

Синтаксический анализ

Теоретическое устройство синтаксических парсеров

Использовались материалы презентаций
Д.Кириянова, М.Апишева, Е.Большаковой.

Содержание

Синтаксический анализ

Деревья составляющих

Деревья зависимостей

Универсальные зависимости

Dependency parsing

Метрики

Инструменты

Синтаксис

Синтаксис — раздел лингвистики, изучающий строение и функциональное взаимодействие различных частей речи в предложениях, словосочетаниях и пр. языковых единицах.

Синтаксический анализ позволяет выявить структуру предложения для его дальнейшего более глубокого изучения и анализа.

Приложения синтаксического анализа:

- ▶ Машинный перевод
- ▶ Реферирование и аннотирование текста
- ▶ Снятие омонимии
- ▶ Вопросно-ответные системы

Синтаксический анализ

- ▶ Определение синтаксических связей между словами
 - ▶ Частичный синтаксический анализ: выделяют связи определенного вида
 - ▶ Полный синтаксический анализ: каждое предложение представляется в виде дерева
- ▶ Выделяют связи двух видов:
 - ▶ составляющие (constituency) – фразовая структура
 - ▶ зависимости (dependency) – иерархическая структура

	Составляющие	Зависимости
Частичный разбор	Группы (NP, VP) (chunking)	Определение семантических ролей (semantic role labelling)
Полный разбор	Дерево составляющих (constituency tree)	Дерево зависимостей (dependency tree)

Генеративная модель языка

- ▶ Язык – множество цепочек слов
- ▶ Правила порождения цепочек описываются формальными грамматиками Хомского
- ▶ Грамматика: правила вида $[aAbB] \rightarrow [aBc]$, слева и справа цепочки терминальных и нетерминальных символов
- ▶ 4 вида грамматик (Неограниченные грамматики; Контекстно-зависимые и неукорачивающие грамматики; Контекстно-свободные грамматики; Регулярные грамматики)
- ▶ Для естественных языков используются контекстно-свободные грамматики вида $A \rightarrow aVa$
 - ▶ Слева – ровно один нетерминальный символ
 - ▶ Справа – произвольная цепочка
- ▶ Дерево вывода цепочки-предложения – дерево составляющих

Деревья составляющих

Модель пытается разбить предложение на более мелкие группы, группы — на подгруппы, и далее, пока не дойдёт до отдельных слов.

Пример: «Мама мыла раму»¹



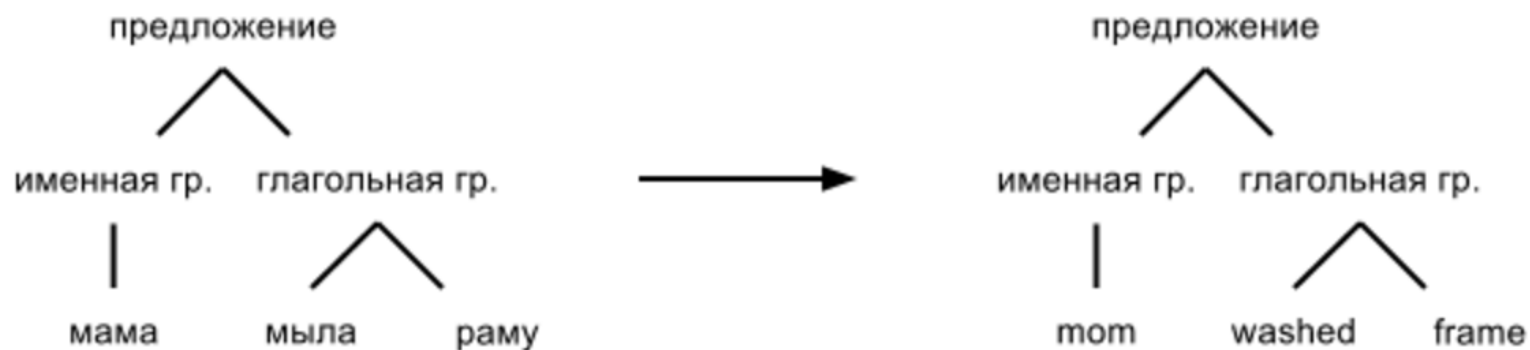
- ▶ разбиваем предложение на именную и глагольную группы;
- ▶ именную группу разбиваем на существительное (мама);
- ▶ глагольную группу на глагол (мыла) и на вторую именную группу;
- ▶ вторую глагольную группу на существительное (раму).

¹<https://habrahabr.ru/post/148124/>

Использование и проблемы модели

Для разбора предложения можно воспользоваться парсерами формальных языков

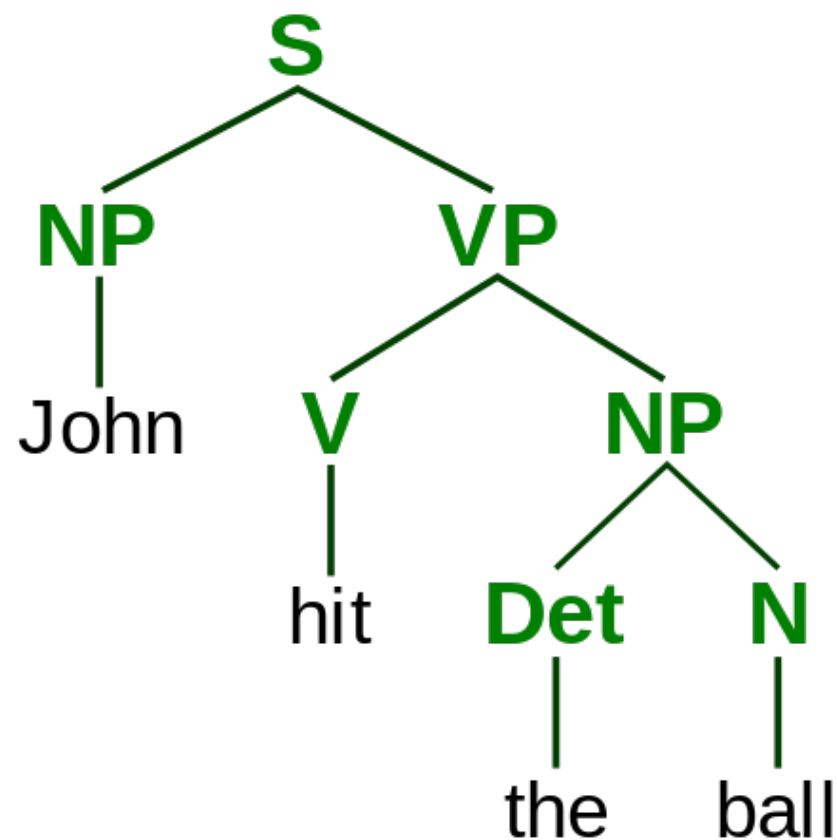
Способ последующего использования — машинный перевод:



Проблемы модели:

- ▶ Неоднозначность — одному предложению могут соответствовать несколько различных деревьев. **Придумайте пример!**
- ▶ Плохо подходит для русского языка — часто в предложениях можно менять порядок слов, не меня смысла («Мыла мама раму»).

