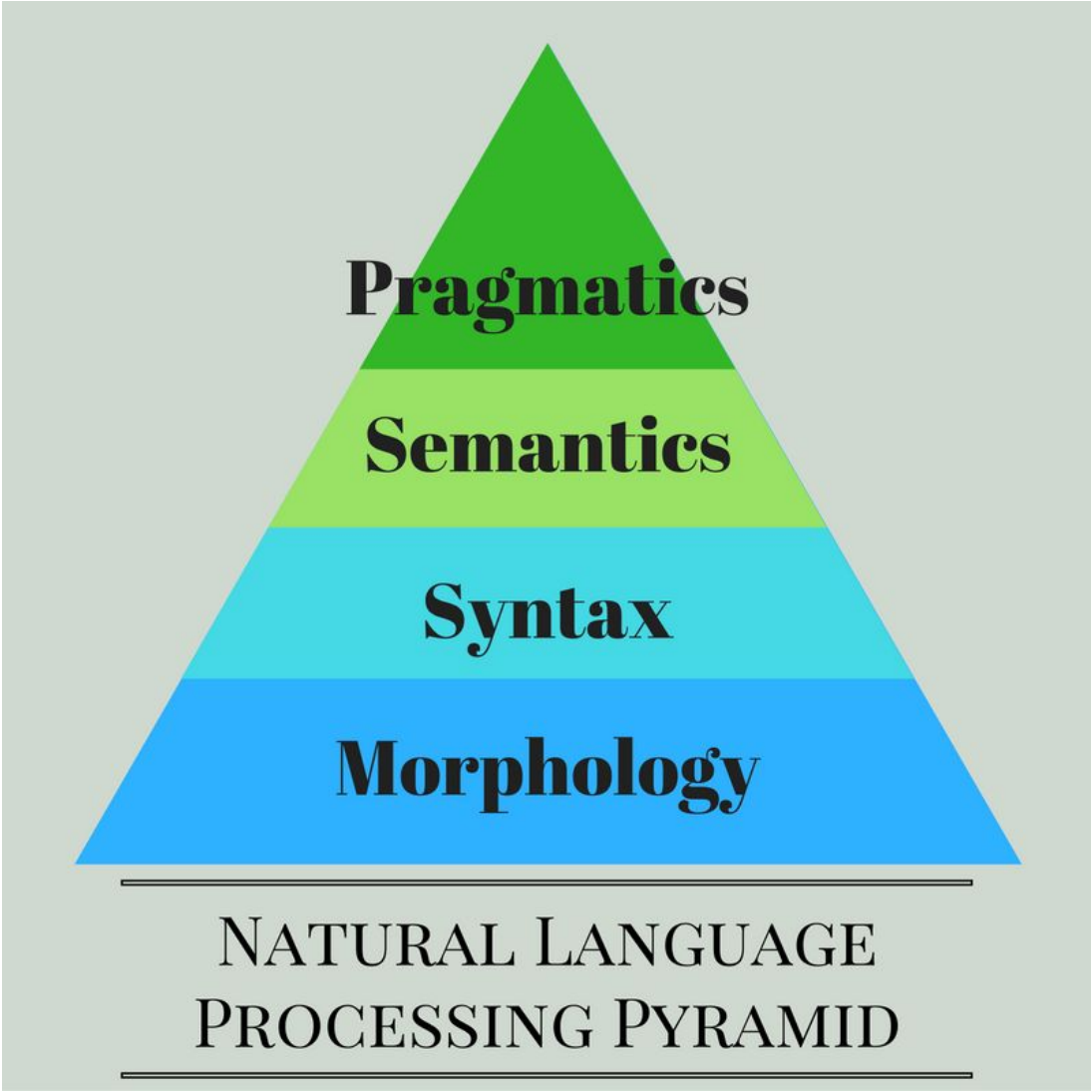
Синтаксический анализ и его применение

Попов Артём

Математические методы анализа текстов
осень 2019

Введение

Вспомним вводное занятие...



Пирамида NLP: с чем уже познакомились

Морфология — слова

- Работа с OOV словами

Синтаксис — фразы/предложения

- Определение частей речи

Семантика — предложения (возможно, с контекстом)

