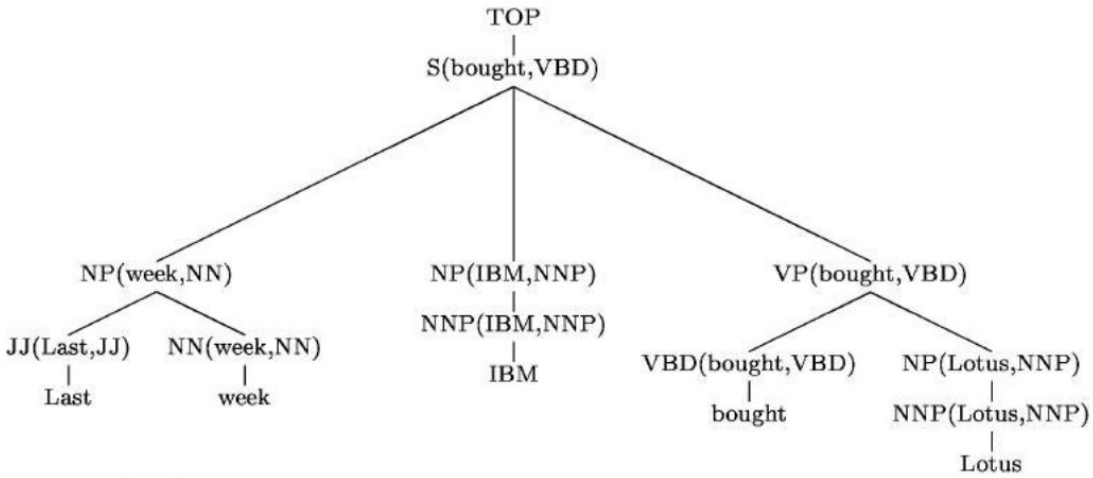
$$P(T,S) = \prod_{i=1}^{n} P(RHS_i \mid LHS_i)$$

Можно использовать вариацию CYK –

probabilistic CYK

Оценка вероятностей – по корпусу

Lexicalized parsers



Internal Rules:

TOP	→	S(bought,VBD)		
S(bought,VBD)	→	NP(week,NN)	NP(IBM,NNP)	VP(bought,VBD)
NP(week,NN)	→	JJ(Last,JJ)	NN(week,NN)	
NP(IBM,NNP)	→	NNP(IBM,NNP)		
VP(bought,VBD)	→	VBD(bought,VBD)	NP(Lotus,NNP)	
NP(Lotus,NNP)	→	NNP(Lotus,NNP)		

Lexical Rules:

JJ(Last,JJ)	→	Last
NN(week,NN)	→	week
NNP(IBM,NNP)	→	IBM
VBD(bought,VBD)	→	bought
NNP(Lotus,NN)	→	Lotus