Пример дерева составляющих



- ▶ S, NP, VP – нетерминальные символы
- ▶ V, N, Det – терминальные символы

Метки-обозначения

S – предложение

NP – именная группа

VP – глагольная группа

AnV – аналитическая
форма глагола (будет делать)

PP – предложная группа

Pronoun – местоимение

Det – местоименное прилагательное

N – имя существительное

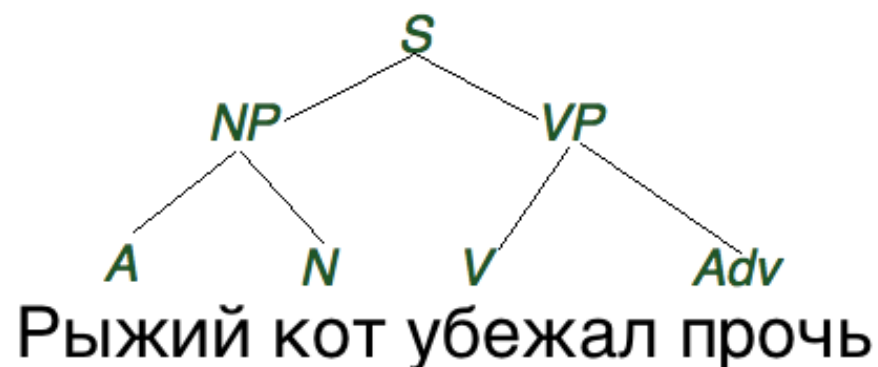
Adv – наречие

Aux – вспомогательный глагол

V – глагол

Prep – предлог

A – имя прилагательное



Задание

- ▶ Построить дерево составляющих предложения и записать его КС-грамматику:

Эти школьники скоро будут писать диктант по русскому языку

- ▶ По КС-грамматике построить нетривиальное дерево составляющих (подобрать предложение):

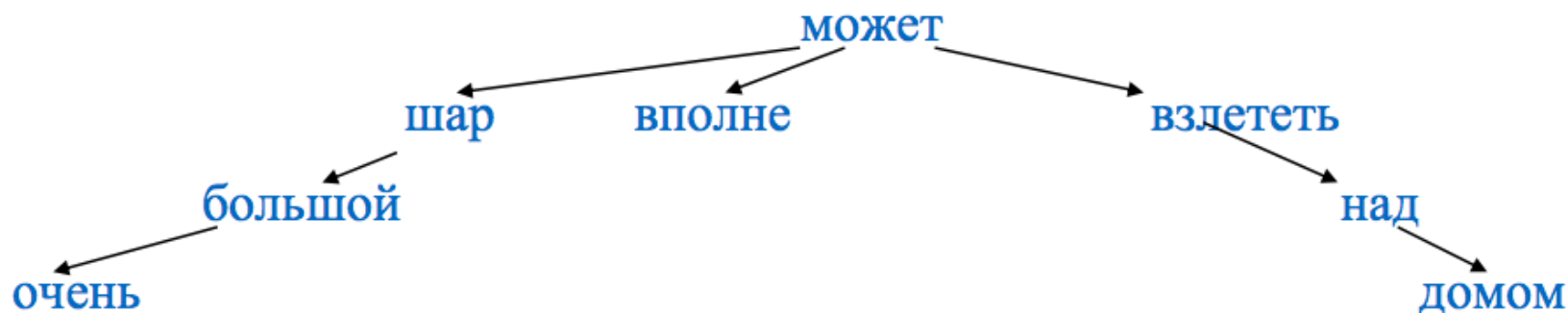
S	→	NP VP
NP	→	Pronoun Noun Det Adj Noun NP PP
PP	→	Prep NP
V	→	Verb Aux Verb
VP	→	V V NP V NP NP V NP PP VP PP

Деревья зависимостей

- ▶ Основа – подчинительная связь слов
- ▶ Дерево зависимостей (подчинения) предложения:
 - ▶ узлы – слова (корень дерева – глагол, сказуемое)
 - ▶ дуги – подчинительная связь (зависимость)
- ▶ Особенность: дерево предложения должно быть дополнено информацией о линейной структуре (т.е. задан порядок слов):

Упрощенный пример дерева зависимостей

Очень большой шар вполне может взлететь над домом



- ▶ На дугах не указаны отношения подчинения

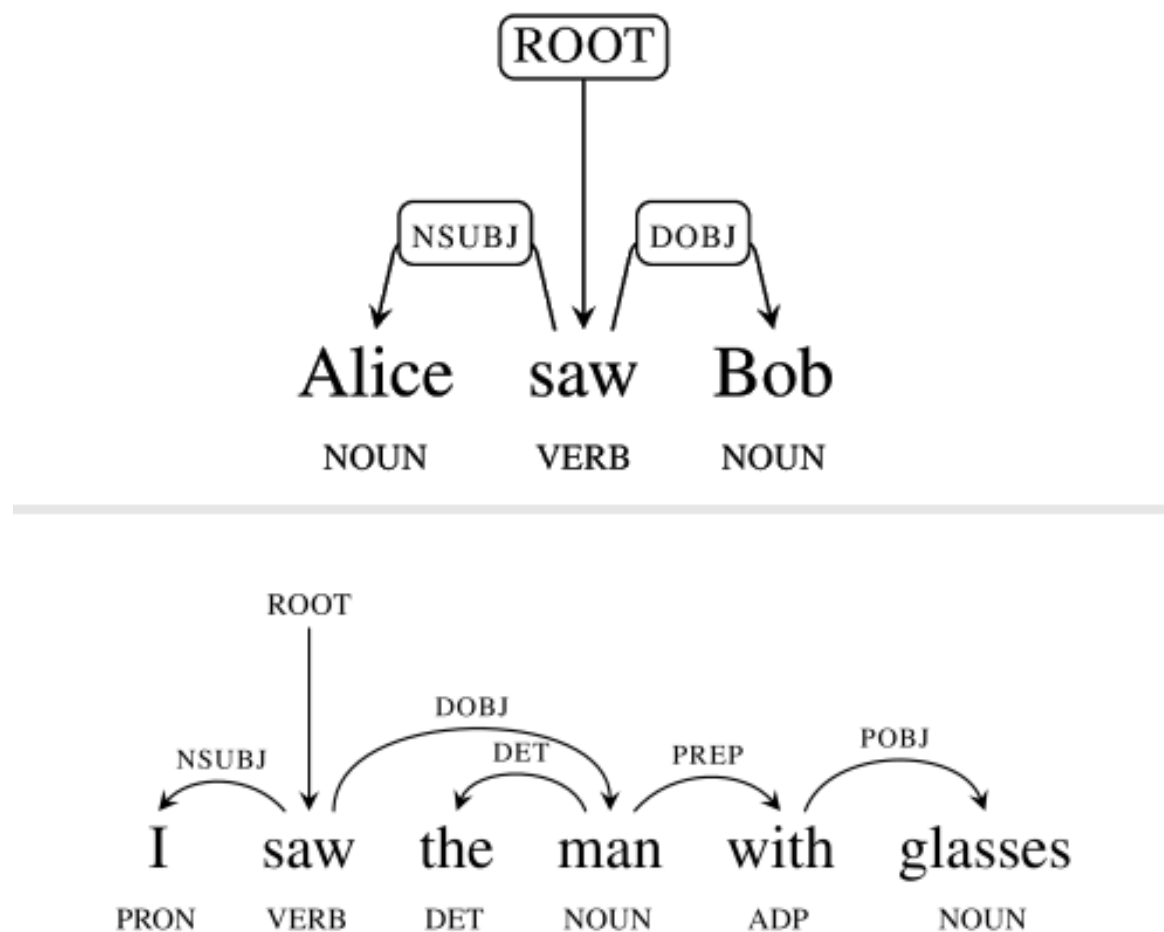
Упражнение: построить упрощенное дерево зависимостей для того же предложения про школьников.