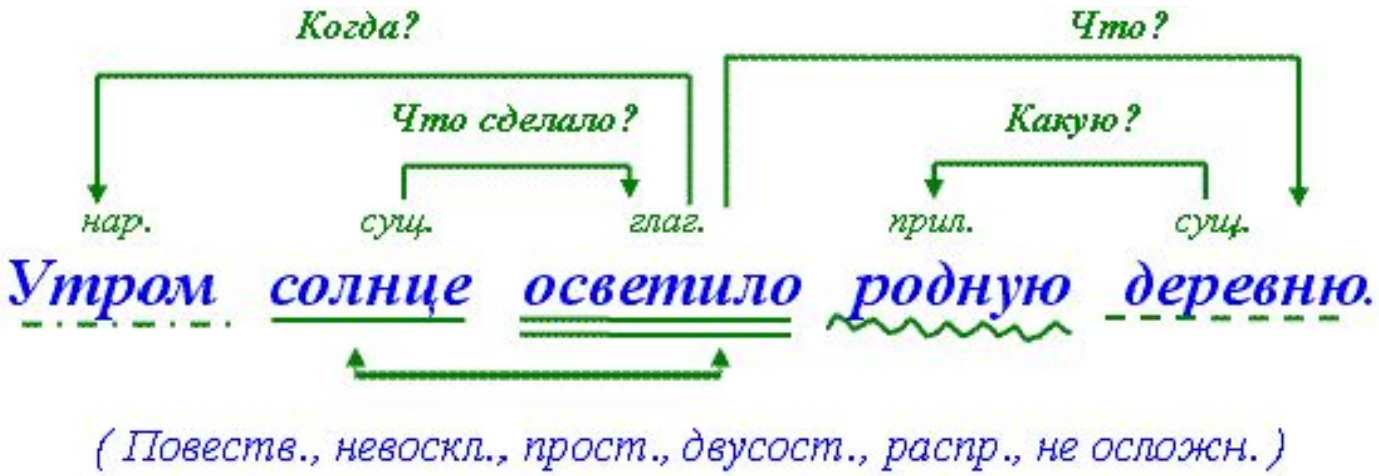
- Распознавание именованных сущностей

Прагматика — текст

- Суммаризация, тематическая сегментация

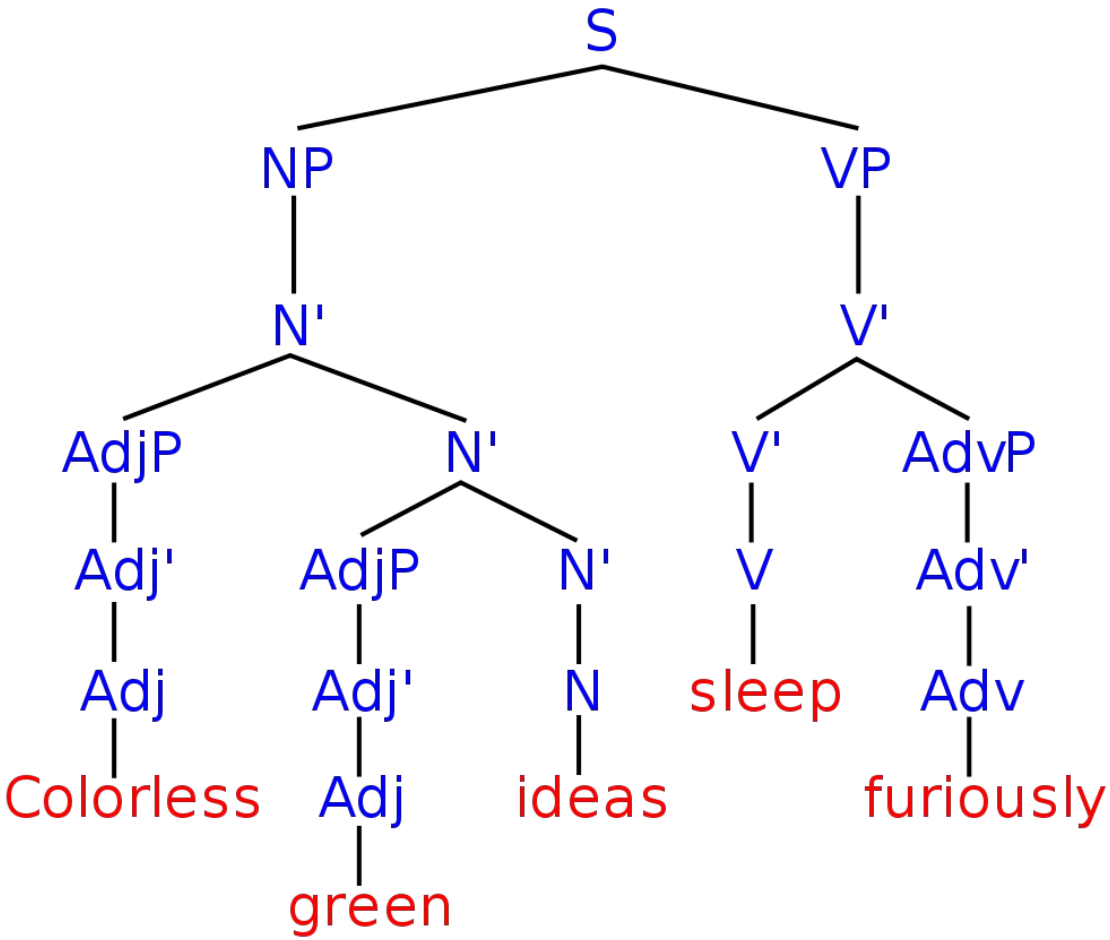
Пример синтаксического разбора

Синтаксический разбор — анализ структуры предложения



Находим не только характеристики слов как в задаче определения частей речи, а зависимости между словами

Грамматически правильное предложение может быть бессмысленным



Зачем это может быть нужно?

- Определение парафразов

  «Карта заблокирована» vs «заблокируйте карту»

- Проверка качества при генерации языка
- Аугментация данных (перестановка/удаление слов)
- Анализ тональности отдельных слов в предложении
- Выделение фактов (fact extraction, information extraction)

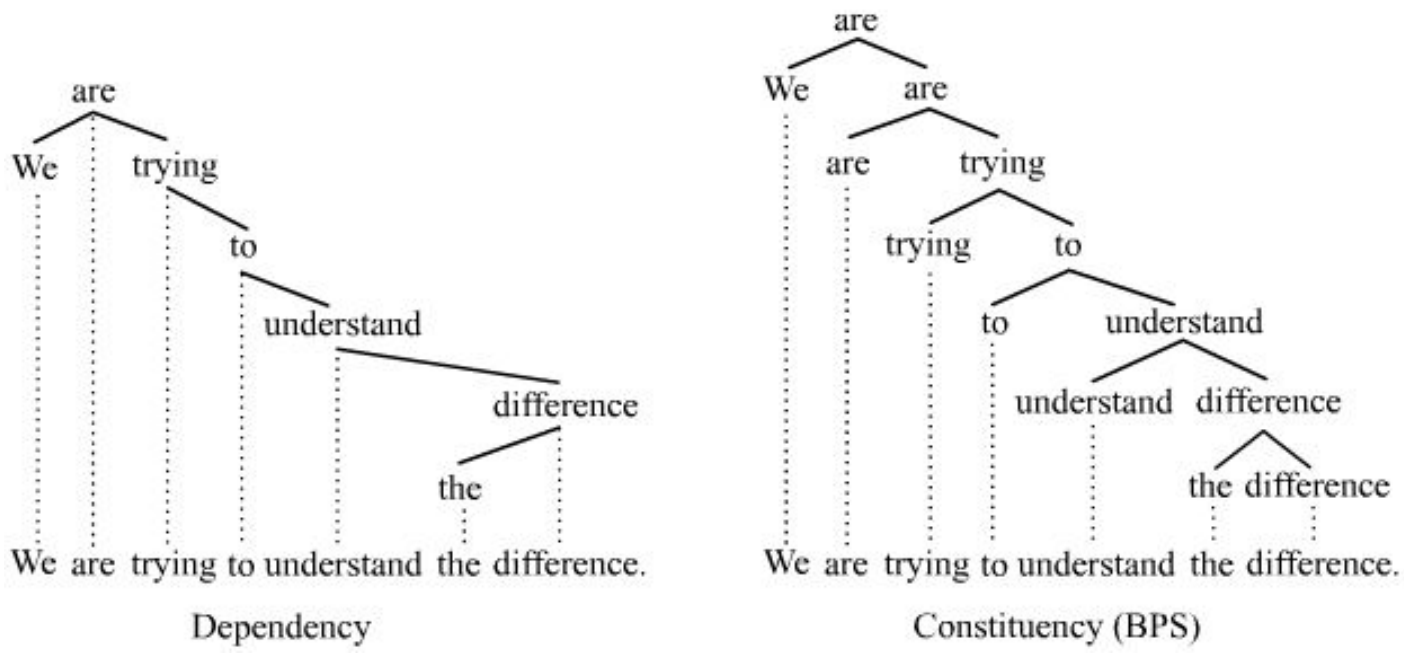
  НанеслиВизит(кто, кому, дата)

- Использование весов для слов в различных задачах

Модели построения разбора

Модели построения разбора

1. Грамматика составляющих (constituency, phrases)
2. Грамматика зависимостей (dependency)



Формальное определение: составляющие

S — линейно упорядоченное множество слов.

Система составляющих на **S** — множество **C** отрезков **S**.

C содержит **S** и каждое слово, входящее в **S**.

Любые два отрезка, входящие в **C**, либо не пересекаются, либо один из них содержится в другом.

Элементы множества **C** — составляющие.

