Автоматический синтаксический анализ

(часть 2)

Екатерина Владимировна Еникеева

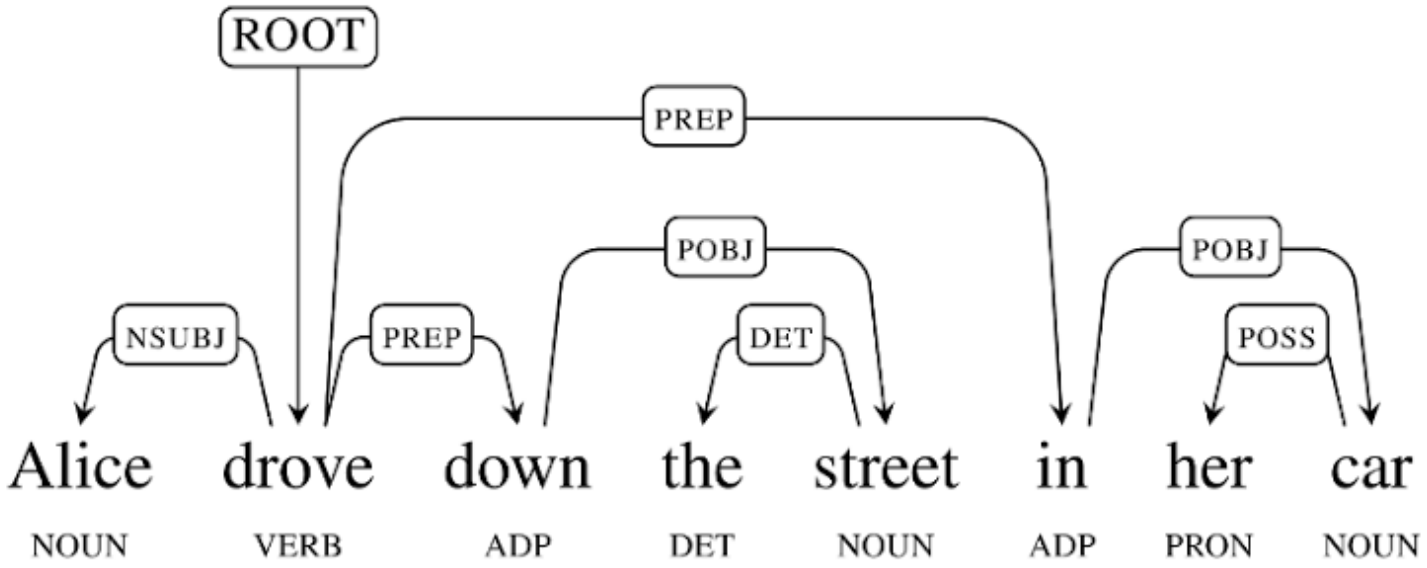
9 октября 2023

Автоматическая обработка естественного языка, лекция 6

План

1. Включить запись!
2. Анонсы
3. Дерево зависимостей
4. Оценка качества в терминах зависимостей
5. Парсинг зависимостей
6. ML-based подходы

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)

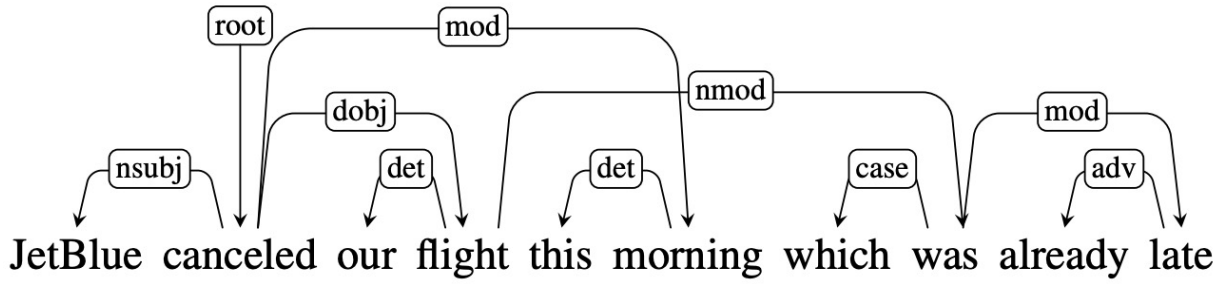
Требования

- 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)



Структура зависимостей

Проблемы:

- принцип единственности вершины
  - *Мы оставили комнату закрытой.*
- сочинительные конструкции
  - *Прошли день и ночь.*
  - *необходимые условия и результаты*
- иерархия синтаксических единиц

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

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.

Как представить данные

- Парсер – конфигурация на каждом шаге
  стек S, список отношений R_c
- Корпус (treebank) – деревья зависимостей
  множества вершин V и отношений R_p

Как получить из parse tree список конфигураций?

