Основы обработки текстов

Лекция 8

Статистические методы синтаксического анализа

Мотивация

- СКС-грамматики позволяют определить лучшее дерево разбора (т.е. устранить многозначность)
- Более точное моделирование языка, по сравнению с n-граммами
  - распознавание речи
  - машинный перевод
  - извлечение информации
  - ...
  - выделение ключевых слов

Обработка текстов

- Стохастические контекстно-свободные грамматики (СКС)
  - разрешение синтаксической многозначности
  - моделирование языка
- Вероятностная версия алгоритма CKY
- Обучение СКС
- Проблемы СКС
  - разделение и слияние нетерминалов
  - СКС с поддержкой лексики
    - алгоритм Коллинза
- Разбор в грамматику зависимости
- Методы оценки

Стохастические контекстно-свободные грамматики

N	множество **нетерминальных символов**
Σ	множество **терминальных символов** (непересекающееся с *N*)
R	множество правил, каждое вида $A \to \beta[p]$ где A - нетерминал, β - строка символов из множества $(\Sigma \cup N)*$ p - вероятность правила $P(\beta \mid A)$, $\sum_{\beta} P(A \to \beta) = 1$
S	символ начала

Пример

Грамматика	Вероятность		
S → NP VP	0.8	+	1.0
S → Aux NP VP	0.1		
S → VP	0.1		
NP → Pronoun	0.2	+	1.0
NP → Proper-Noun	0.2		
NP → Det Nominal	0.6		
Nominal → Noun	0.3	+	1.0
Nominal → Nominal Noun	0.2		
Nominal → Nominal PP	0.5		
VP → Verb	0.2	+	1.0
VP → Verb NP	0.5		
VP → VP PP	0.3		
PP → Prep NP	1.0		

Лексикон

Det → the | a | that | this
0.6 0.2 0.1 0.1

Noun → book | flight | meal | money
0.1 0.5 0.2 0.2

Verb → book | include | prefer
0.5 0.2 0.3

Pronoun → I | he | she | me
0.5 0.1 0.1 0.3

Proper-Noun → Houston | NWA
0.8 0.2

Aux → does
1.0

Prep → from | to | on | near | through
0.25 0.25 0.1 0.2 0.2

Разрешение многозначности

- Вероятность разбора

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

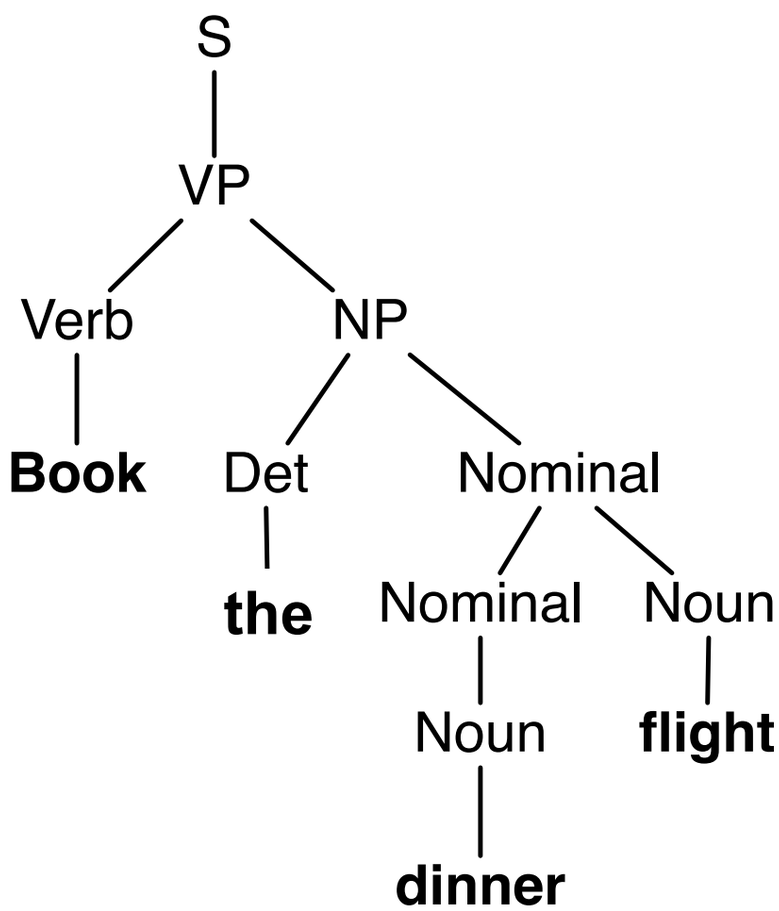
- Вероятность $P(T,S) = P(T)P(S|T) = P(T)$
- Выбор наиболее вероятного дерева разбора $\hat{T}(S) = \arg\max_T P(T|S)$