Пример разбора: составляющие

S — исходное предложение

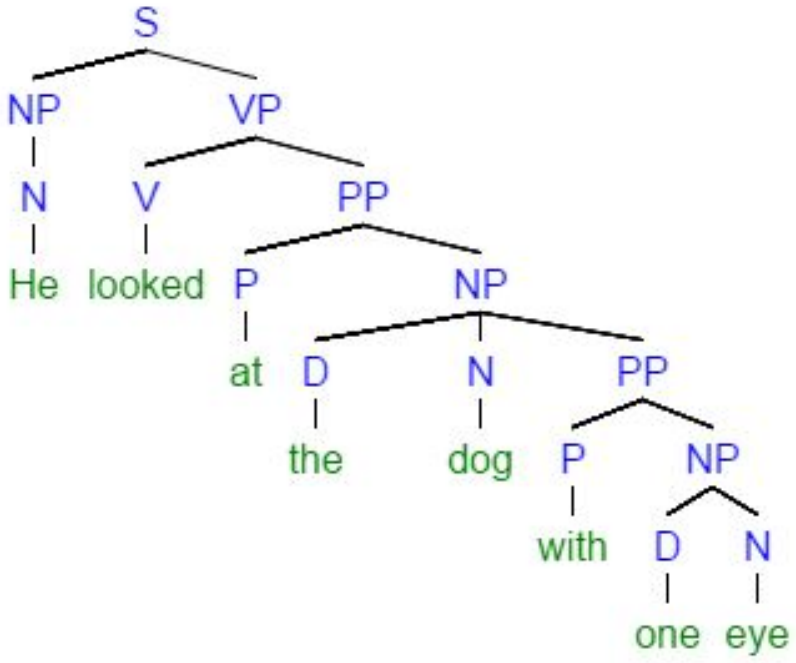
VP — глагольная группа, verb phrase (глагол + зависимые)

NP — именная группа, noun phrase (существительное)

PP — предложная группа, prepositional phrase

AP — группа прилагательного, adjective phrase

D (Det) — детерминативы (артикли, указательные и т.п.)



Использование СР на практике

- Составляющие можно перемещать в рамках предложений

  John talked [to the children] [about rules].

  John talked [about rules] [to the children].

  **John talked rules to the children about.*

- Составляющие можно заменять на похожие

  I sat [on the box / on top of the box / in front of you].

Резюме: составляющие

- Подход популярен в лингвистике
- Лучше описан в “учебной” литературе
- Плохо применим для языков, в которых может быть произвольный порядок слов (привет, русский язык)
- Часто описывается контекстно-свободными языками
- Для построения разбора может использоваться алгоритм CYK (Cocke-Younger-Kasami)

Jurafsky and Martin, Speech and Language Processing (три главы)

лекция про CP MMAT 2018 (ВМК)

Формальное определение: зависимости

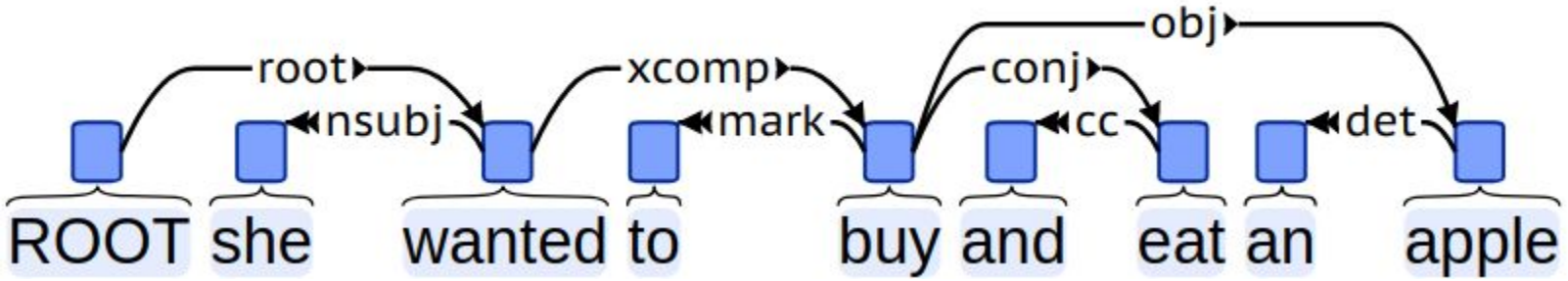
Дерево зависимостей — направленный граф, т.ч.:

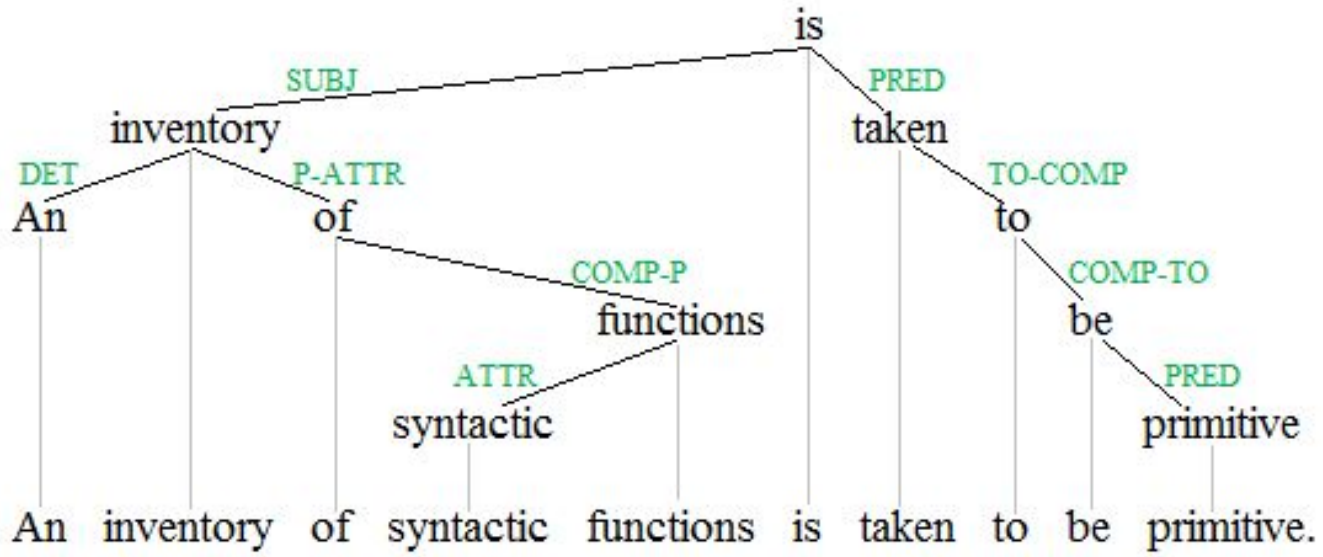
- граф является деревом
- вершины графа — слова и [ROOT]
- в каждую вершину, кроме [ROOT], входит одно ребро
- в [ROOT] не входит ни одно ребро

Рёбра дерева зависимостей описывают зависимость одного слова от другого.

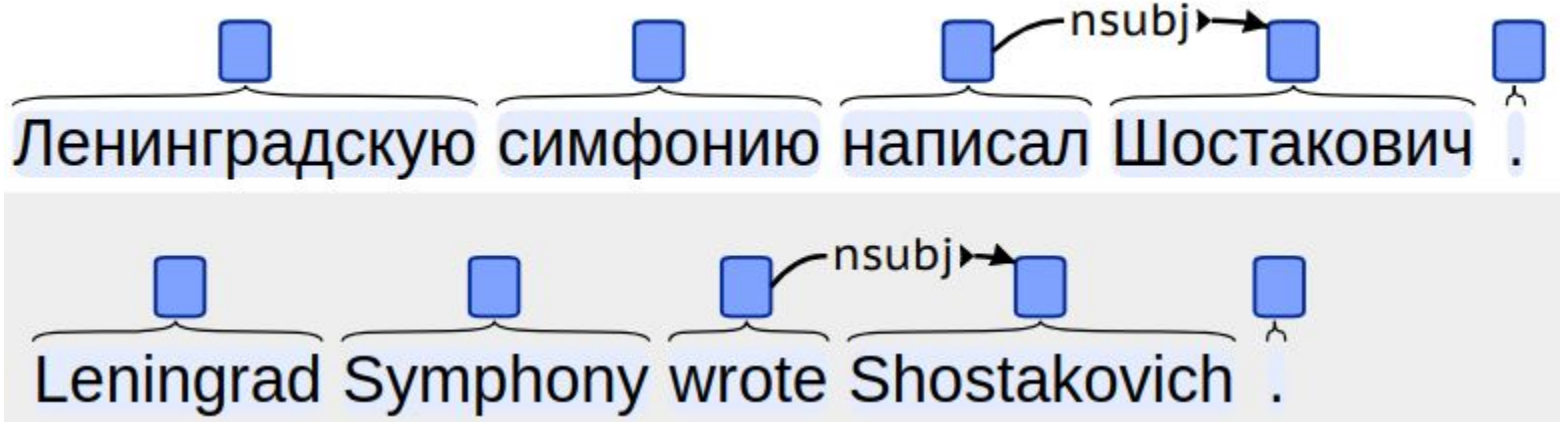
Рёбра могут иметь “тип” связи.