Типы синтаксических связей

Наиболее распространенные типы:

- Прямообъектное: *уделить* → *внимание*, *вижу* → *лес*
- Определительное: *очень* ← *хорошо*, *важный* ← *вопрос*
- Отпредложное: *в* → *здание*, *хлеб* → *с* → *маслом*
- Предикат (сказуемое) и субъект (подлежащее):
спасатели ← *обнаружили*
- Посессивное: *книга* → *врача*
- Аппозитивное (приложение): *мальчик* ← *Петя*
- Количественное: *пять* ← *машин*
- обстоятельство: *лежать* → *на* → *полу*
- Ограничительное: *не* → *для* → *всех*

Примеры деревьев зависимостей




- ▶ Все слова в предложении связаны отношением типа “хозяин-слуга”, имеющим различные подтипы

Последующее использование

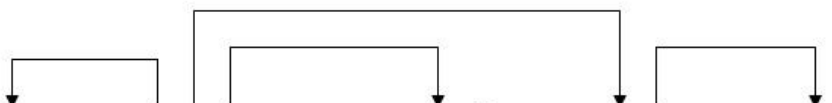
Снятие синтаксической омонимии:

1. Он вынул трубку из глины.



A syntactic tree diagram for the sentence "Он вынул трубку из глины." The root node branches into four children: "Он", "вынул", "трубку", and "из глины".

2. Он вынул трубку из глины.



A syntactic tree diagram for the sentence "Он вынул трубку из глины." The root node branches into three children: "Он", "вынул", and "трубку из глины". The node "трубку из глины" further branches into "трубку" and "из глины".

Еще применения синтаксического разбора

- «Банкомат съел карту» vs «карта съела банкомат»;
- Определение правильности грамматики фразы (при порождении речи);
- Синтаксическая роль токена как метрика его важности (подлежащее важнее определения), использование весов в классификаторе.

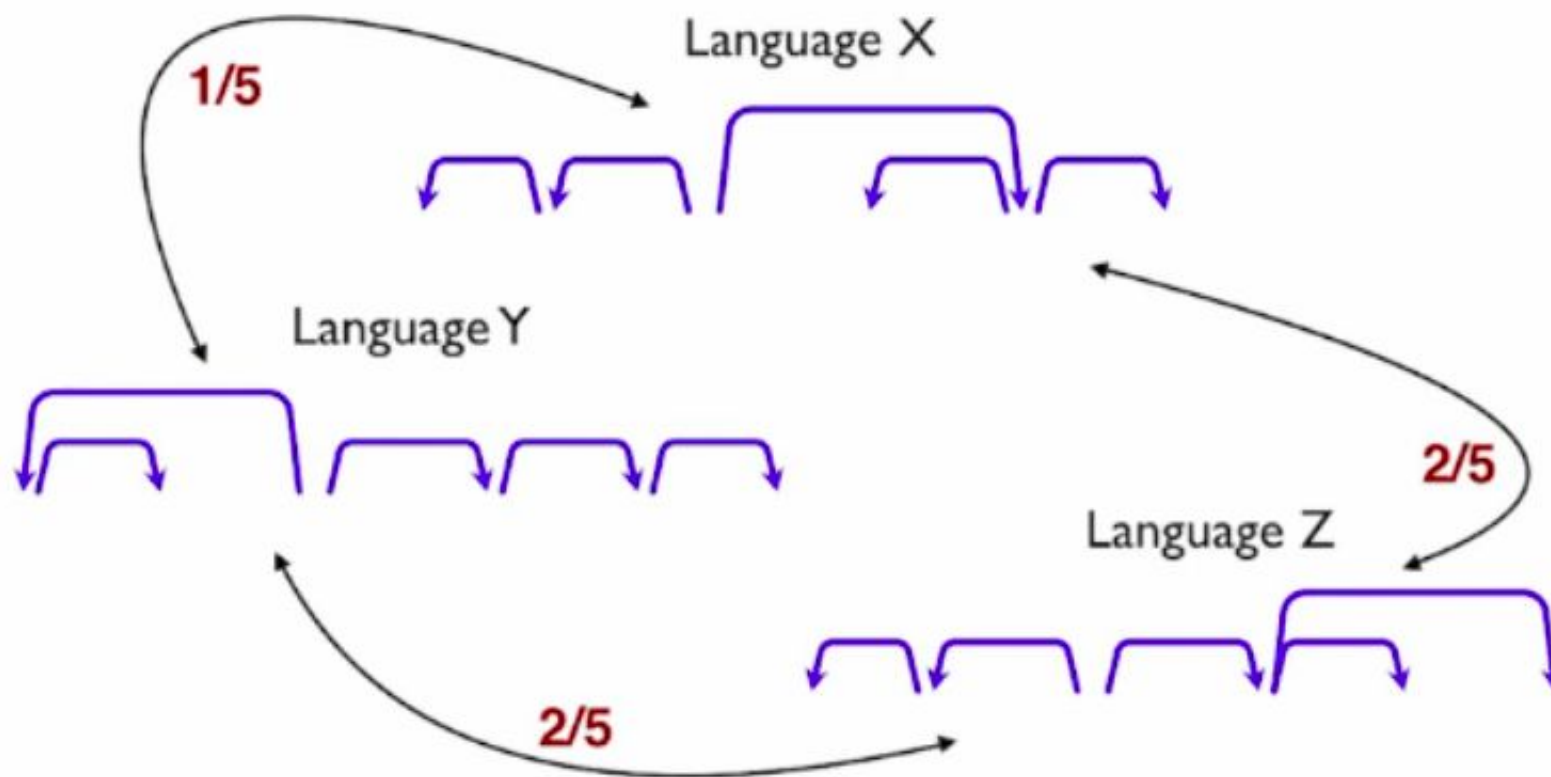
Итоги по деревьям

- Деревья составляющих
 - фиксируют словосочетания, но не связи между ними
 - отражают одновременно синтаксическую и линейную структуру предложения
 - не позволяют представлять разорванные синт. единицы
 - больше подходят для описания синтаксиса языков со **строгим (жестким)** порядком слов (английский и др.).
- Деревья зависимостей
 - отображают разнотипные связи, но только между словами;
 - имеют сложности в отображении неподчинительных отношений, например, однородных членов предложения:
красивый и умный
 - подходят для языков с достаточно **свободным порядком слов** (русский, испанский и др.).