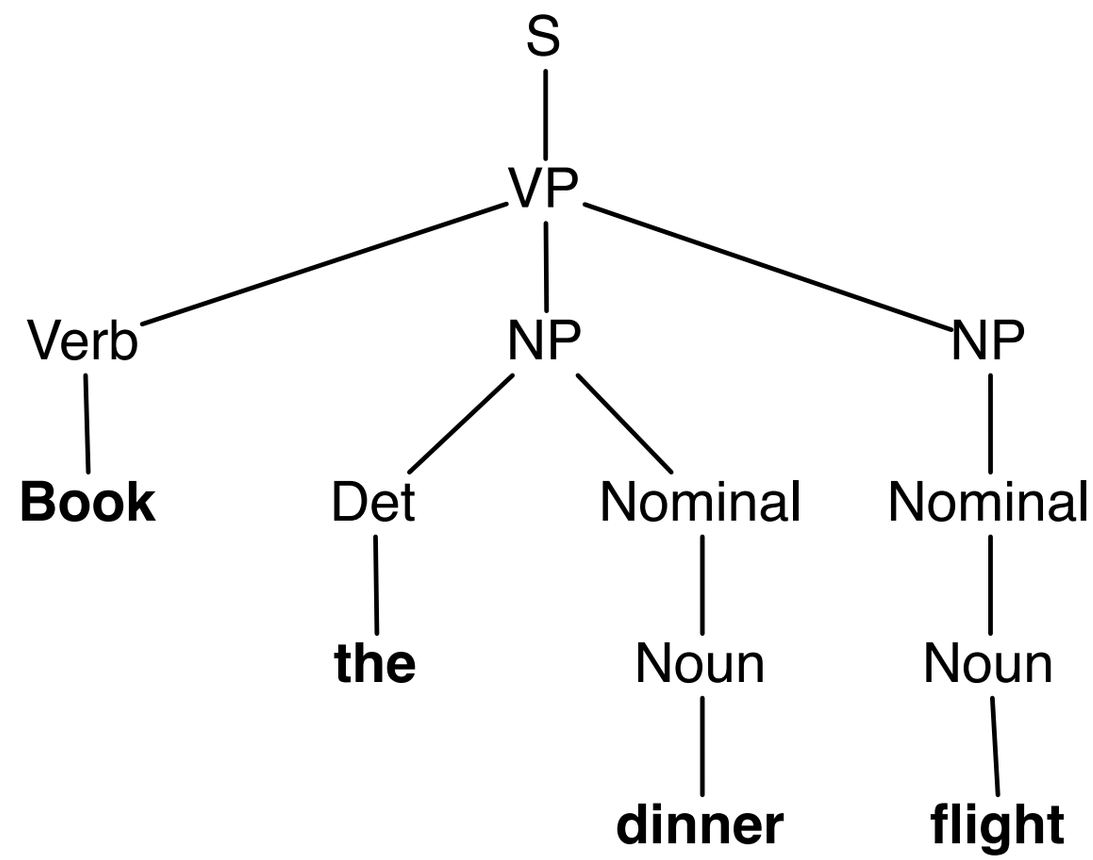
$$\hat{T}(S) = \arg\max_T \frac{P(T,S)}{P(S)}$$

$$\hat{T}(S) = \arg\max_T P(T,S)$$

$$\hat{T}(S) = \arg\max_T P(T)$$

Разрешение многозначности





P(T-left) = .05*.20*.20*.20*75*.30*.60*.10*.40=**$2.2*10^{-6}$**
P(T-right) = .05*.10*.20*.15*.75*.75*.30*.60*.10*.40=**$6.1*10^{-7}$**

Моделирование языка

$$P(S) = \sum_T P(T, S) = \sum_T P(T)$$

- Вариант 1:
  - Этап 1: с помощью n-граммной модели получить m лучших предложений
  - Этап 2: выбрать наиболее вероятное предложение на основе грамматики
- Вариант 2:
  - Модифицировать парсер для предсказания следующего слова (Xu et. al 2002)

Вероятностная версия алгоритма CKY

- Добавляем в каждую ячейку вероятность нетерминального символа
- Ячейка [i,j] должна содержать наиболее вероятный вывод, покрывающий с i+1 по j слова и содержать их вероятность.
- При трансформации грамматики к нормальной форме необходимо сохранить вероятности правил

Преобразование грамматики

Оригинальная грамматика

S → NP VP	0.8
S → Aux NP VP	0.1
S → VP	0.1
NP → Pronoun	0.2
NP → Proper-Noun	0.2
NP → Det Nominal	0.6
Nominal → Noun	0.3
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → Verb	0.2
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

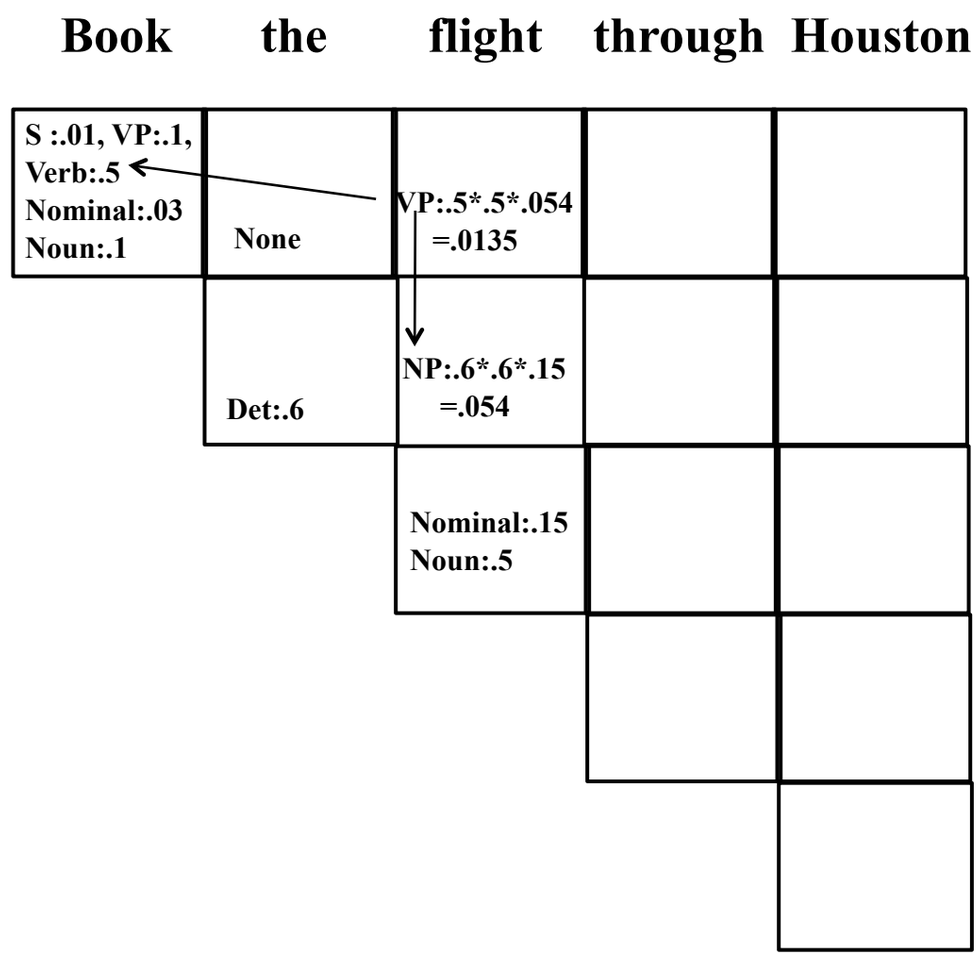
Грамматика в нормальной форме Хомского

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book | include | prefer	
0.01 0.004 0.006	
S → Verb NP	0.05
S → VP PP	0.03
NP → I | he | she | me	
0.1 0.02 0.02 0.06	
NP → Houston | NWA	
0.16 .04	
NP → Det Nominal	0.6
Nominal → book | flight | meal | money	
0.03 0.15 0.06 0.06	
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book | include | prefer	
0.1 0.04 0.06	
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