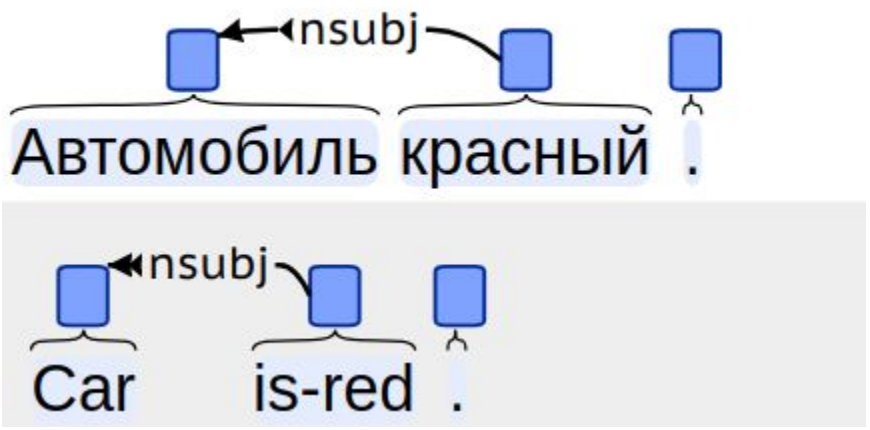
Пример разбора: зависимости



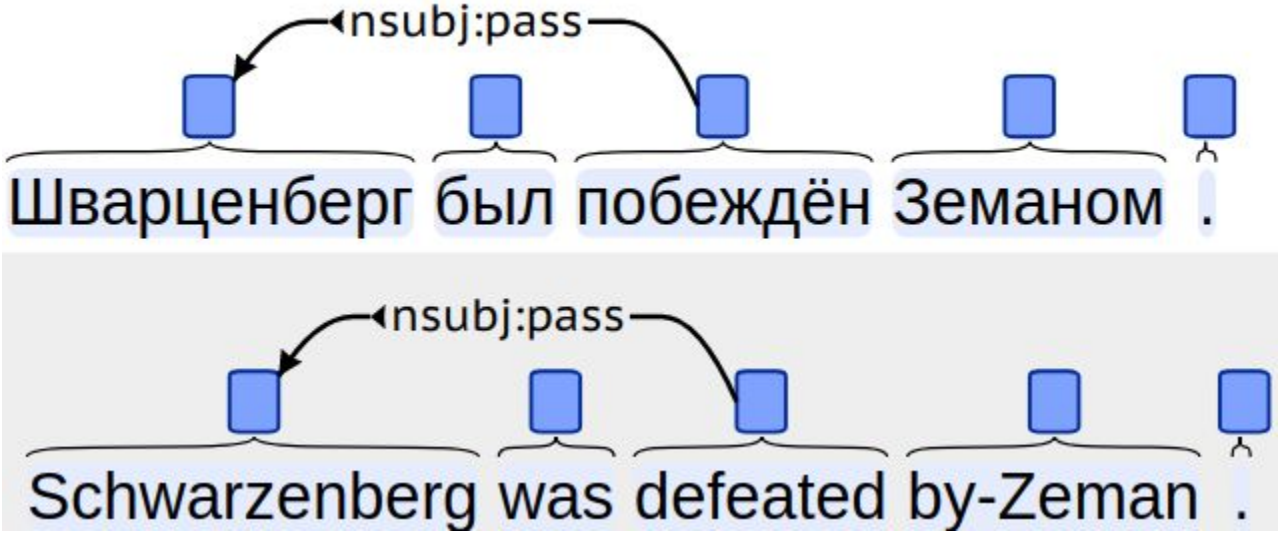


Пример связей: nsubj

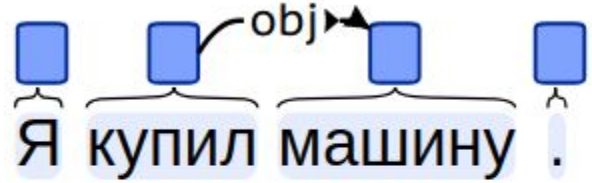


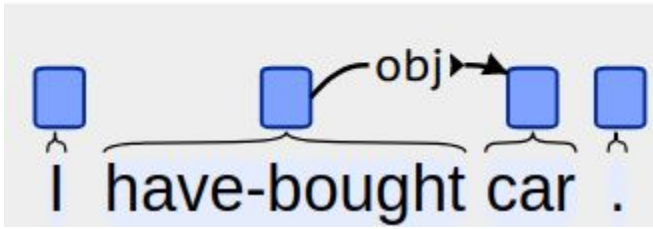


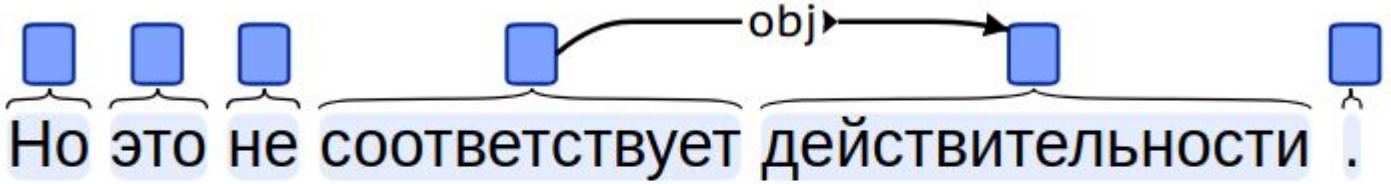
Пример связей: nsubj:pass

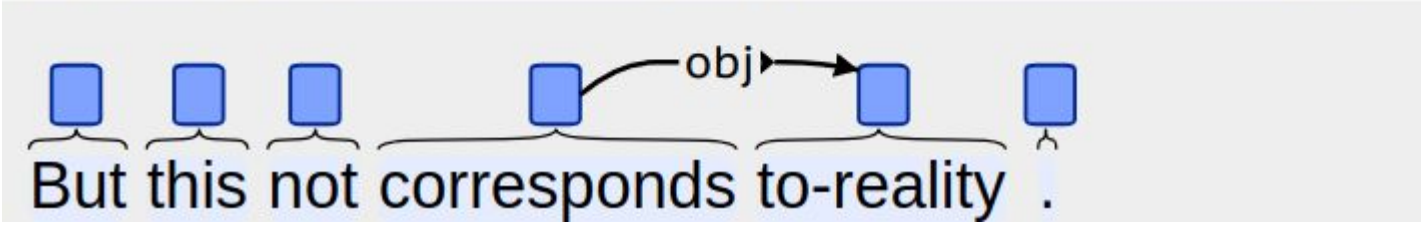


Пример связей: obj

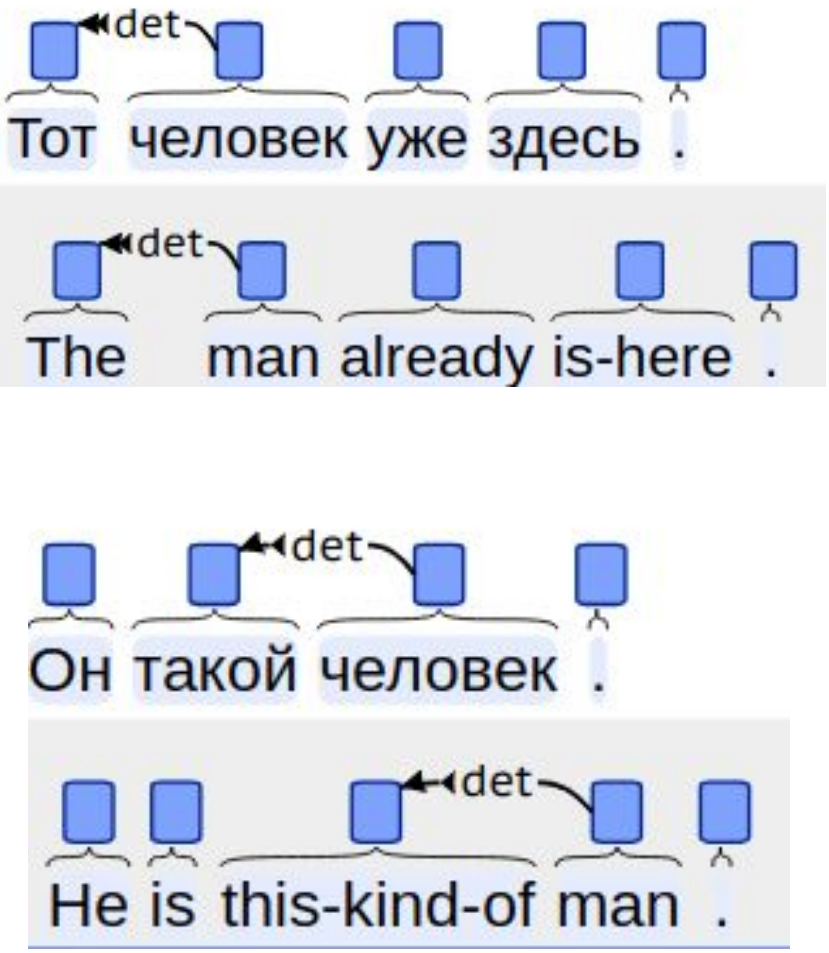








Пример связи: det

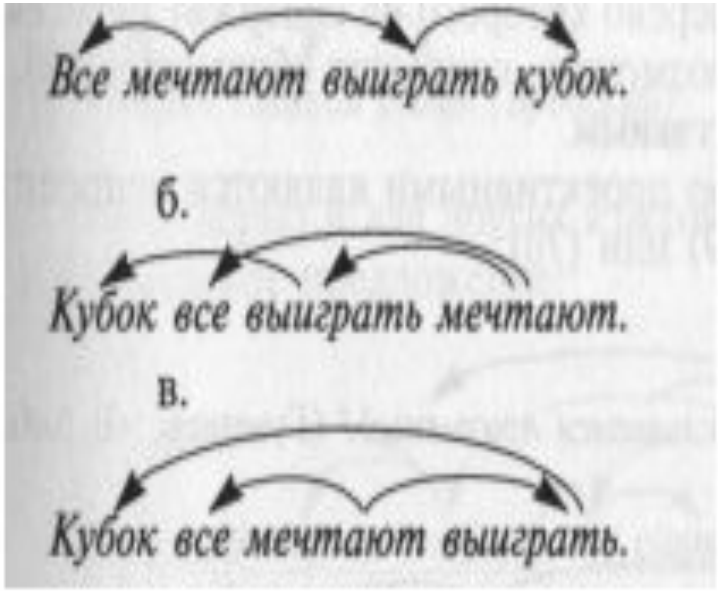


Проективность

Предложение называется проективным, если:

1. Ни одна из стрелок не пересекает другую стрелку;

2. Никакая стрелка не накрывает корневую (~ сказуемое -> подлежащее)



Проективное

непроективное — нарушен принцип пересечения

непроективное — нарушен принцип обрамления

Phrases vs Dependencies

- Хорошо разработанные в лингвистике теории;
- Отчасти (!) формально (!) взаимозаменяемые