- симулируем парсинг при условии готового дерева

  $LEFTARC(r)$: $(S_1\ r\ S_2) \in R_p$

  $RIGHTARC(r)$: $(S_2\ r\ S_1) \in Rp$ и $\forall r', w$ таких что $(S_1\ r' w) \in R_p$
  $(S_1\ r'\ w) \in R_c$

  $SHIFT$: во всех остальных случаях

Задача

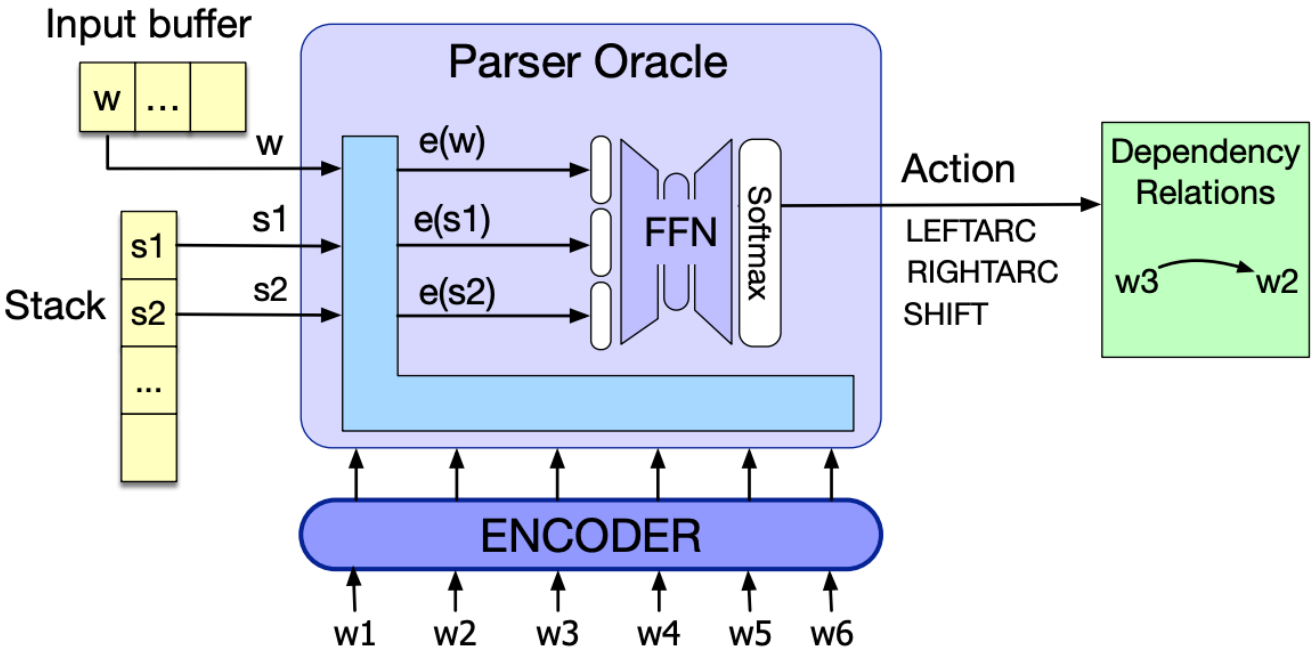
Stack	Word buffer	Relations
[root, canceled, flights]	[to Houston]	(canceled → United) (flights → morning) (flights → the)

SHIFT

Features: classic

- Базовые признаки:
  - токен, лемма, POS-тег …
- Объекты, для которых можем извлечь фичи:
  - слова стека s_1, s_2, …
  - слова входного буфера b_1 …
- Feature templates:
  - $< s_1.w, op >$
  - $< b_1.t, op >$
  - $< s_1.t + s_2.t, op >$

Features: embeddings



Улучшения

- Arc eager approach
  - Добавляет операцию *Reduce*
- Beam search:
  - На каждом шаге разрешаем несколько вариантов
  - Параметр – beam width N
  - Кладём лучшие разборы в список – не должен быть длиннее N
  - Лучшие по какому признаку?
    для конфигурации c и операции t:

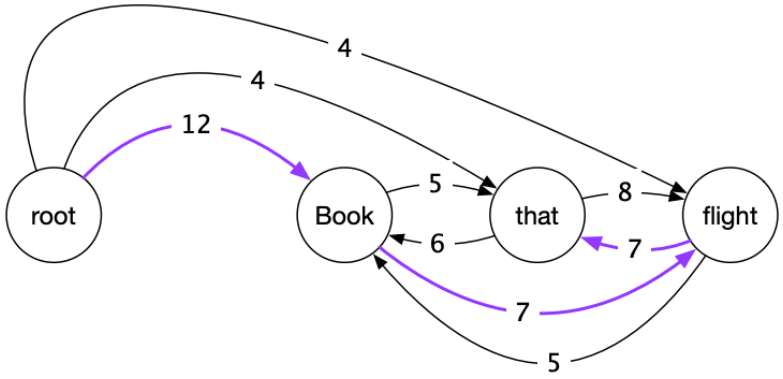
$$ScoreC(c_i) = ScoreC(c_{i-1}) + Score(t_i, c_i)$$

Graph-based parsing

1. Оценка отношений / edge scoring
   приписываем каждой паре токенов вес, используя вероятностный классификатор

2. (optional) Label scoring

3. Выбор дерева
   алгоритм **maximum spanning tree**
   - покрывает все вершины
   - начинается в root
   - имеет максимальный вес



Инструменты

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

Примеры

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