Демо

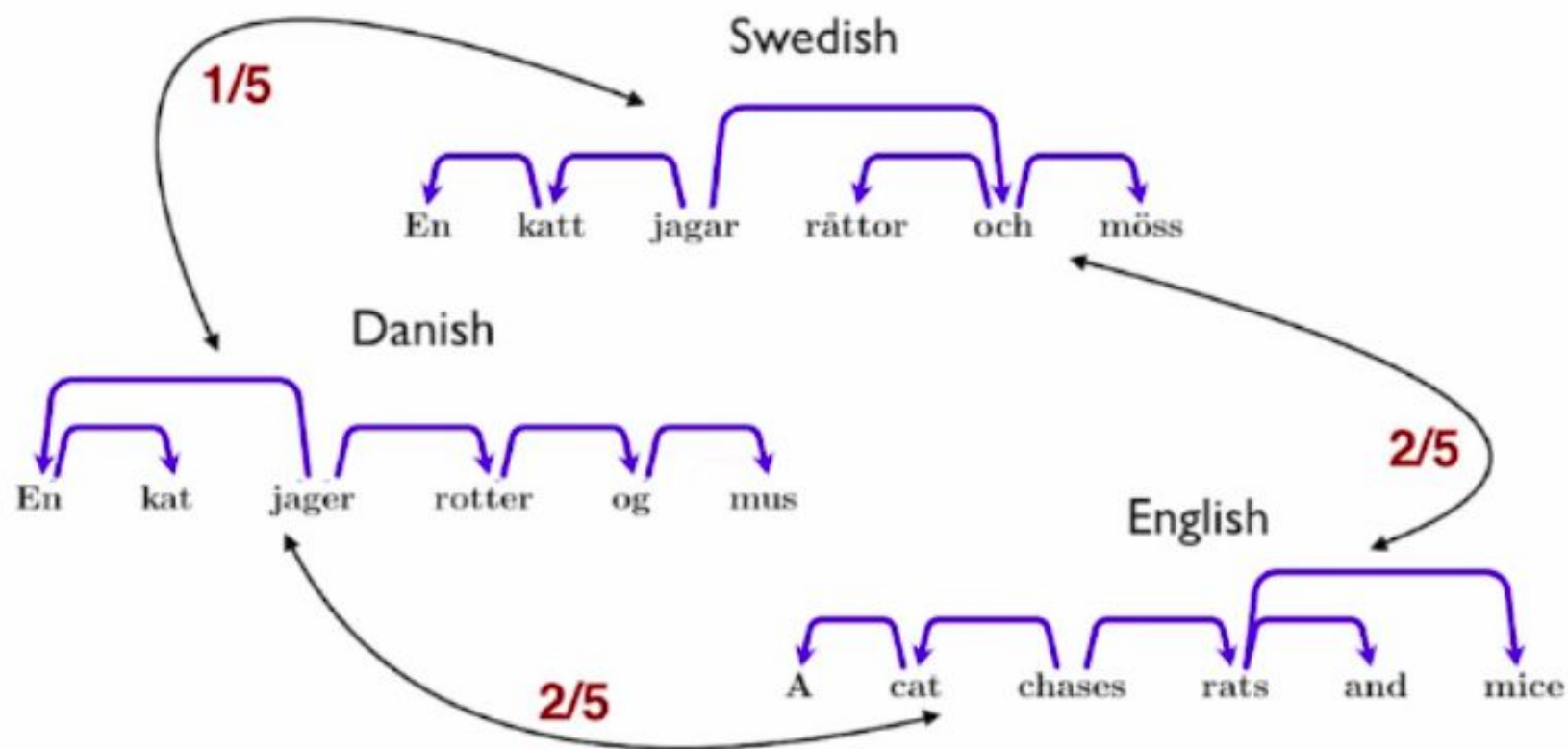
- ▶ Berkley Tomcat constituency parser
<http://tomato.banatao.berkeley.edu:8080/parser/parser.html>
- ▶ ARK dependency parser (Carnegie Mellon)
<http://demo.ark.cs.cmu.edu/parse>

Универсальные зависимости



Which languages are most closely related?

Универсальные зависимости

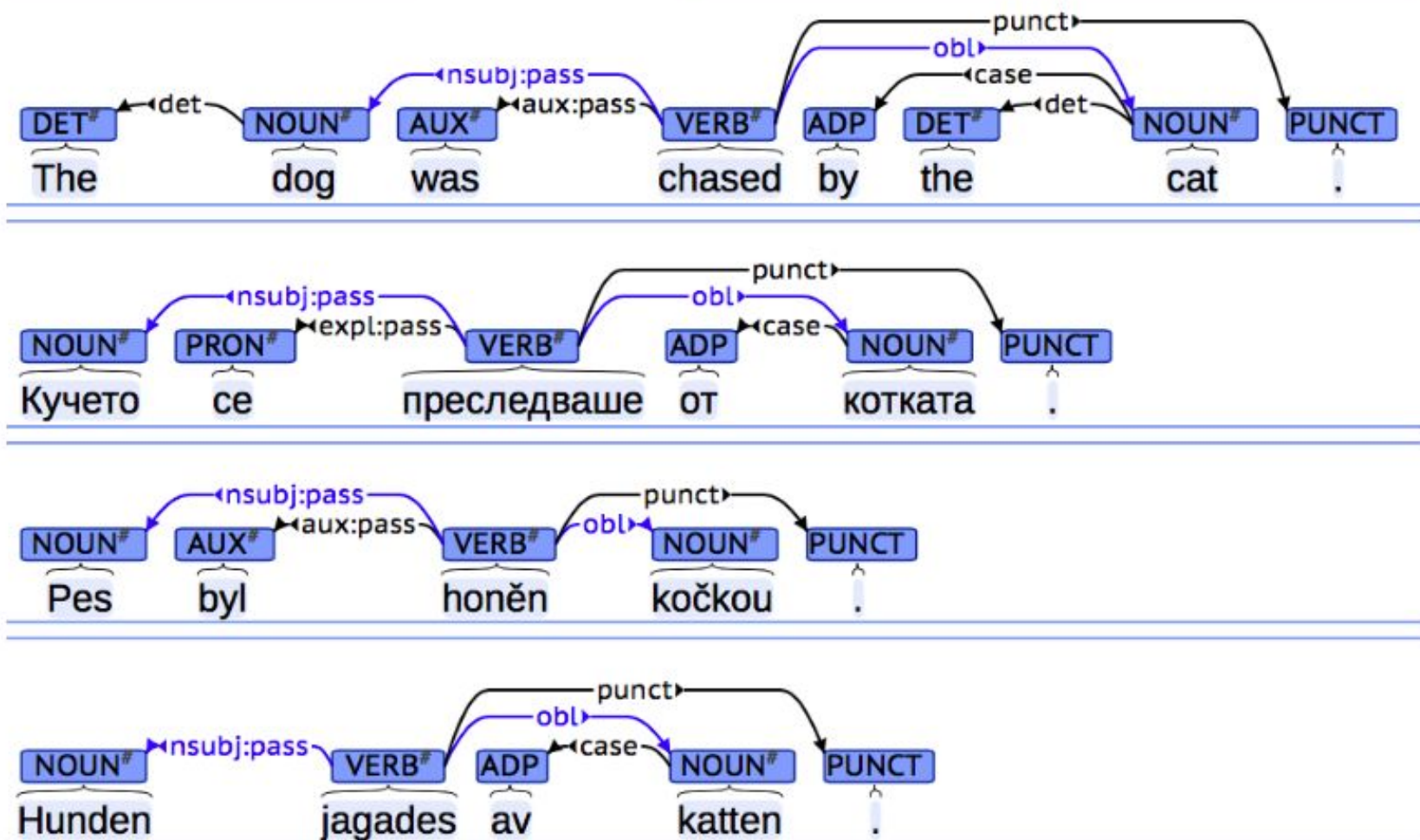


Which languages are most closely related?