- Нет главной и нет вторичной
- Есть проблемы на периферии у обеих

Он сам увидел их семью

Он увидел их при помощи своих семи глаз

Эти типы стали есть в цехе

Построение дерева зависимостей

Данные для обучения

Хотим обучить алгоритм генерирующий по предложению его синтаксический разбор.

Для того, чтобы обучить алгоритм, нам нужна размеченная выборка: предложения и их разбор.

Удивительно, но для большинства языков такие выборки (treebanks) несложно найти: treebanks для разных языков.

Проект Universal dependencies

Лингвистическая проблема: несоответствие терминов и правил из грамматик зависимостей разных языков.

Data Science проблема: обучить синтаксический парсер для многих языков.

Решение: http://universaldependencies.org/

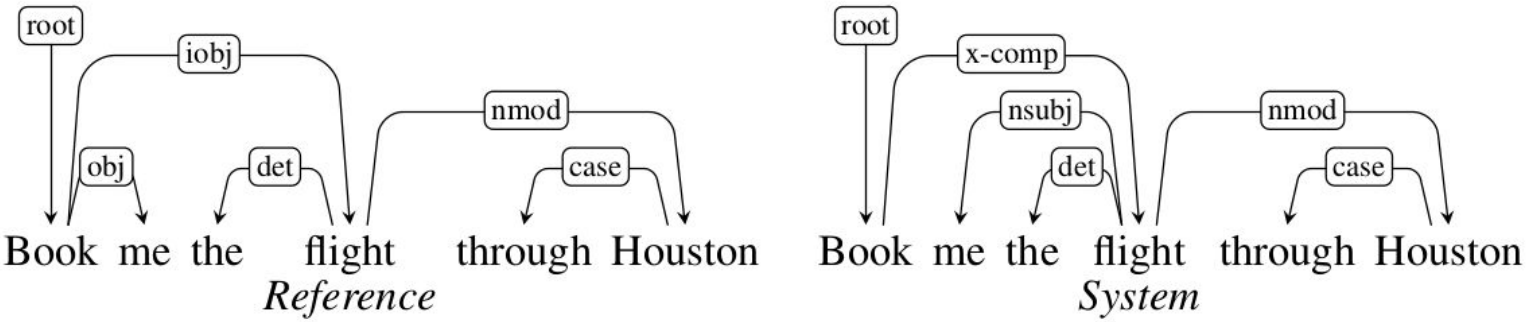
- 100 корпусов для 60 языков, все теги зависимостей унифицированы.