Вероятностная версия алгоритма CKY

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None			
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

Вероятностная версия алгоритма CKY



Вероятностная версия алгоритма CKY

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135		
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

Вероятностная версия алгоритма CKY

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	
			Prep:.2	

Вероятностная версия алгоритма CKY

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Вероятностная версия алгоритма CKY

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Вероятностная версия алгоритма CKY

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Вероятностная версия алгоритма CKY

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1		S:.05*.5*.054 =.00135		
	None	VP:.5*.5*.054 =.0135	None	S:.05*.5* .000864 =.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Вероятностная версия алгоритма CKY

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:.03*.0135* .032 =.00001296 S:.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Вероятностная версия алгоритма CKY

Выбираем наиболее вероятное дерево разбора

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Обучение СКС

- Вычисление вероятности на основе банка деревьев

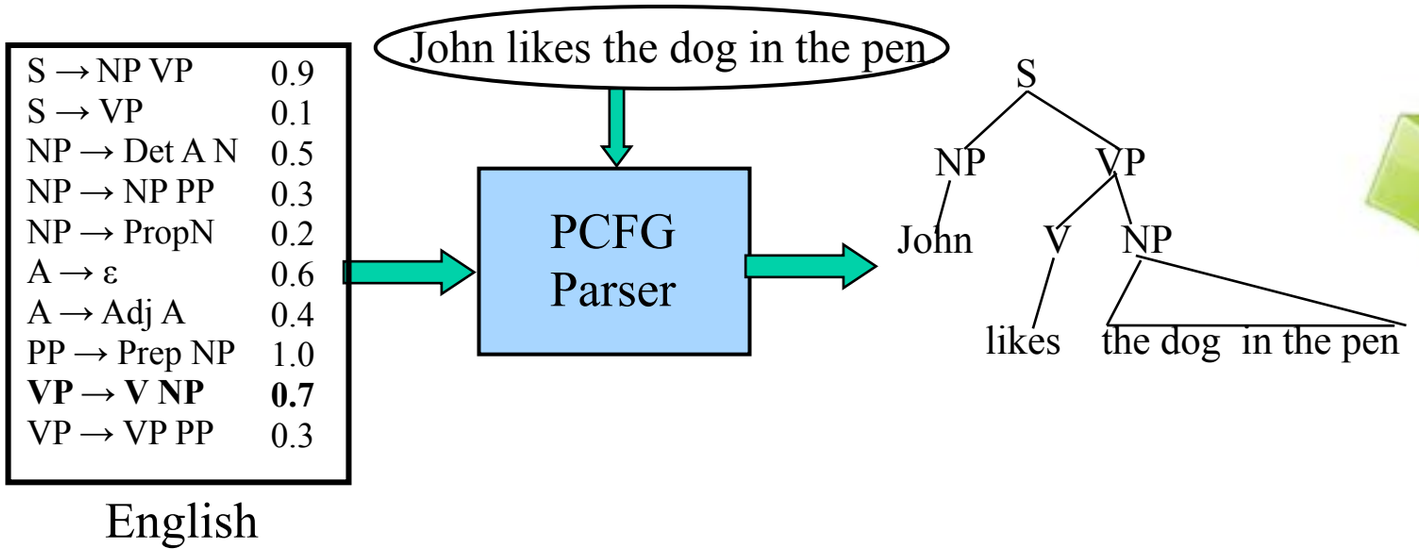
$$P(\alpha \rightarrow \beta|\alpha) = \frac{Count(\alpha \rightarrow \beta)}{\sum_{\gamma} Count(\alpha \rightarrow \beta)} = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$$

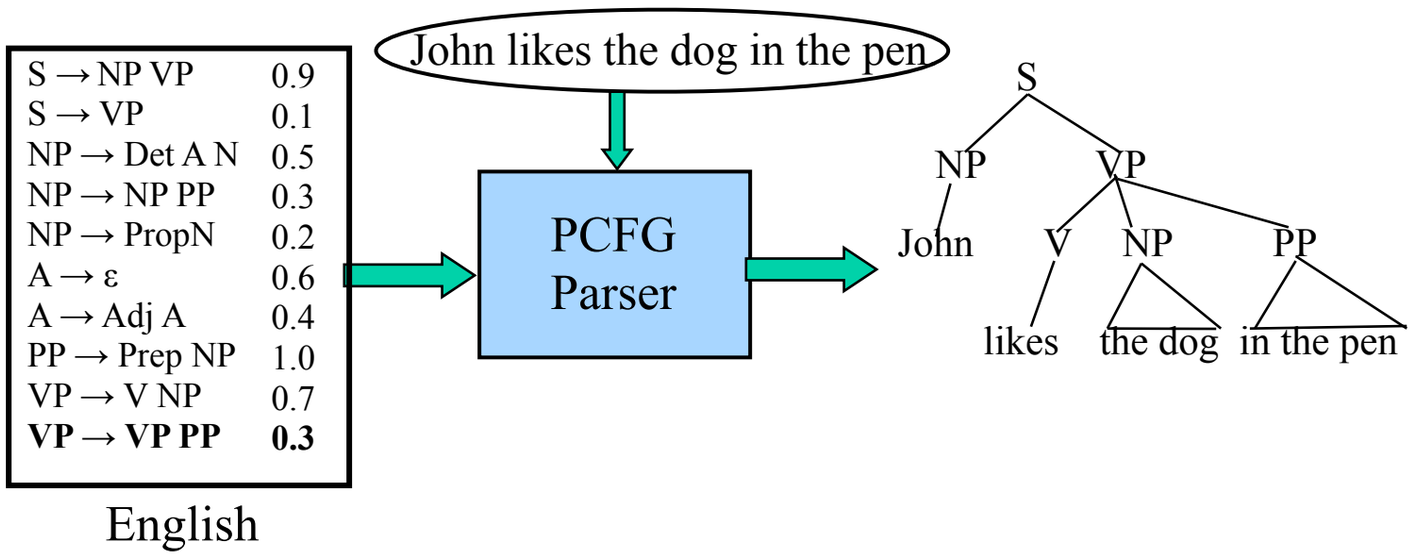
- Вывод без тренировочного множества (EM)
  - На основе множества предложений построить множество наиболее вероятных синтаксических разборов
  - Обновить значения вероятностей на основе полученных данных
  - (Manning and Schutze 1999)