Проект Universal dependencies

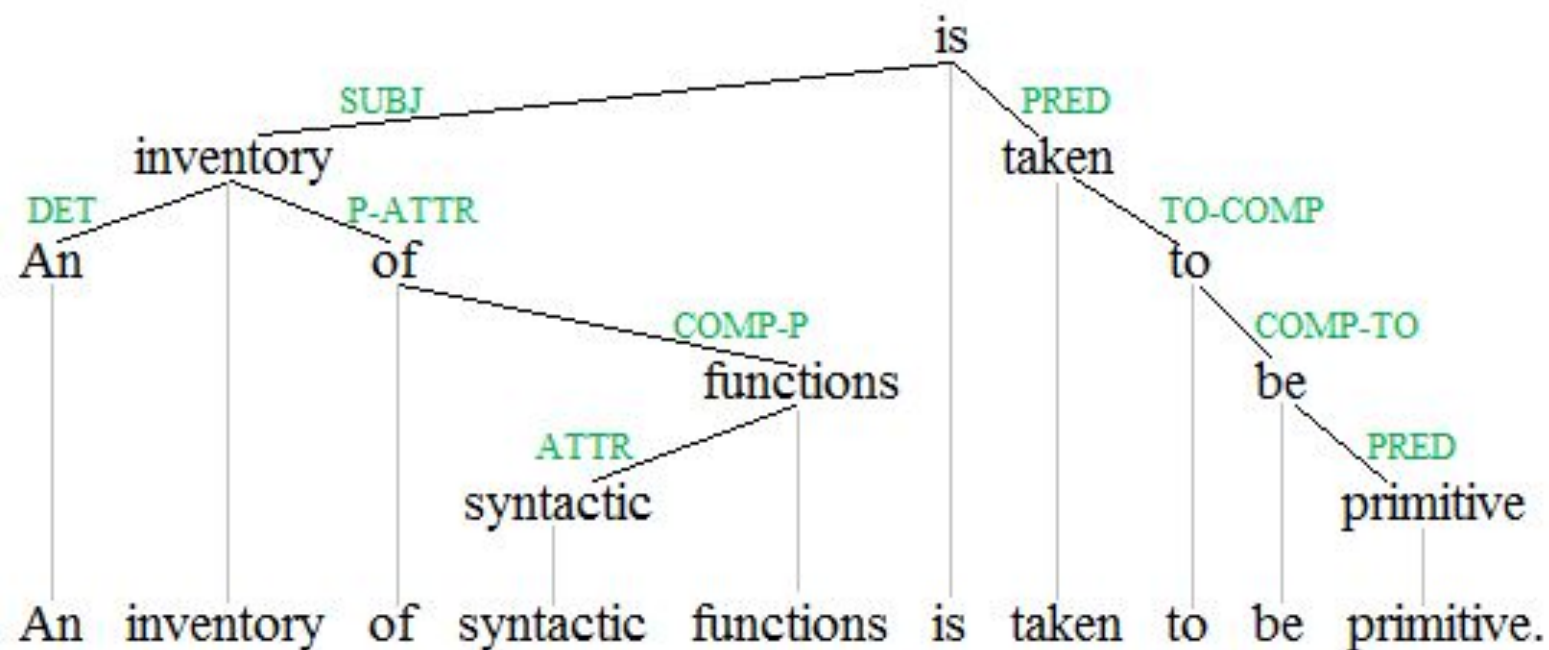
- Лингвистическая проблема: несоответствие терминов и правил из грамматик зависимостей разных языков;
- Computational challenge: обучить синтаксический парсер для многих языков, включая low-resource languages;
⇒ <http://universaldependencies.org/>
- > 100 версионированных трибанков (размеченных корпусов) для 60 языков, теги зависимостей [унифицированы](#).

Универсальные зависимости

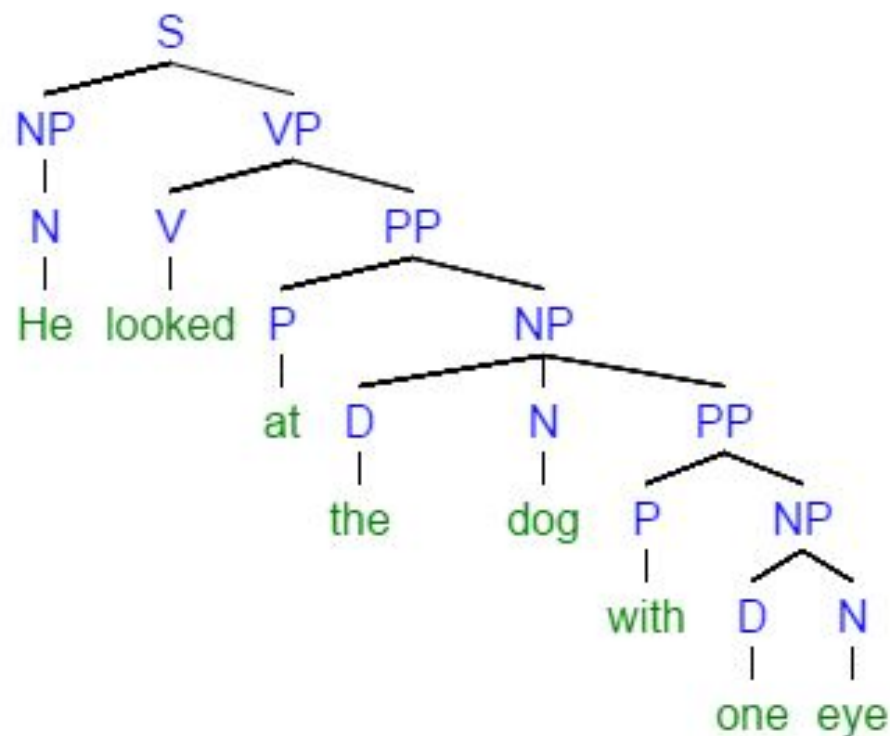


<https://events.yandex.ru/lib/talks/3516/>

Контрольный вопрос:
составляющие или зависимости?



А здесь составляющие или зависимости?



VP - глагольная группа, verb phrase
(грубо: состоит из глагола и зависимых;
но не подлежащее)

NP - именная группа, noun phrase
(грубо: вершина — существительное)

PP - предложная группа, prepositional
phrase

AP - группа прилагательного, adjective
phrase

D (Det) - детерминативы: артикли, указ.,
притяж., определительные
местоимения, квантификаторы,
числительные, вопросительные слова
...

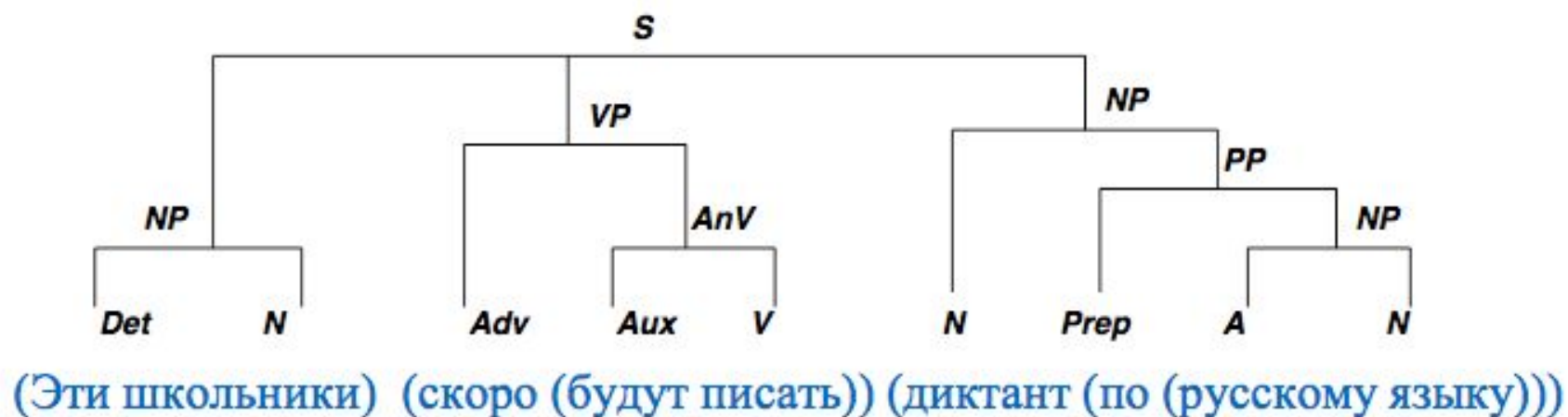
Какой еще может быть разбор?

Возможные ответы на вопросы из упражнений

Эти типы стали есть на складе.

Подпись начальника отдела или доверенного лица.