Метрики качества для построенного дерева

Unlabeled Attachment Score (UAS) — доля правильно угаданных рёбер

Labeled Attachment Score (LAS) — доля правильно угаданных рёбер с правильным типом метки

Можно оценивать число правильных полных разборов по выборке или усреднять UAS/LAS



Подходы построения дерева зависимостей

Вход: предложение

1. Transition-based — жадный способ построения дерева
2. Graph-based — полный поиск по всем возможным деревьям

Основная проблема: построить дерево

Предсказать метки по построенному дереву проще

(классификатор на каждую пару вершин по их признакам)

Graph-based подход

Идея: для каждой пары слов **h, m** в предложении **s** оцениваем, нужно ли их добавить в дерево

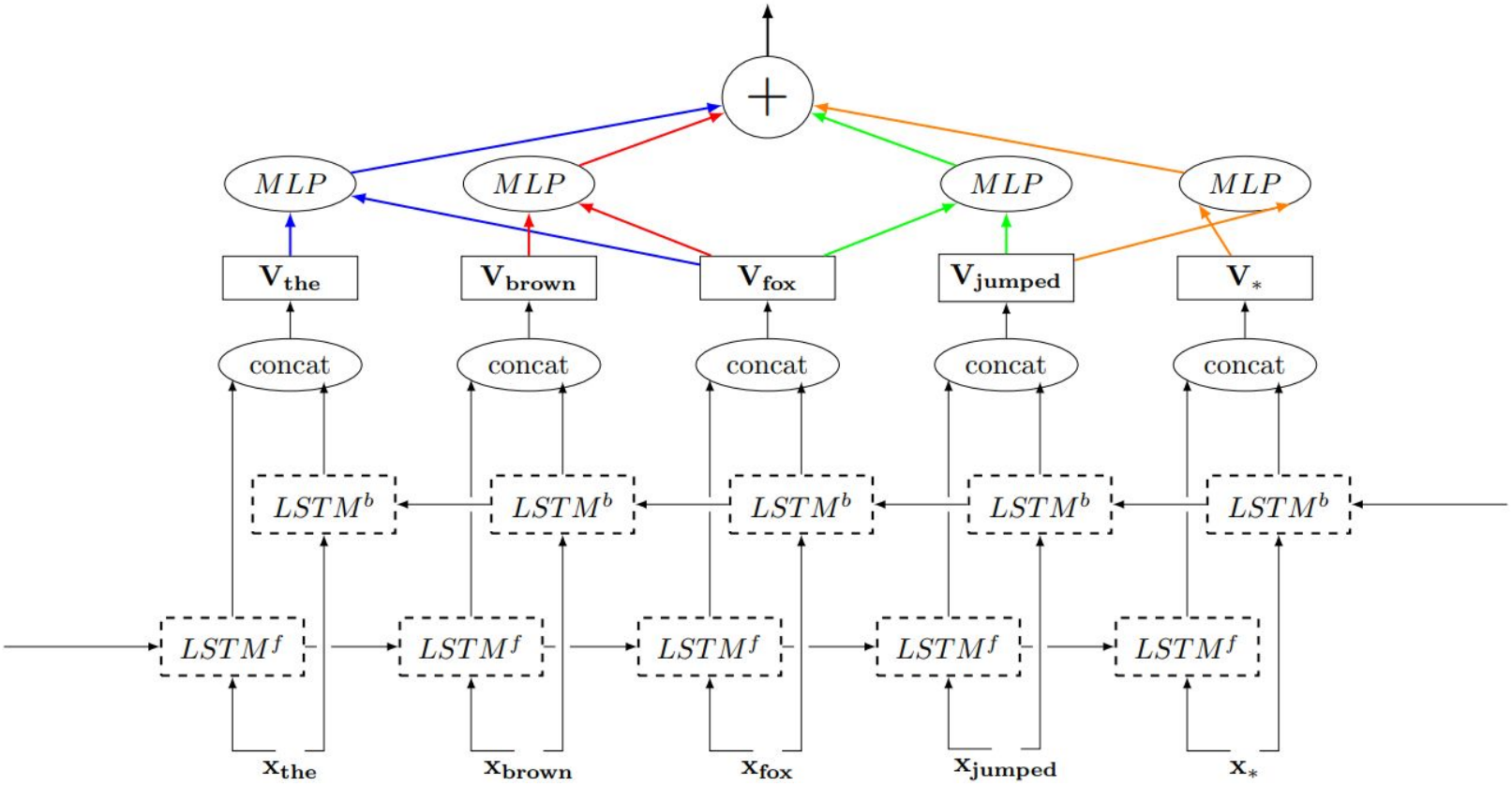
Например так:

v = BiLSTM(s)

score = MLP(concat(v_h, v_w))

Функция потерь для **s** и правильного дерева **y**:

$$max\Big(0, 1 - \max_{y' \neq y} \sum_{(h,m) \in y'} MLP(v_h \circ v_m) + \sum_{(h,m) \in y} MLP(v_h \circ v_m)\Big)$$



Kiperwasser et al (2016); Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations

Особенности graph-based подхода

- лучше transition-based (особенно на длинных предложениях)
- гораздо медленнее transition-based на этапе применения

Этап применения:

1. Посчитать оценки всех пар вершин
2. Выбрать максимальное остовное дерево
3. Если учитывать проективность, то (2) сложнее...

Transition-based подход: сущности

Пусть у нас есть:

- список токенов (изначально — всё предложение)
- стек (изначально — [ROOT])
- конфигурация — итоговый набор зависимостей (изначально — пустая)

Также у нас есть фиксированный набор действий, которые могут менять три сущности.

Nivre (ACL 2003): An Efficient Algorithm for Projective Dependency Parsing

Transition-based подход: действия

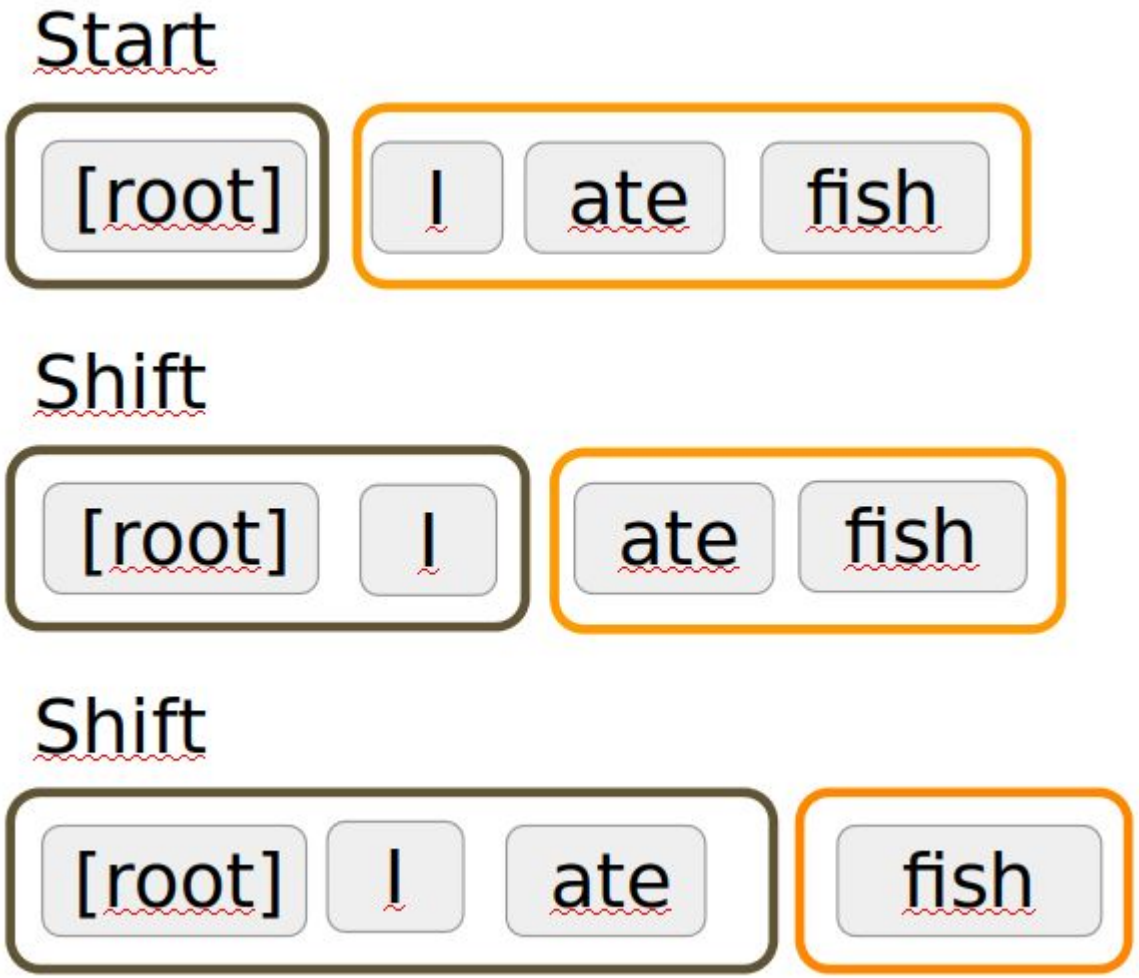
- **LeftArc** (если второй элемент стека не ROOT) — проводим зависимость от первого токена на верхушке стека к второму, и выкидываем второй из стека
- **RightArc** — проводим зависимость от второго токена на верхушке стека к первому, и выкидываем первый из стека
- **Shift** — переносим очередное слово из буфера в стек