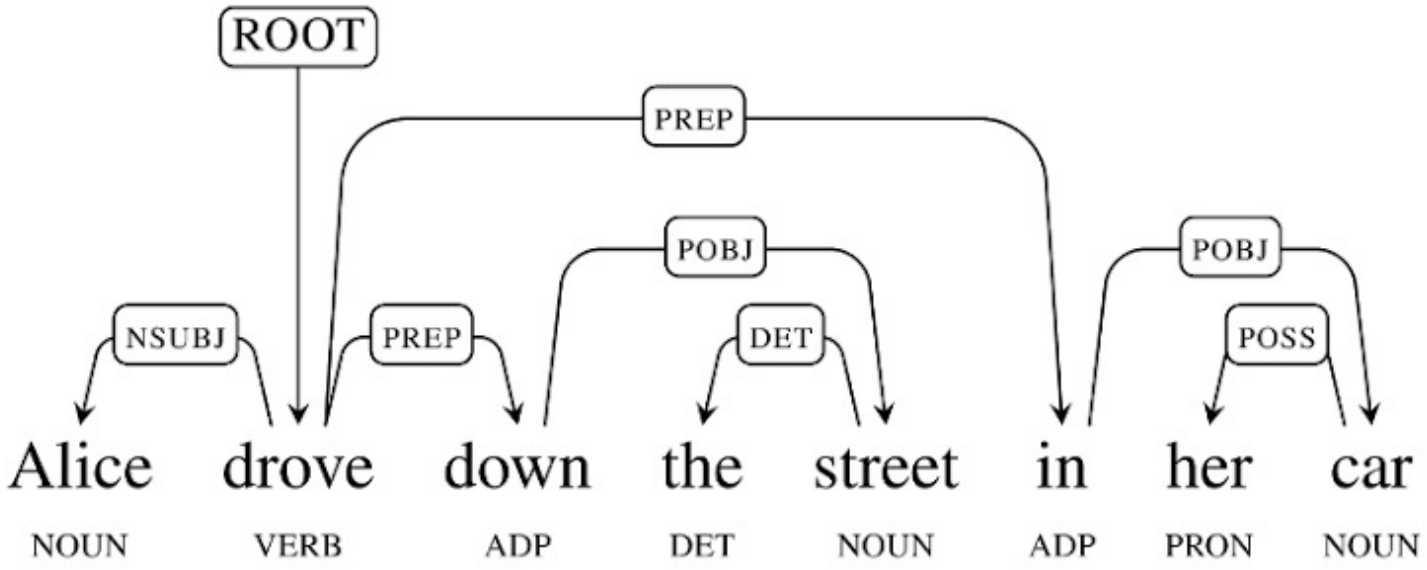
Вероятностный парсинг

- Collins Parser — *Collins M. (2000). Head-driven Statistical Models for Natural Language Parsing. Computational Linguistics, 29(4).*

http://www.cs.columbia.edu/~mcollins/code.html

- Charniak Parser — *Charniak E. (1997). Statistical Parsing with a Context-Free Grammar and Word Statistics. AAAI-97.*

Структура зависимостей



Структура зависимостей

Элементы:

- зависимости (ребро, edge) : типы зависимостей
- узлы / вершины (node / vertex)
  - вершины / главные слова / хозяева
  - зависимые / подчинённые / слуги

> дерево – граф, у которого есть корневой узел (корень)

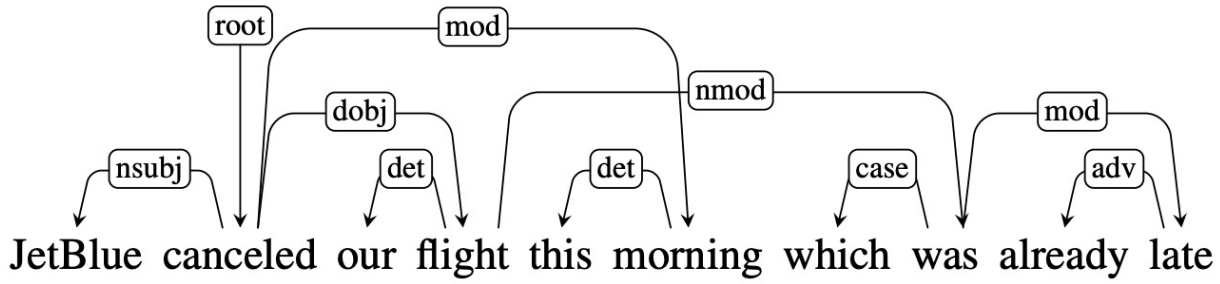
Требования

- Существует единственный корень
- В каждую вершину входит только одна стрелка
- Для каждой вершины существует единственный путь до неё от корня

Структура зависимостей

Важное свойство деревьев зависимостей – **проективность**

- никакая пара стрелок не пересекается (принцип непересечения стрелок)
- никакая стрелка не накрывает корневой узел (принцип обрамления стрелок)



Описание зависимостей

- Могут быть получены из структуры составляющих с использованием алгоритмов head-finding:
  - задаются в КС-грамматике или по правилам
- Упорядоченные пары (head, dep)
  (flight, morning) (<root>, book)
- Пары с указанием направления стрелки:
  rightarc, leftarc
- Тройки (head, dep, relation)

