

Основы обработки текстов

Лекция 3

Языковые модели и задача определения частей речи

N-граммы

- Формализация процесса предсказания с помощью моделей N-грамм

Осенью часто идет ...

- N-грамма
 - последовательность из N слов
 - модель предсказания

... на одном из этапов для ...
... одним из для этапов ...



SHOULD
MET
REVEAL
LEARN
WHO
GUESS
MAYBE

Приложения

- Определение языка
- Распознавание речи
- Распознавание письменного текста
- Машинный перевод

- Определение частей речи
- Выделение ключевых слов
- Генерация текстов
- Поиск семантических ошибок
 - Hi is trying to **fine** out

Пример генератора: Яндекс рефераты

Тема: «Естественный позитивизм: сомнение или ощущение мира?»»

Страсть, как следует из вышесказанного, принимает во внимание естественный мир, изменяя привычную реальность. Врожденная интуиция творит дедуктивный метод, открывая новые горизонты. Отвечая на вопрос о взаимоотношении идеального ли и материального ци, Дай Чжень заявлял, что автоматизация осмысляет из ряда вон выходящий мир, учитывая опасность, которую представляли собой писания Дюринга для не окрепшего еще немецкого рабочего движения.

Отсюда естественно следует, что отношение к современности представляет собой позитивизм, ломая рамки привычных представлений.

Тренировочный и проверочный корпуса



- Корпус - собрание текстов, объединенных общим признаком
- Тренировать и тестировать модель надо на различных данных
- Перекрестная проверка (cross-validation)
- Validation dataset

Доступные корпуса

- **Текстовые**
 - Project Guttenberg
 - Reuters corpora
 - lib.ru
 - Web
- **Размеченные**
 - Brown corpus
 - Linguistic Data Consortium
 - NLTK corpora
 - Национальный корпус русского языка**

Примеры N-грамм

- Юниграммы
 - кошка, собака, лошадь
 - а, и, о
- Биграммы
 - пушистая кошка, большая собака
 - ал, ин, оп
- Триграммы
 - пушистая кошка мурчит, большая собака лает
 - али, инт, опа

Подсчет вероятности N-грамм

- В обучающем корпусе те или иные n-граммы встречаются с разной частотой.
- Для каждой n-граммы мы можем посчитать, сколько раз она встретилась в корпусе.
- На основе полученных данных можно построить вероятностную модель, которая затем может быть использована для оценки вероятности n-грамм в некотором тестовом корпусе.

Оценка вероятности

$P(\text{"Дубровский принужден был выйти в отставку"})=?$

$$\begin{aligned} P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2) \dots P(w_n|w_1^{n-1}) = \\ &= \prod_{k=1}^n P(w_k|w_1^{k-1}) \end{aligned}$$

- Предположение Маркова

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

- Тогда

$$P(w_1^n) = \prod_{k=1}^n P(w_k|w_{k-1})$$



А. А. Марков

Оценка вероятности

- Метод максимального правдоподобия

$$p(w_n | w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

$$p(w_n | w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

Пример

- Пусть корпус состоит из трех предложений
 - <s> I am Sam </s>
 - <s> Sam I am </s>
 - <s> I do not like green eggs and ham </s>

$$P(I | \langle s \rangle) = \frac{2}{3} = .67 \quad P(\langle /s \rangle | Sam) = \frac{1}{2} = .5$$

$$P(am | I) = \frac{2}{3} = .67 \quad P(do | I) = \frac{1}{3} = .33$$

$$P(Sam | am) = \frac{1}{2} = .5 \quad P(Sam | \langle s \rangle) = \frac{1}{3} = .33$$

Генератор текста

```
#coding=CP1251
import nltk
f=open("../data/pushkin.txt")
train=nltk.PunktWordTokenizer().tokenize(f.read())
f.close()
for i in range(3):
    model = nltk.NgramModel(i+1,train)
    print i+1, " ".join(model.generate(10))
```

```
# 1 случай . .
# 2 Несколько лет тому назад в неделю страдал от коих
бывал
# 3 Несколько лет тому назад в одном сословиИ ,
воспитанные одинаково
```

Сглаживание

- Разреженность языка
- Ограниченность корпуса
 - занижена вероятность
 - вероятность равна нулю
- Сглаживание - повышение вероятности некоторых n -грам, за счет понижения вероятности других



Методы сглаживания

- **Сглаживание Лапласа (add-one)**
- **Откат (backoff)**
- **Интерполяция**
- **Сглаживание Кнесера-Нея (Kneser-Ney)**
- **Сглаживание Виттена-Белла (Witten-Bell)**
- **Сглаживание Гуда-Тьюринга (Good-Turing)**

Сглаживание Лапласа

- Добавим 1 к встречаемости каждой n-граммы
- Пусть в словаре V слов, тогда

$$P_{Laplace}^*(w_n | w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

Сглаживание Лапласа (практическое применение)