Проблемы СКС

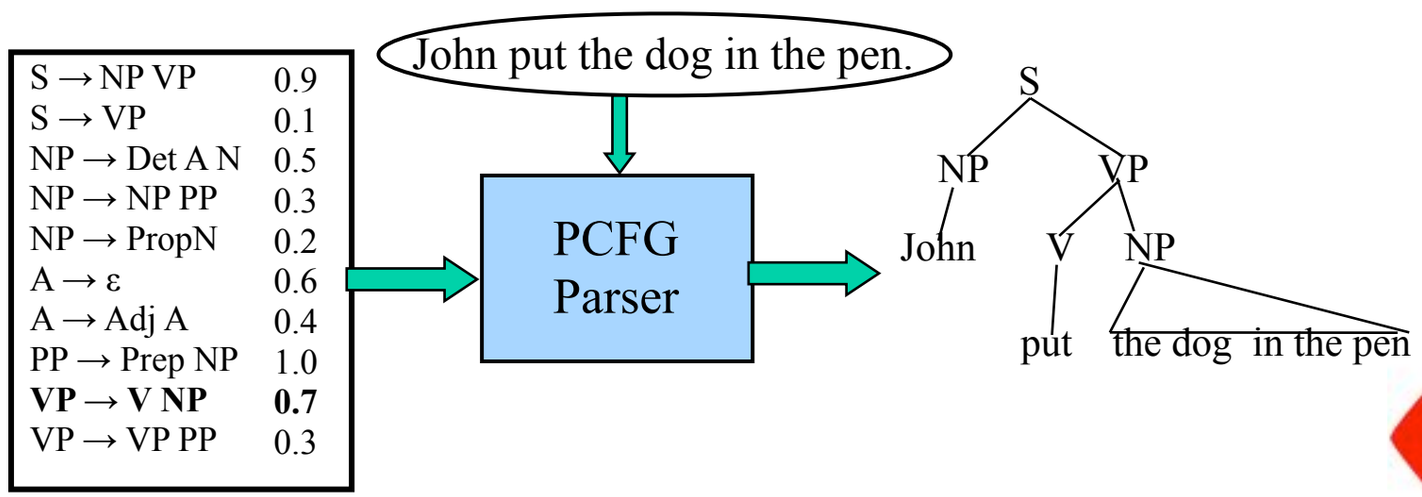
- Предположение о независимости правил, не позволяет хорошо моделировать структурные зависимости в дереве разбора
- СКС не могут моделировать синтаксические факты о конкретных словах

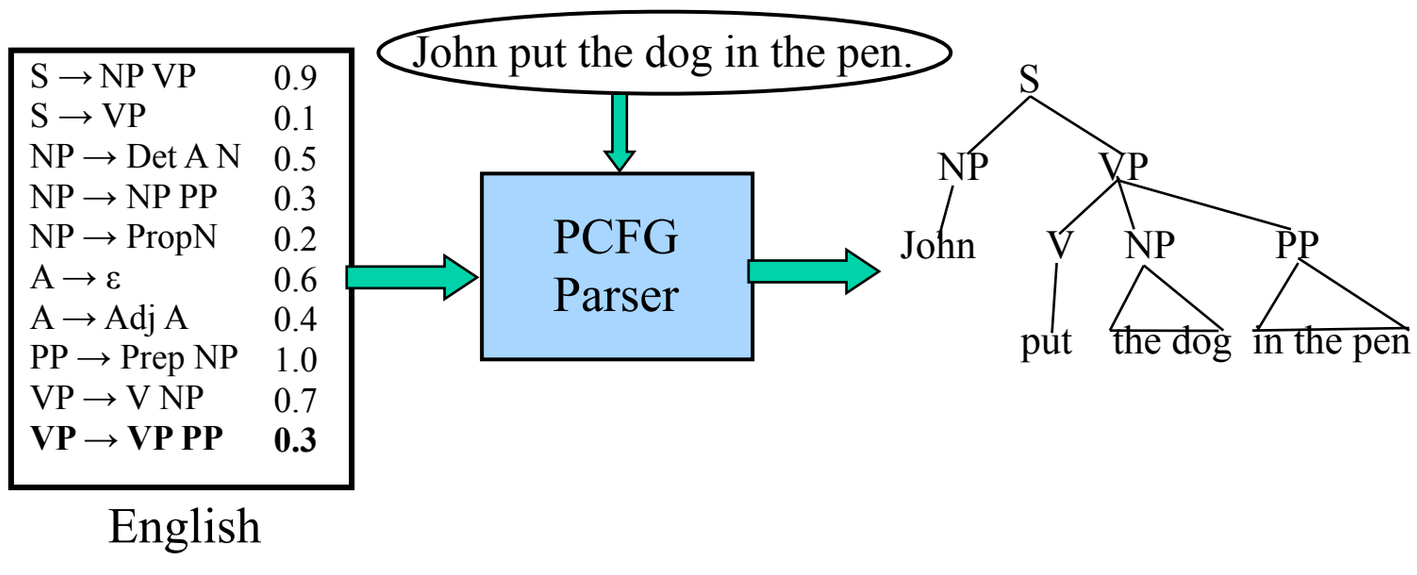
Пример


S → NP VP 0.9
S → VP 0.1
NP → Det A N 0.5
NP → NP PP 0.3
NP → PropN 0.2
A → ε 0.6
A → Adj A 0.4
PP → Prep NP 1.0
VP → V NP 0.7
VP → VP PP 0.3
English
John likes the dog in the pen
PCFG Parser
S
NP
VP
John
V
NP
likes
the dog
in the pen