Оценка качества (зависимости)

- *Unlabelled Attachment Score* (*UAS*) = доля верно приписанных вершин
- *Labelled Attachment Score* (*LAS*) = доля верно приписанных вершин + размеченных отношений (accuracy по паре тегов)
- *Morphology-Aware Labeled Attachment Score* (*MLAS*)
- *Bilexical dependency score* (*BLEX*)

См. CoNLL Evaluation

Парсер зависимостей

простой подход для языков программирования — shift-reduce parser

- грамматика
- стек (stack)
- список входных токенов

➢ для первых 2 элементов стека находим соответствие в грамматике и т.д.

Transition-based parser

Состоит из

- грамматики (oracle – «чёрный ящик»)
- «конфигурации парсера»
  - стек (stack)
  - список входных токенов
  - список отношений, составляющих дерево зависимостей

Парсер зависимостей

Операции переходов (transition operators)

- LeftArc – первое слово – главное, второе – зависимое, убираем второе из стека
- RightArc – второе слово – главное, первое – зависимое, убираем первое из стека
- Shift – берем очередное слово из входных токенов и кладём в стек

Arc standard approach

Ограничения:

- Анализируются только токены (обычно 2) в верху стека
- Когда у токена находится вершина, он удаляется из стека

Пример

Разбор предложения «*Book me the morning flight*»

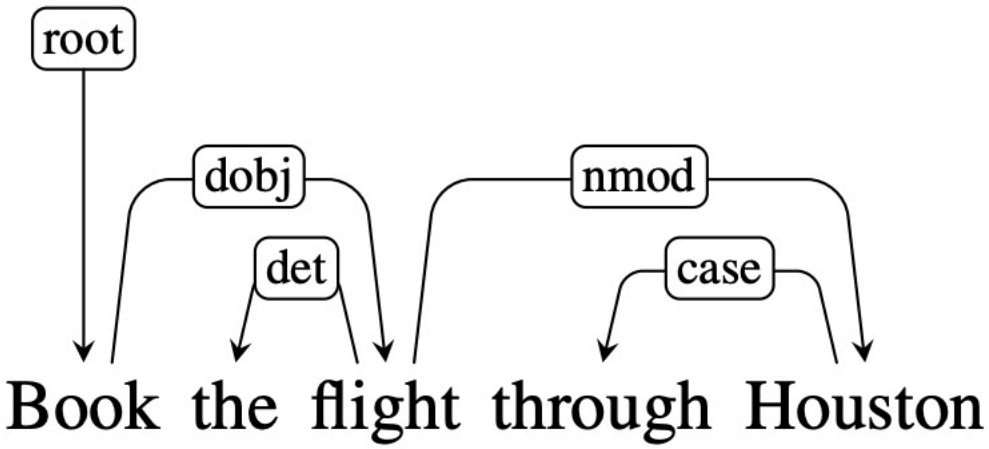
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

Arc eager approach

Операции переходов (transition operators)

- LeftArc – входное слово – главное, верх стека – зависимое, убираем верхнее из стека
- RightArc – верх стека – главное, входное слово – зависимое, убираем верхнее из стека
- Shift – берем очередное слово из входных токенов и кладём в стек
- Reduce – удаляем верхнее из стека

Пример



Обучение парсеров

- результат LeftArc / RightArc получается с помощью предсказателя (Oracle) – выбор подходящего отношения зависимости и главного слова
- это как раз и можно обучать!

Nivre J. (2009). Non-projective Dependency Parsing in Expected Linear Time. ACL IJCNLP 2009.

Инструменты

- NLTK
- UD Pipe (1 / 2)
- Stanford CoreNLP / stanza
- MaltParser
- spaCy
- DeepPavlov

Практика

https://colab.research.google.com/drive/1GMDA17iP_-SiJq8_MMQTT20VrWPC3blE?usp=sharing

Применение

разрешение анафоры

- *Иван хочет, чтобы Петр пригласил его к себе.*
- *Джейн сказала Биллу, что она идёт в университет.*
- *Он рассказал им об Иване.*