Эти школьники скоро будут писать диктант по русскому языку.



Дерево зависимостей [[Jurafsky & Martin 2017](#)]

“Dependency tree is a directed graph that satisfies the following constraints:

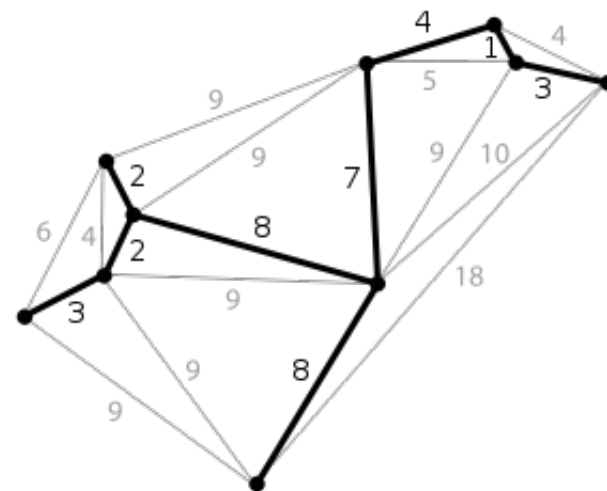
- There is a single designated root node that has no incoming arcs.
- With the exception of the root node, each vertex has exactly one incoming arc.
- There is a unique path from the root node to each vertex in V .”

Dependency parsing: алгоритмы

Построение дерева зависимостей по предложению.

Два основных подхода

- **transition-based**
жадно набираем дерево
- **graph-based** (minimum spanning tree)
ищем минимальное остовное дерево в полном графе всех возможных связей



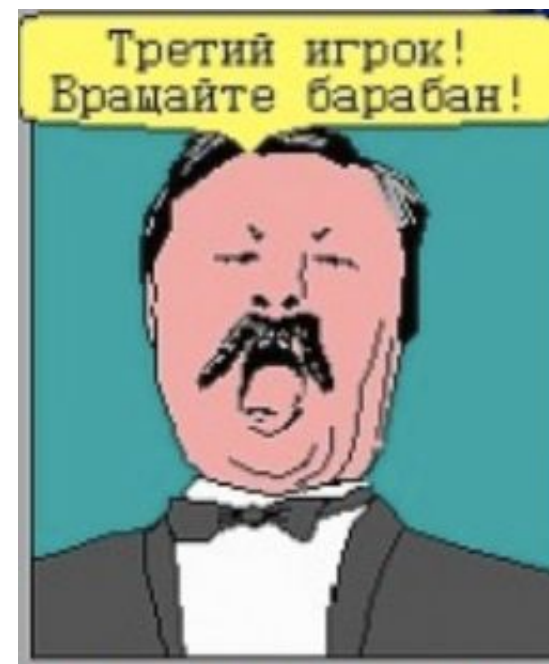
Подход **graph-based** быстрее работает при непроективности и обычно лучше работает с длинными предложениями.

Transition-Based DP: интуиция

Представим, что нам Леонид Якубович открывает по одному слову, а мы строим разбор предложения на лету

Book...

Окей, что-то про книгу, книга может быть и подлежащим

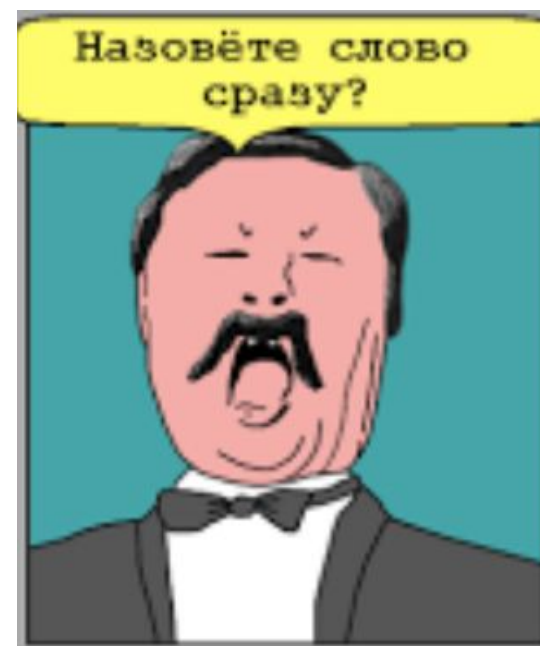


Transition-Based DP: интуиция

Представим, что нам Леонид Якубович открывает по одному слову, а мы строим разбор предложения на лету

Book me...

ан нет! “забронируй меня” или “забронируй мне”,
но **me** явно зависимое



Transition-Based DP: интуиция

Представим, что нам Леонид Якубович открывает по одному слову, а мы строим разбор предложения на лету

Book me the...

пока непонятно, но всё-таки это просьба
забронировать **что-то**

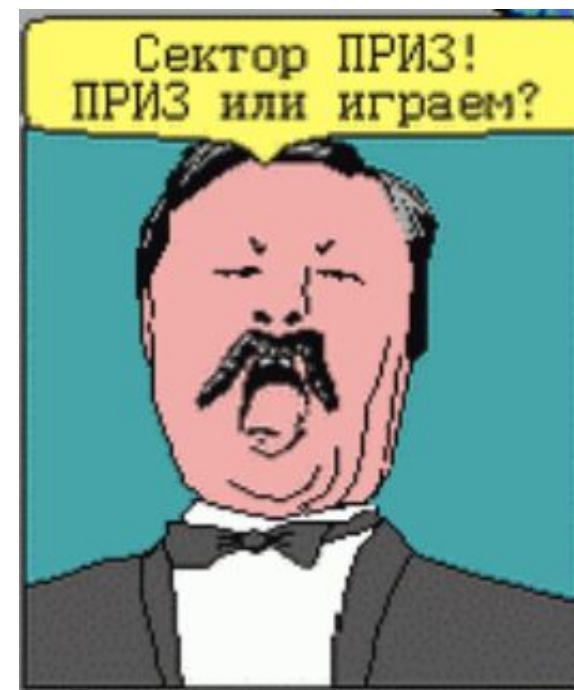


Transition-Based DP: интуиция

Представим, что нам Леонид Якубович открывает по одному слову, а мы строим разбор предложения на лету

Book me the morning...

“забронируй мне утро?” странновато, конечно (парсеру зависимостей это м.б. и не важно), но — **morning** может зависеть от **book**:
“забронируй мне утро у стоматолога”