В некоторых системах есть четвёртое действие:

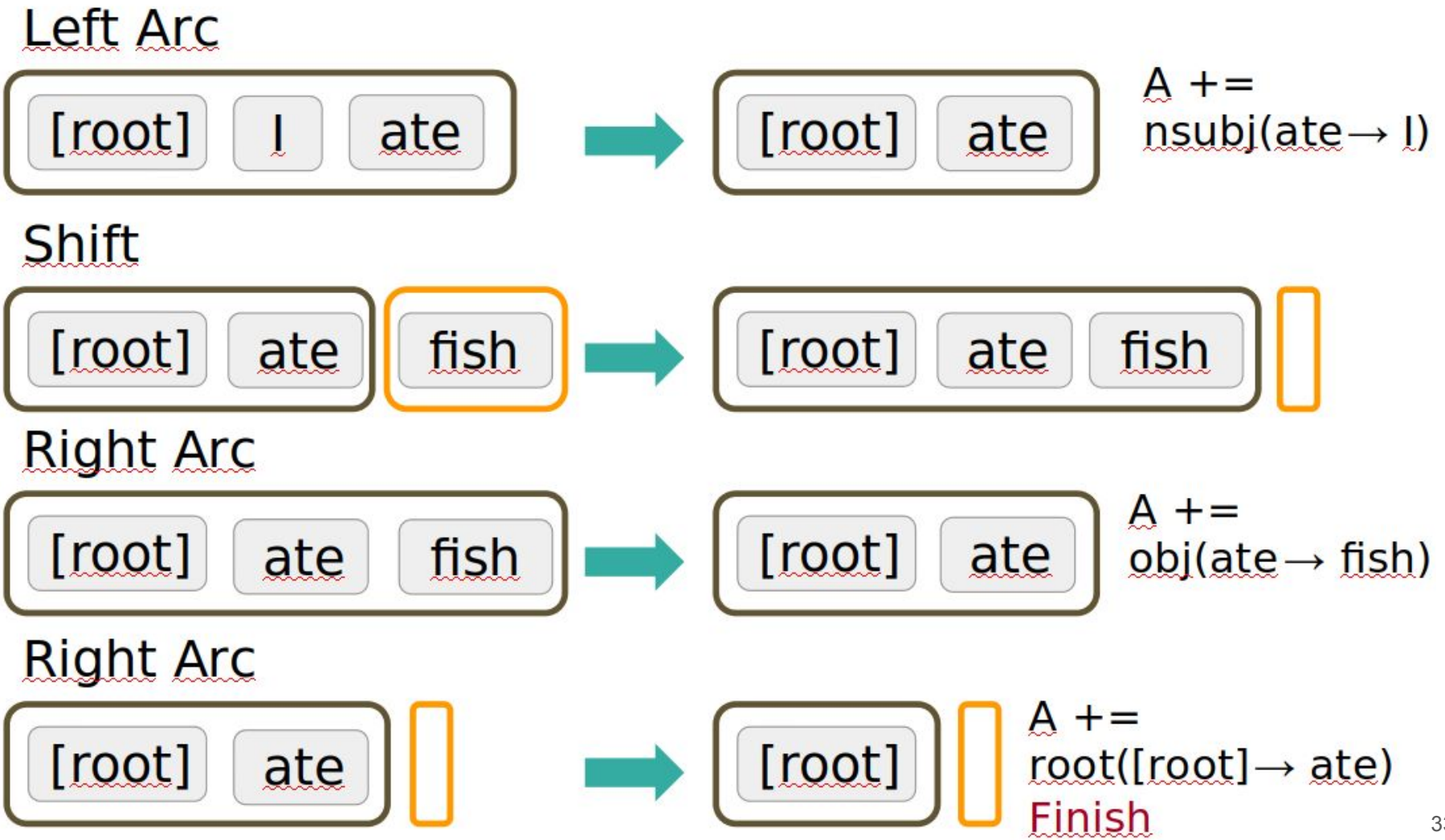
- **Swap** — вернуть второй элемент стека в буфер

Nivre et al (ACL 2009); An Improved Oracle for Dependency Parsing with Online Reordering

Пример работы на предложении “I ate fish”



Пример работы на предложении “I ate fish”



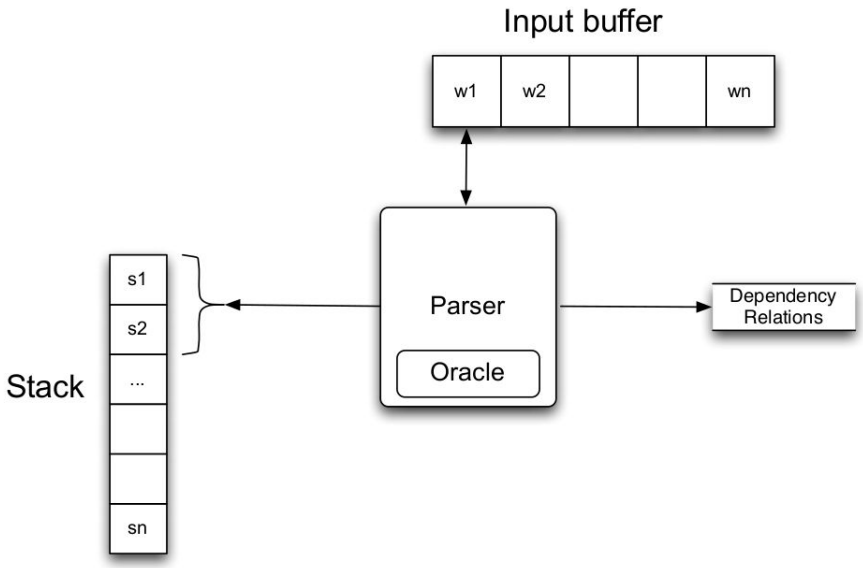
Пример работы на предложении “Book me the morning flight”

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	

Алгоритм применения модели на тесте

function DEPENDENCYPARSE(*words*) **returns** dependency tree

state ← {[root], [*words*], [] } ; initial configuration
while *state* **not final**
t ← ORACLE(*state*) ; choose a transition operator to apply
state ← APPLY(*t*, *state*) ; apply it, creating a new state
return *state*



Модификации transition-based подхода

Проблема: зависимые удаляются из стека сразу после того, как мы смогли приписать им вершину.