S → NP VP 0.9
S → VP 0.1
NP → Det A N 0.5
NP → NP PP 0.3
NP → PropN 0.2
A → ε 0.6
A → Adj A 0.4
PP → Prep NP 1.0
VP → V NP 0.7
VP → VP PP 0.3
English
John likes the dog in the pen
PCFG Parser
S
NP
VP
John
V
NP
PP
likes
the dog
in the pen

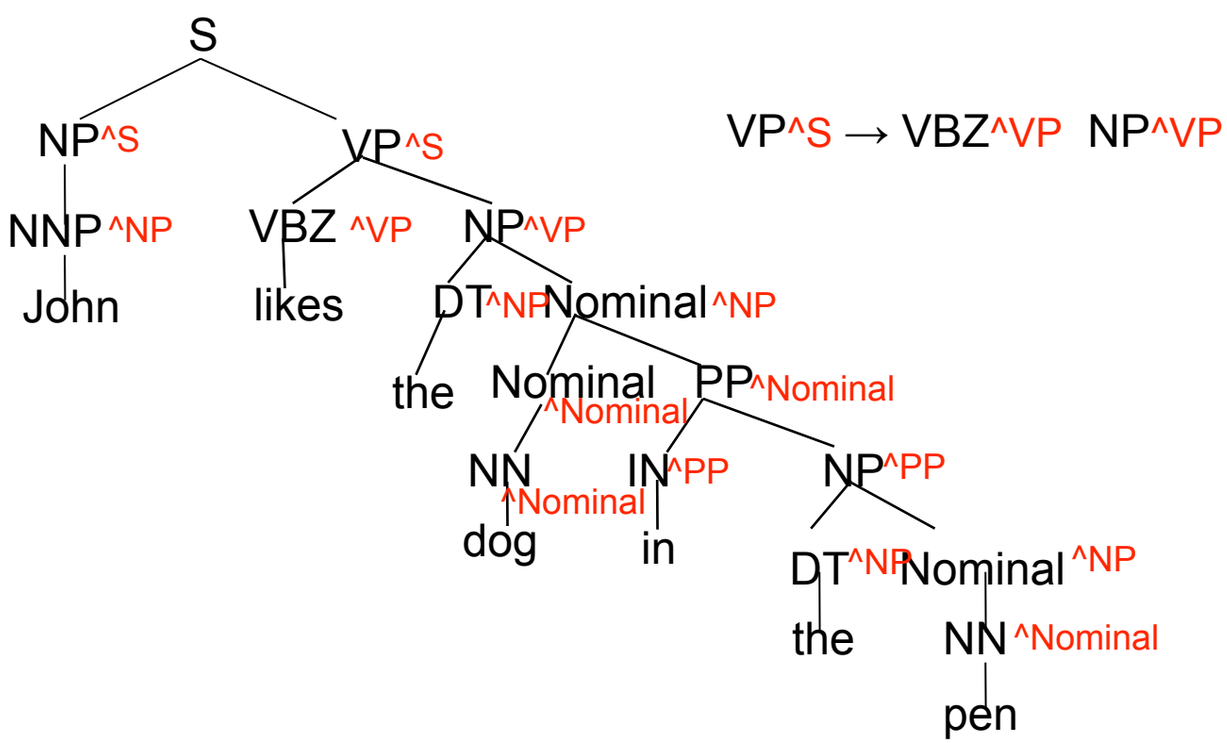
Пример


S → NP VP 0.9
S → VP 0.1
NP → Det A N 0.5
NP → NP PP 0.3
NP → PropN 0.2
A → ε 0.6
A → Adj A 0.4
PP → Prep NP 1.0
VP → V NP 0.7
VP → VP PP 0.3
English
John put the dog in the pen.
PCFG Parser
S
NP
VP
John
V
NP
put
the dog
in the pen


S → NP VP 0.9
S → VP 0.1
NP → Det A N 0.5
NP → NP PP 0.3
NP → PropN 0.2
A → ε 0.6
A → Adj A 0.4
PP → Prep NP 1.0
VP → V NP 0.7
VP → VP PP 0.3
English
John put the dog in the pen.
PCFG Parser
S
NP
VP
John
V
NP
PP
put
the dog
in the pen

Решение проблемы зависимостей

- Для добавления контекстуальной информации нетерминалы можно разделить на несколько, используя родительские узлы в дереве разбора (parent annotation)