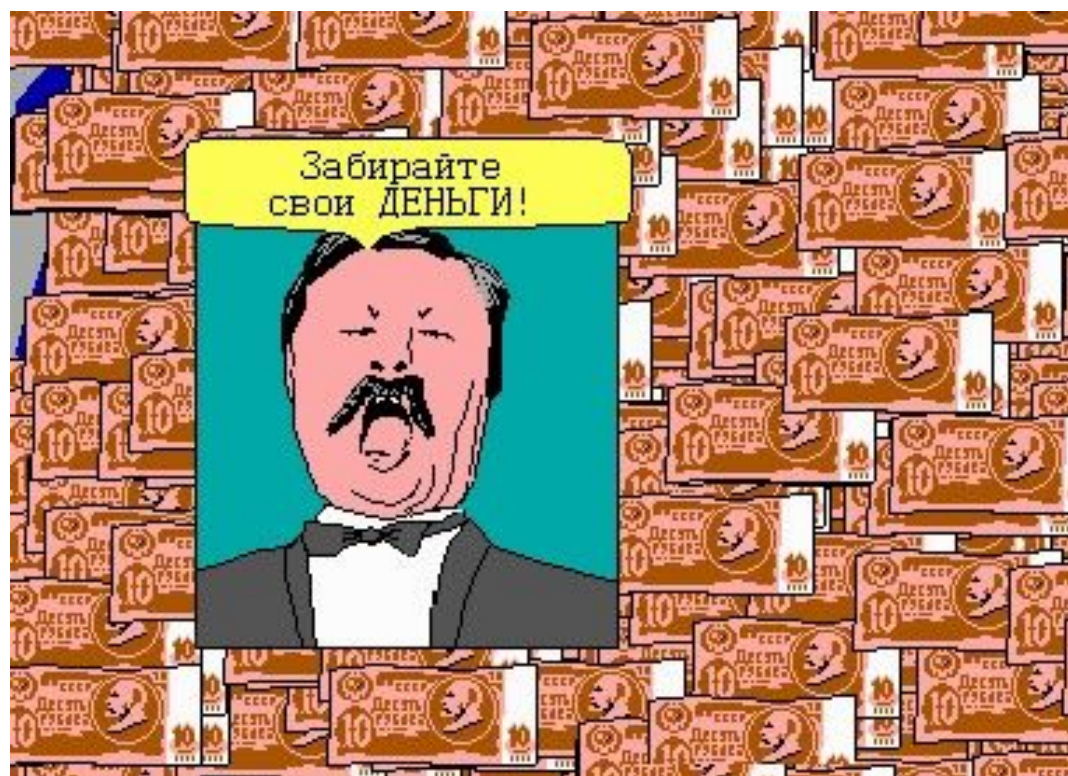


Transition-Based DP: интуиция

Представим, что нам Леонид Якубович открывает по одному слову, а мы строим разбор предложения на лету

Book me the morning flight.

А вот теперь всё ясно!



Transition-Based DP: arc-standard parsing

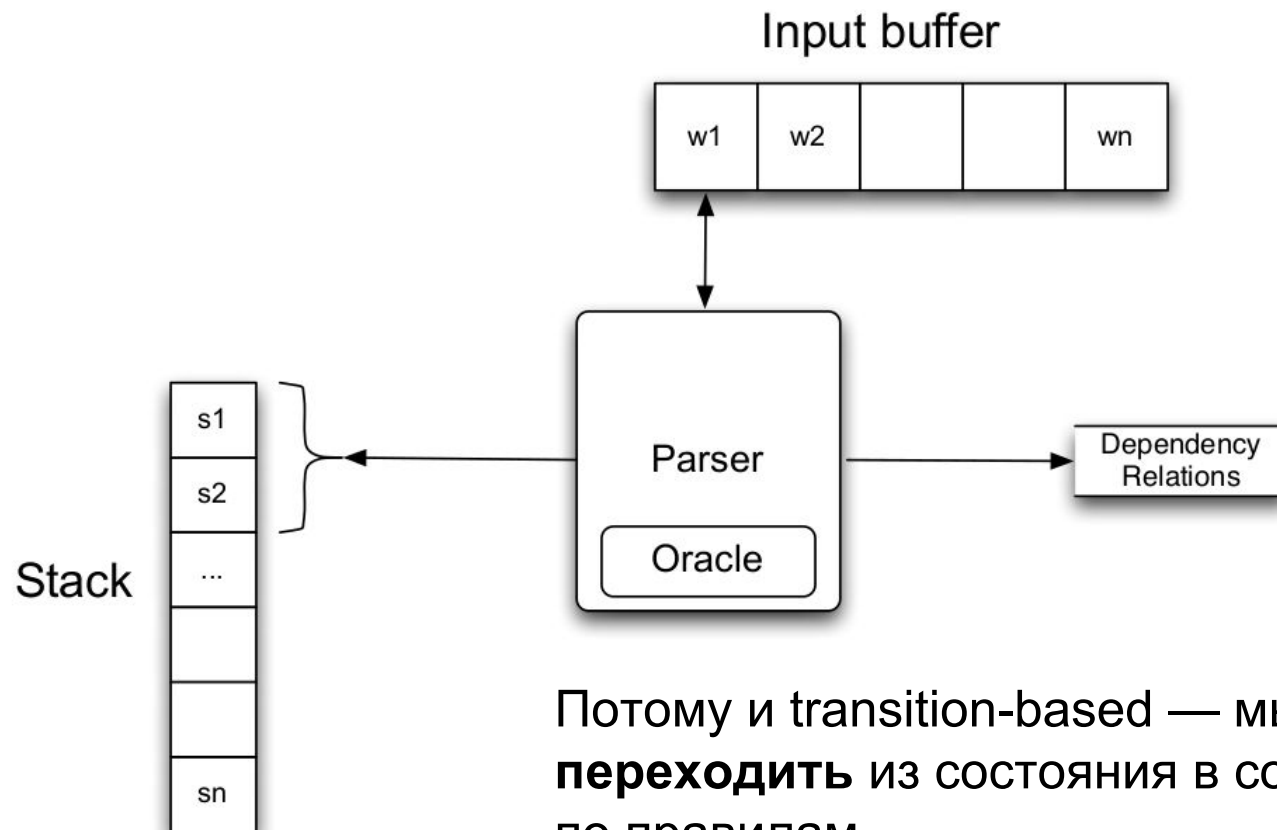
Есть список токенов, стек (изначально в нем только root) и конфигурация (изначально пустая).

Три способа изменить конфигурацию:

- **LeftArc** [применим, если второй элемент стека не ROOT]
проводим зависимость между токеном на верхушке стека и вторым + выкидываем второй из стека
 - **RightArc**
аналогично проводим зависимость в другую сторону + выкидываем верхушку стека
 - **Shift**
переносим очередное слово из буфера в стек
- + **Swap** вернуть второй элемент стека в буфер, см. [[Nivre 2009](#)]

Transition-Based DP: arc-standard parsing

Ключевое понятие: **конфигурация** = **состояние** процесса разбора:
входящие токены, верхушка стека и набор уже построенных отношений
(то, что мы “держали в уме”, когда играли; да, аналогия не вполне точна)



Потому и transition-based — мы сейчас будем **переходить** из состояния в состояние системы по правилам.

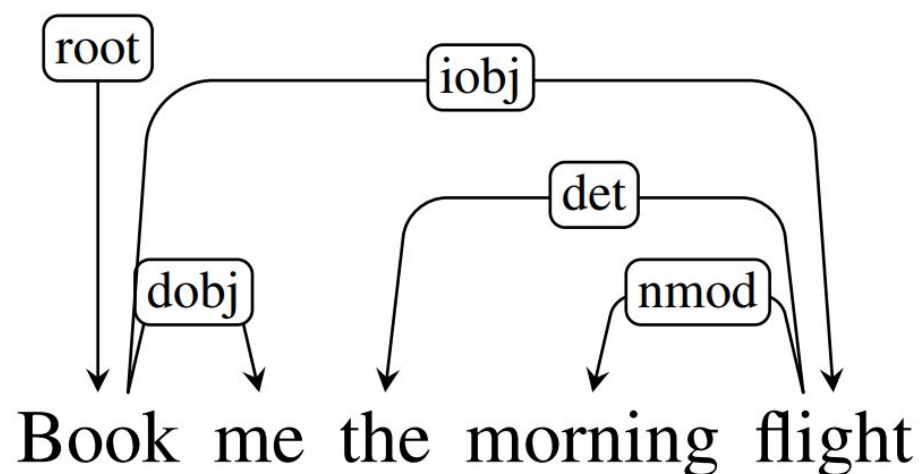
Фичи (признаки) и трибанки (treebanks)

In linguistics, a **treebank** is a parsed text corpus that annotates syntactic or semantic sentence structure.