Но при этом у них могут быть свои зависимые…

Хорошая новость: алгоритм будет учится выкидывать их из стека в последнюю очередь.

Но можно модифицировать алгоритм

Transition-based arc-eager parsing

- LeftArc (если второй элемент стека не ROOT) — проводим зависимость от токена на верхушке буфера к токену на верхушке стека, выкидываем верхушку стека
- RightArc — проводим зависимость от токена на верхушке стека к токену на верхушке буфера, добавляем в стек верхушку буфера
- Shift — добавляем в стек верхушку буфера
- Reduce (если уже есть связь, ведущая в вершину) — выкидываем верхушку стека

Пример работы arc-eager

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

Что обучаем?

Классификатор действий

Признаки: стек, буфер, конфигурация

Первая система — 10^6-10^7 индикаторных признаков

Пример:

$$s1.w = \text{good} \wedge s1.t = \text{JJ}$$
$$s2.w = \text{has} \wedge s2.t = \text{VBZ} \wedge s1.w = \text{good}$$
$$lc(s_2).t = \text{PRP} \wedge s_2.t = \text{VBZ} \wedge s_1.t = \text{JJ}$$
$$lc(s_2).w = \text{He} \wedge lc(s_2).l = \text{nsubj} \wedge s_2.w = \text{has}$$

Классический нейросетевой парсер

Каждому слову соответствует вектор размерности 3d (вектора для слов, Pos-тегов, dependency меток)

Входной вектор составляется по 18 словам:

- 3 верхних слова в буфере
- 3 верхних слова в стеке
- 2 ближайших ребёнка слева и справа двух слов стека
- 1 ближайший ребёнок слева и справа для первых детей слева и справа двух слов стека

Chen et al (ACL 2014): A Fast and Accurate Dependency Parser using Neural Networks

Архитектура сети

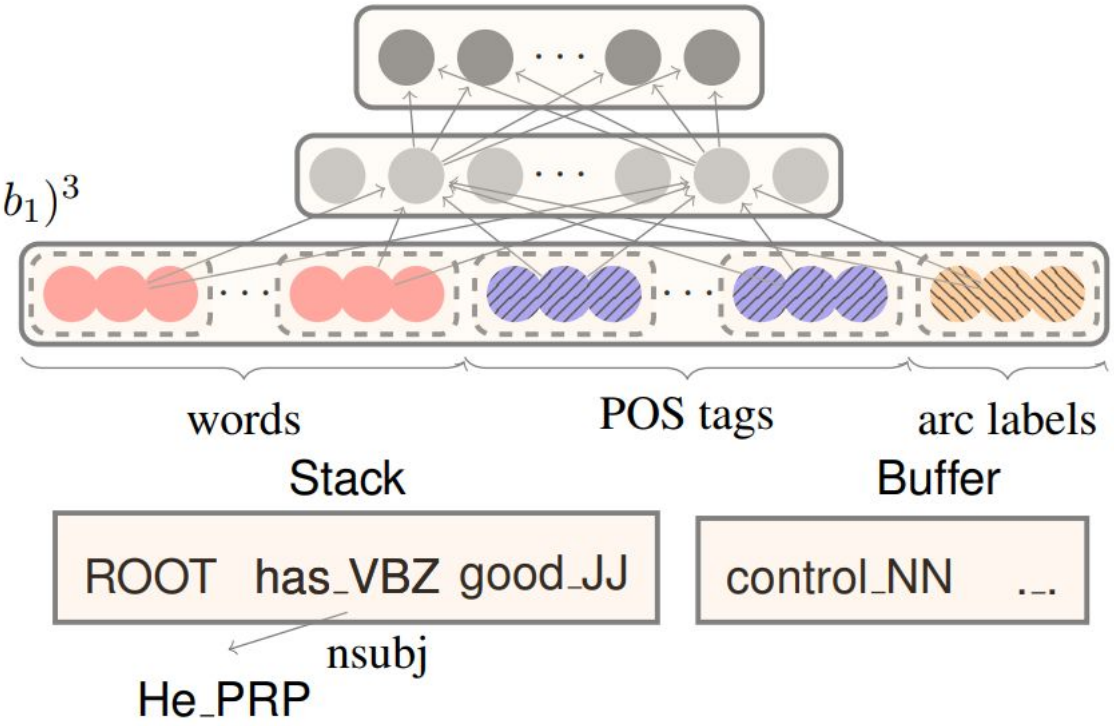
Softmax layer:

$$p = \texttt{softmax}(W_2 h)$$

Hidden layer:

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

Input layer: $[x^w, x^t, x^l]$



Особенности обучения и применения

Функционал обучения: кросс-энтропия по действиям

Применение: жадная генерация действий + beamsearch

Это не очень хорошо...

При обучении мы не учитываем способ применения.

Можно адаптировать все трюки из предыдущих лекций: генерация действий при обучении, CRF и т.п.

Что использовать на практике? UDPipe

UDPipe — пайплайн, обучаемый токенизации, лемматизации, морфологическому тэггингу и парсингу, основанному на грамматике зависимостей.

Есть готовые модели (в том числе и для русского языка).

Для синтаксиса — парсер похожий на рассмотренный.

+ помните, что если подавать на вход не сырой текст, а обработанный другими теггерами/лемматизаторами, могут быть проблемы

Использование синтаксиса в анализе тональности

Отзывы из приложения доставки

Ценник выше среднего, а так вполне неплохо, правда рыба на филе оставляет желать лучшего

Даю 2 звезды за то, что рис в роллах сварен правильно, качество сашими на высоте. Суп с морепродуктами это вода, абсолютно безвкусный и естественно холодный.

Заказывала горячие роллы, но привезли холодные. В салате цезарь не было помидоров, порции маленькие. Вообще роллы мне понравились, но больше заказывать не буду.

Анализ тональности для сущности

Тональность отзыва и отдельной сущности может различаться. Хотим определять тональность сущностей.

1) Сопоставим сущности из списка покупок со словами.
2) Построим синтаксическое дерево для всех предложений. Выделим группу, в которую входит нужное нам слово.
3) Классифицируем только фразу с этим словом без привязки к остальному тексту.

А какие проблемы?

- Построение синтаксического дерева — очень долгая операция
- Идеальные деревья получаются только на чистых текстах
- Много нюансов при предобработке данных

Полезные ссылки

- лекция Дениса Кирьянова в ВШЭ
- лекция Маннига в Стэнфорде
- Об архитектуре парсера в Spacy + библиография;
- Программа воркшопа на EMNLP-18;
- Материалы курса на ESSLLI-18;
- J. Nivre’s workshop at EACL-2014;
- SyntaxRuEval-2012.