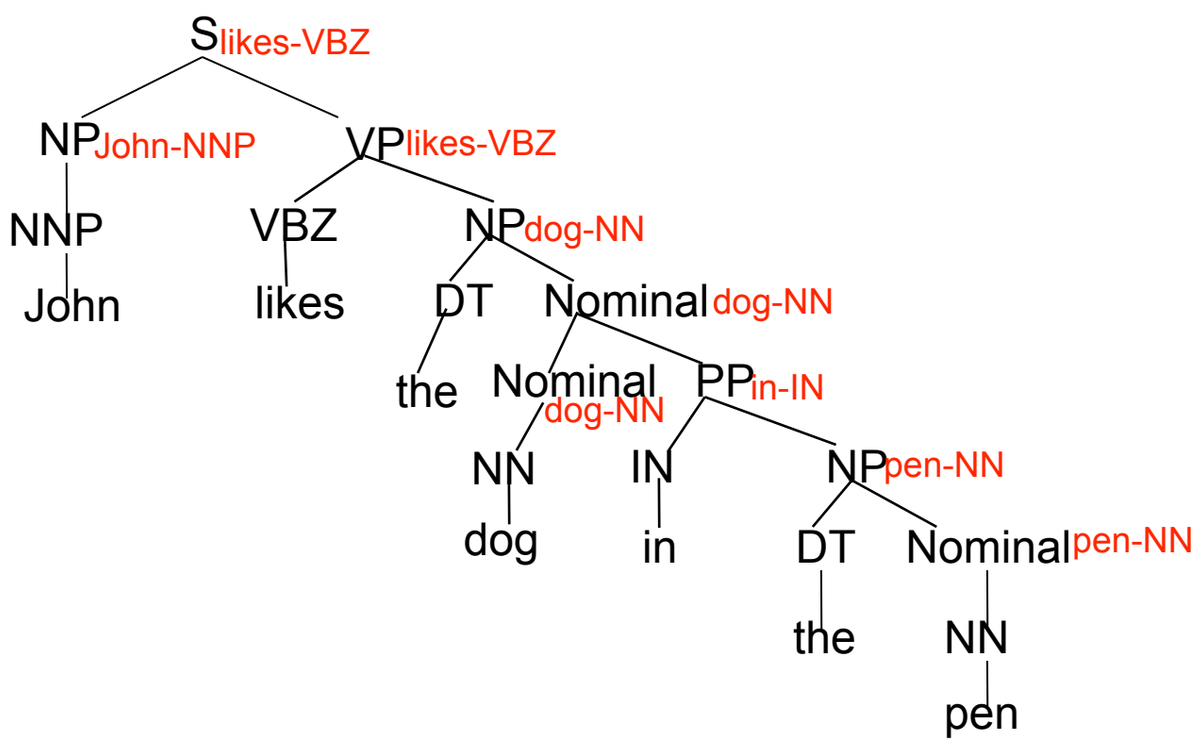


Разделение и слияние

- Разделение нетерминальных символов сильно увеличивает грамматику
- Лучше разделять нетерминалы, только если это приведет к улучшению точности
- Также можно объединять некоторые нетерминальные символы, чтобы достичь болшей точности
- Метод: эвристический поиск наилучшей комбинации разделений и слияний которая будет максимизировать правдоподобие банка деревьев

СКС с поддержкой лексики

- Расширение правил
  - VP → VP PP
  - VP (put) → VP (put) PP (in)
  - VP (put,VDB) → VP (put, VDB) PP (in,IN)



Оценка вероятности

- Точная оценка невозможна, так как не существует достаточного количества деревьев
- Необходимо сделать еще предположения (о независимости), которые помогут оценить эти вероятности
- Алгоритмы (Collins, 1999), (Charniak, 1997)

Алгоритм Коллинза

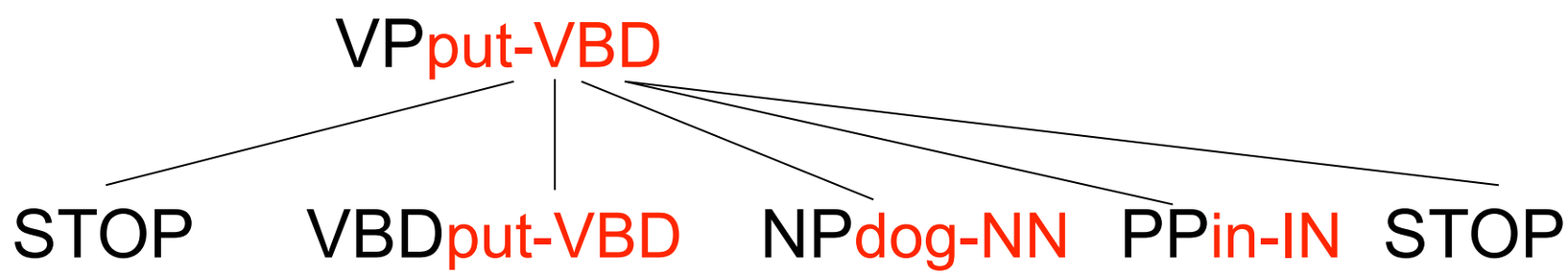
- Использует простую производящую модель
- LHS $\rightarrow L_n L_{n-1} \ldots L_1 H\ R_1 \ldots R_{m-1} R_m$
  - H-вершина группы
  - L - символы слева
  - R - символы справа
  - По краям символы STOP
- Вероятности левых и правых символов зависят только от вершины группы и нетерминала в левой части правила

Пример

VPput-VBD → VBDput-VBD NPdog-NN PPin-IN

VPput-VBD → STOP VBDput-VBD NPdog-NN PPin-IN STOP

L_1 H R_1 R_2 R_3



P(VPput-VBD → VBDput-VBD NPdog-NN PPin-IN) =

= P_H(VBDput-VBD| VPput-VBD) *

* P_L(STOP | VBDput-VBD, VPput-VBD) *

* P_R(NPdog-NN | VBDput-VBD, VPput-VBD) *

* P_R(PPin-IN | VBDput-VBD, VPput-VBD) *

* P_R(STOP | VBDput-VBD, VPput-VBD)

Оценка вероятностей

- Вероятности можно оценить на основе банка деревьев

$$P_R(PP_{\text{in-IN}} \mid VP_{\text{put-VBD}}) = \frac{\text{Count}(PP_{\text{in-IN}} \text{ справа от вершины в правиле для } VP_{\text{put-VBD}})}{\text{Count}(\text{символы справа от вершины в правиле для } VP_{\text{put-VBD}})}$$

- Сглаживание можно осуществлять через откат или линейную интерполяцию

$$\begin{aligned} smP_R(PP_{\text{in-IN}} \mid VP_{\text{put-VBD}}) = {} & \lambda_1 P_R(PP_{\text{in-IN}} \mid VP_{\text{put-VBD}}) \\ & + (1-\lambda_1)\,(\lambda_2 P_R(PP_{\text{in-IN}} \mid VP_{\text{VBD}}) + \\ & \qquad (1-\lambda_2)\, P_R(PP_{\text{in-IN}} \mid VP)) \end{aligned}$$