ID	FORM	LEMMA	UPOS	FEATS	HEAD	DEPREL
1	Переведи	перевести	VERB	Mood=Imper Number=Sing Person=2	0	root
2	маме	мама	NOUN	Animacy=Anim Case=Dat Gender=Fem Number=Sing	1	iobj
3	сто	сто	NUM	Case=Nom	4	nummod
4	рублей	рубль	NOUN	Animacy=Inan Case=Gen Gender=Masc Number=Plur	1	nmod

Пример работы [[Jurafsky & Martin 2017](#)]

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



Transition-based parsing

Есть разные модификации, в частности, arc-eager вместо arc-standard:

Проблема: зависимые удаляются из стека сразу после того, как мы смогли приписать им вершину. Но при этом у них могут быть свои зависимые — и проверяя, можем ли мы уже проводить связь или надо ждать еще зависимых, мы рискуем свернуть не туда. См. в разборе выше *book*: мы быстро поняли, что *book* — это root, но связали их лишь на последнем шаге.

Решение: слегка поменять операции, см. далее.

- + еще бывает beam search для того, чтобы не быть настолько жадными (иногда это тоже вызывает проблемы)

Transition-based DP: arc-eager parsing

- + **LeftArc**

проводим зависимость между токеном на верхушке **буфера** и токеном на верхушке стека, выкидываем верхушку стека вообще;

- + **RightArc**

то же, но зависимость в другую сторону, и верхушка буфера становится верхушкой стека

- + **Shift**

переносим очередное слово из буфера в стек

- + **Reduce**

выкидываем верхушку стека

=> Из стека можно выкинуть только то, у чего уже нашлась вершина правее!

Пример: Transition-based arc-eager parsing

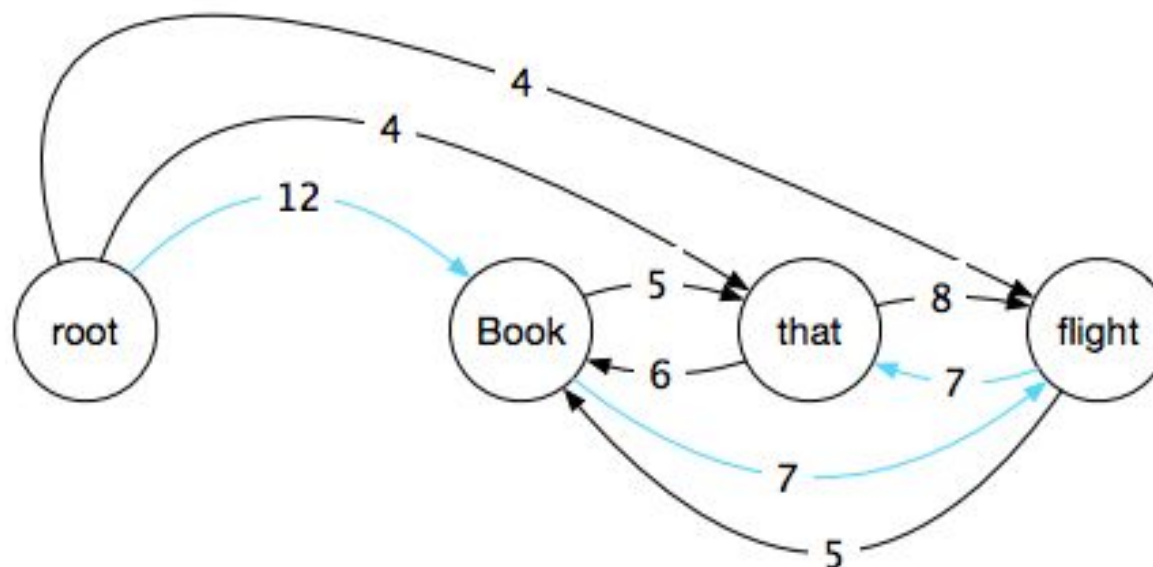
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, the, flight, through, houston]	RIGHTARC	(root → book)
1	[root, book]	[the, flight, through, houston]	SHIFT	
2	[root, book, the]	[flight, through, houston]	LEFTARC	(the ← flight)
3	[root, book]	[flight, through, houston]	RIGHTARC	(book → flight)
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	

Сравните с arc-standard

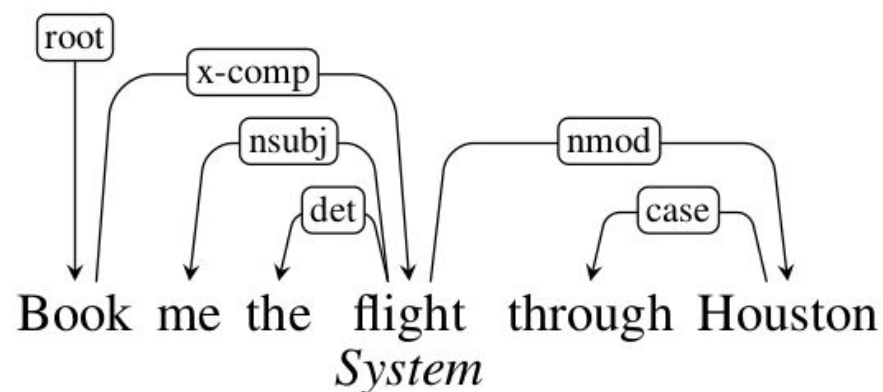
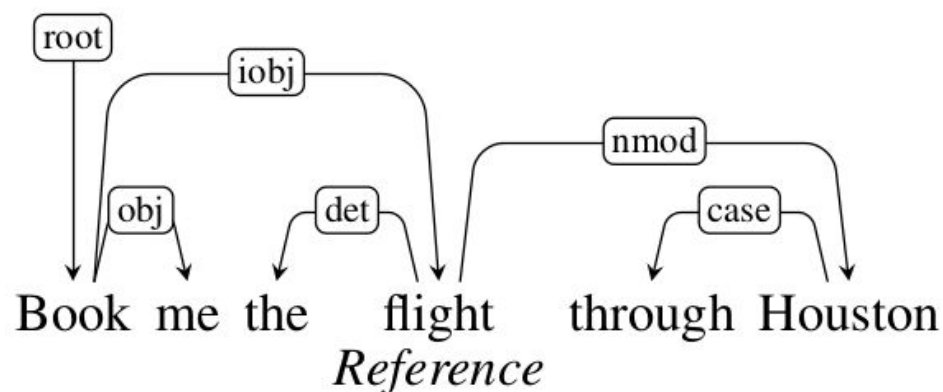
Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, 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	

Graph-based dependency parsing

- Изначально имеем полный орграф;
- Все ребра и все типы связей;
- При обучении учимся скорить связи;
- Фичи те же, что у transition-based + могут быть фичи про порядок слов;
- Находим минимальное (максимальное) остовное дерево



Оценка качества



Unlabeled Attachment Score (UAS) = 5/6

(правильно приписана вершина)

Labeled Attachment Score (LAS) = 4/6

(правильно приписана вершина И тип метки)

- + macro-averaged vs micro-averaged
(по предложениям vs независимо от предложений)

Accuracy, Точность, Полнота, F-мера.

Инструменты. UDPipe

- UDPipe — пайплайн, обучаемый токенизации, лемматизации, морфологическому тэггингу и парсингу, основанному на грамматике зависимостей;
- [Статья об архитектуре](#), [репозиторий с кодом обучения](#), [мануал](#);
- Есть готовые [модели](#) (в том числе и для РЯ);
- Подобранные для каждого корпуса параметры обучения [зарелизены](#).