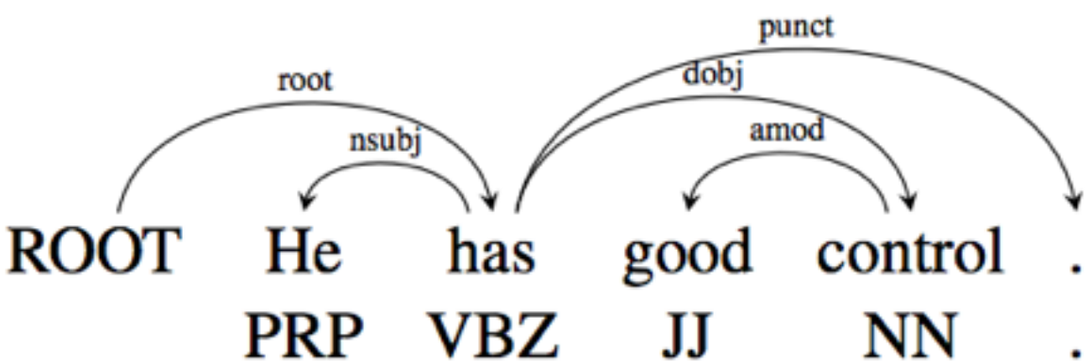
Разбор в грамматику зависимостей

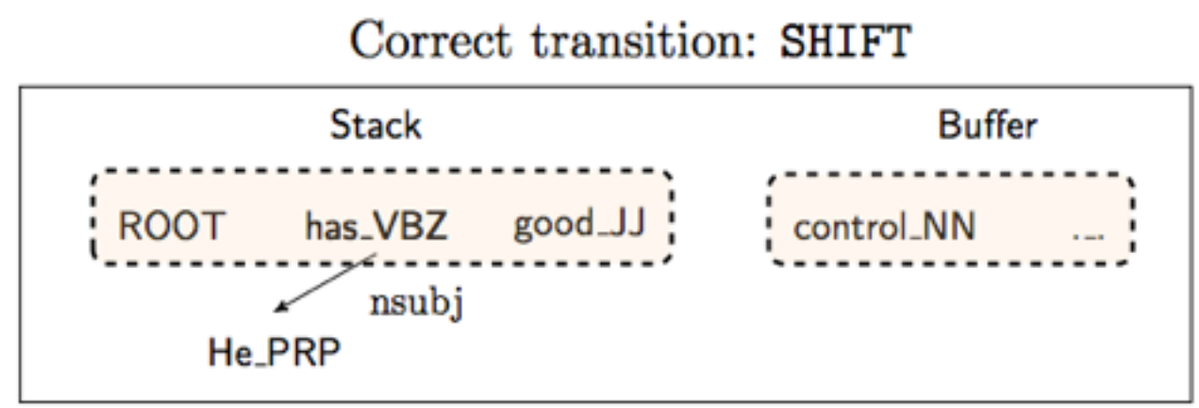
- Dependency parser
  - Malt parser (2006)
  - Stanford Neural Network Dependency Parser (2014)
- Итеративный алгоритм разбора предложения $w_1, w_2, \ldots, w_n$
- Состояние парсера $c = (s, b, A)$:
  - стек $s = [Root]$
  - буфер $b = [w_1, w_2, \ldots, w_n]$
  - множество дуг зависимостей $A = \emptyset$

Разбор в грамматику зависимостей

- Остановка
  - стек содержит один узел *ROOT*
  - и буфер пуст
- На каждой итерации происходит выбор одного из трех правил (для выбора используется классификатор)
  - *LEFT-ARC(label): добавление дуги* $s_1 \rightarrow s_2$ *с меткой label и удаление* s_2 *из стека. Предусловие:* $|s| \geq 2$
  - *RIGHT-ARC(label): добавление дуги* $s_2 \rightarrow s_1$ *с меткой label и удаление* s_1 *из стека. Предусловие:* $|s| \geq 2$
  - *SHIFT: перенос* b_1 *из буфера в стек.* $|b| \geq 1$

Пример





Transition	Stack	Buffer	A
	[ROOT]	[He has good control .]	$\emptyset$
SHIFT	[ROOT He]	[has good control .]	
SHIFT	[ROOT He has]	[good control .]	
LEFT-ARC(nsubj)	[ROOT has]	[good control .]	$A \cup$ nsubj(has,He)
SHIFT	[ROOT has good]	[control .]	
SHIFT	[ROOT has good control]	[.]	
LEFT-ARC(amod)	[ROOT has control]	[.]	$A \cup$amod(control,good)
RIGHT-ARC(dobj)	[ROOT has]	[.]	$A \cup$ dobj(has,control)
...	...	...	...
RIGHT-ARC(root)	[ROOT]	[]	$A \cup$ root(ROOT,has)

Выбор правила

- Обучение на размеченном корпусе (например, SynTagRus)
- Признаки

Часть речи t

Слово w

Элемент в стеке s_i

Самый левый $lc_1(s_i)$
и самый правый $rc_1(s_i)$
потомки s_i

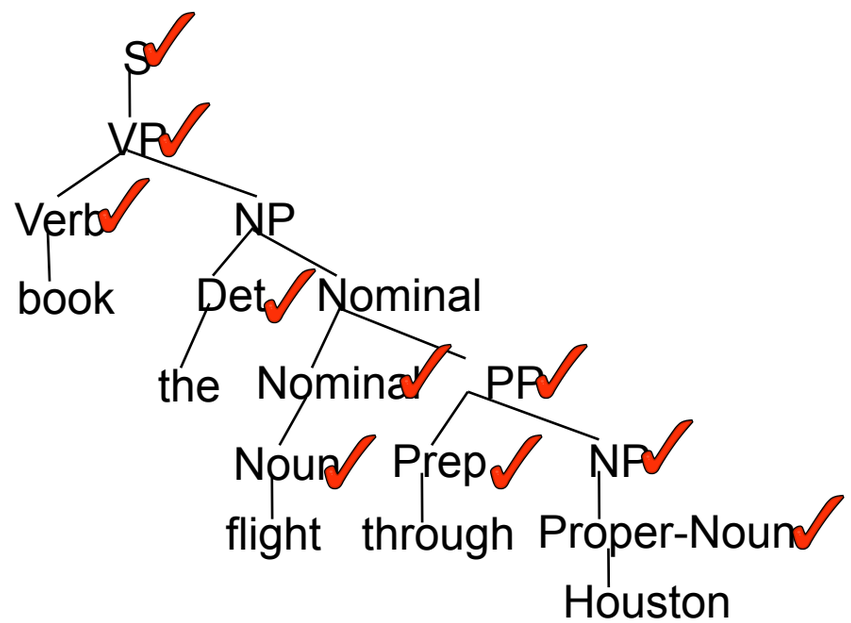
Single-word features (9)
$s_1.w$; $s_1.t$; $s_1.wt$; $s_2.w$; $s_2.t$; $s_2.wt$; $b_1.w$; $b_1.t$; $b_1.wt$
Word-pair features (8)
$s_1.wt \circ s_2.wt$; $s_1.wt \circ s_2.w$; $s_1.wts_2.t$; $s_1.w \circ s_2.wt$; $s_1.t \circ s_2.wt$; $s_1.w \circ s_2.w$ $s_1.t \circ s_2.t$; $s_1.t \circ b_1.t$
Three-word feaures (8)
$s_2.t \circ s_1.t \circ b_1.t$; $s_2.t \circ s_1.t \circ lc_1(s_1).t$; $s_2.t \circ s_1.t \circ rc_1(s_1).t$; $s_2.t \circ s_1.t \circ lc_1(s_2).t$; $s_2.t \circ s_1.t \circ rc_1(s_2).t$; $s_2.t \circ s_1.w \circ rc_1(s_2).t$; $s_2.t \circ s_1.w \circ lc_1(s_1).t$; $s_2.t \circ s_1.w \circ b_1.t$

Оценка качества алгоритма

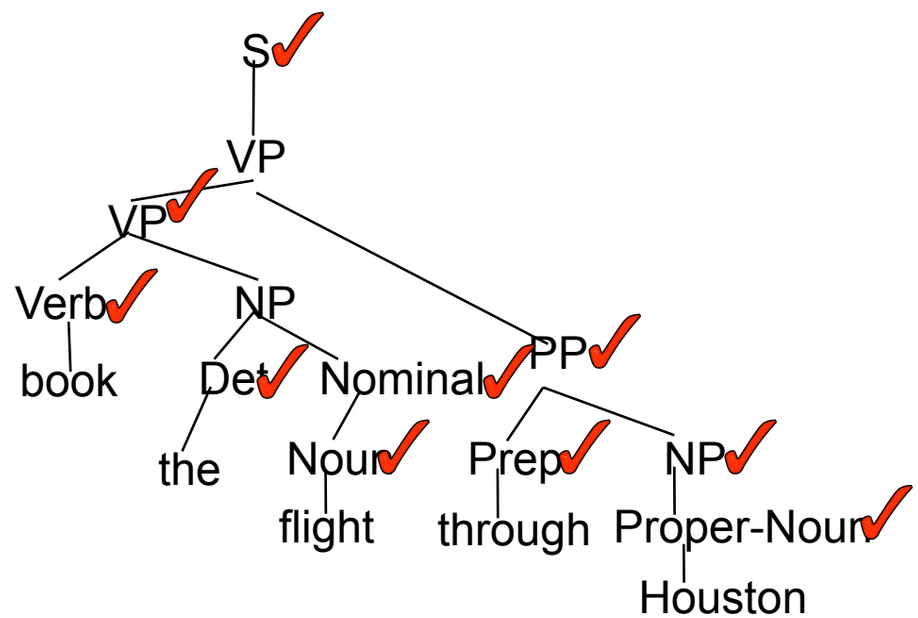
- Метрика PARSEVAL: пусть P - дерево разбора, созданное алгоритмом, T - дерево разбора, созданное экспертами
  - Точность = (# правильных компонент в P) / (# компонент в T)
  - Полнота = (# правильных компонент в P) / (# компонент в P)
  - F-мера = 2PR / (P + R)
- Современные алгоритмы показывают точность и полноту более 90%

Оценка качества алгоритма

T - дерево, размеченное вручную



P - вычисленное дерево



компонент: 12

компонент: 12

правильных компонент: 10

Точность = 10/12= 83.3%

Полнота = 10/12= 83.3%

F_1 = 83.3%

Делают ли люди синтаксический разбор?

- Психолингвистика
- Алгоритмы синтаксического разбора могут быть использованы для предсказания времени, которое потребуется человеку для прочтения каждого слова в предложении
- Чем выше вероятность слова, тем скорость чтения больше
- Для моделирования этого эффекта требуется инкрементальный алгоритм

Предложения с временной неоднозначностью

- Garden path sentence
  - Complex houses married students
  - The horse raced past the barn fell
- Инкрементальные парсеры могут найти и объяснить сложность таких предложений

Сложность языка

- Является ли естественный язык регулярным?
  - контр-пример был на прошлой лекции
- Является ли естественный язык контекстно-свободным?
  - Диалект немецкого языка в Швейцарии содержит контекстно-зависимые конструкции вида $a^n b^m c^n d^m$
- Сложность понимания людьми
  - чем проще конструкция, тем легче понимание смысла текста

Заключение

- Статистические модели, такие как СКС позволяют разрешать многозначность
- СКС может можно выучить на основе банка деревьев
- Учет лексики и разделение нетерминальных символов позволяет разрешить дополнительные неоднозначности
- Точность современных алгоритмов синтаксического разбора высока, но не достигает уровня экспертного разбора

Следующая лекция

- Лексическая семантика и разрешение лексической многозначности