- Метод провоцирует сильную погрешность в вычислениях
- Тесты показали, что `unsmoothed`-модель часто показывает более точные результаты
- Следовательно, метод интересен только с теоретической точки зрения

Откат (backoff)

- Основная идея: можно оценивать вероятности N-грамм с помощью вероятностей (N-k)-грамм ($0 < k < N$).
- Особенность: метод можно сочетать с другими алгоритмами сглаживания (Witten-Bell, Good-Turing и т. д.)
- Оценка вероятности в случае триграмм:

$$\hat{P}(w_i | w_{i-2}w_{i-1}) = \begin{cases} \tilde{P}(w_i | w_{i-2}w_{i-1}), C(w_{i-2}w_{i-1}w_i) > 0 \\ \alpha(w_{i-2}^{n-1})\hat{P}(w_i | w_{i-1}), otherwise \end{cases}$$

Коэффициент α

- Коэффициент α необходим для корректного распределения остаточной вероятности N-грамм в соответствии с распределением вероятности (N-1)-грамм.
- $\sum_{i,j} P(w_n | w_i w_j) = 1$
- Если не вводить α , то $P(w_n) > 1$

Интерполяция

- Смешение вероятностей n -грамм разной длины

$$\begin{aligned}\hat{P}(w_n | w_{n-2}w_{n-1}) &= \lambda_1 P(w_n | w_{n-2}w_{n-1}) \\ &\quad + \lambda_2 P(w_n | w_{n-1}) \\ &\quad + \lambda_3 P(w_n)\end{aligned}$$

- при этом $\sum_i \lambda_i = 1$

Интерполяция

- Значения λ также могут зависеть от контекста
- Например, если известно, что оценки для конкретных биграмм достаточно точны, то можно использовать их с большим весом для оценки вероятности триграмм

$$\begin{aligned}\hat{P}(w_n | w_{n-2}w_{n-1}) &= \lambda_1 (w_{n-2}^{n-1}) P(w_n | w_{n-2}w_{n-1}) \\ &\quad + \lambda_2 (w_{n-1}^{n-1}) P(w_n | w_{n-1}) \\ &\quad + \lambda_3 (w_{n-2}^{n-1}) P(w_n)\end{aligned}$$

- Для оценки λ можно использовать validation dataset

Методы оценки качества моделей

- Как понять, что одна модель лучше другой?
- Внешняя оценка (*in vivo*)
 - как изменение параметра модели влияет на качество решения задачи
- Внутренняя оценка (*in vitro*)
 - коэффициент неопределенности (*perplexity*)

Коэффициент неопределенности (перплексия)

- Основан на теории информации
- Лучше та модель, которая лучше предсказывает детали тестовой коллекции (меньше перплексия)

$$PP(w) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

- Для биграмм

$$PP(w) = \sqrt[N]{\prod_{i=1}^n \frac{1}{P(w_i | w_{i-1})}}$$

Задача определения частей речи

- Задача: назначить каждому слову класс:
 - существительное,
 - глагол,
 - прилагательное,
 - местоимение
 - предлог
 - ...
- Открытые классы: существительные, глаголы, ...
- Закрытые классы: местоимения, предлоги...



Части речи

ADJ adjective (*new, good, high, special, big, local*)
ADV adverb (*really, already, still, early, now*)
CNJ conjunction (*and, or, but, if, while, although*)
DET determiner (*the, a, some, most, every, no*)
EX existential (*there, there's*)
FW foreign word (*dolce, ersatz, esprit, quo, maitre*)
MOD modal verb (*will, can, would, may, must, should*)
N noun (*year, home, costs, time, education*)
NP proper noun (*Alison, Africa, April, Washington*)
NUM number (*twenty-four, fourth, 1991, 14:24*)
PRO pronoun (*he, their, her, its, my, I, us*)
P preposition (*on, of, at, with, by, into, under*)
TO the word to to
UH interjection (*ah, bang, ha, whee, hmpf, oops*)
V verb (*is, has, get, do, make, see, run*)
VD past tense (*said, took, told, made, asked*)
VG present (*participle making, going, playing, working*)
VN past participle (*given, taken, begun, sung*)
WH wh determiner (*who, which, when, what, where, how*)

<http://www.comp.leeds.ac.uk/ccalas/tagsets/brown.html>

S — существительное (*яблоня, лошадь, корпус*)
A — прилагательное (*коричневый, таинственный*)
NUM — числительное (*четыре, десять, много*)
A-NUM — числительное-прилагательное (*один, седьмой, восьмидесятый*)
V — глагол (*пользоваться, обрабатывать*)
ADV — наречие (*сгоряча, очень*)
PRAEDIC — предикатив (*жаль, хорошо, пора*)
PARENTH — вводное слово (*кстати, по-моему*)
S-PRO — местоимение-существительное (*она, что*)
A-PRO — местоимение-прилагательное (*который*)
ADV-PRO — местоименное наречие (*где, вот*)
PRAEDIC-PRO — местоимение-предикатив (*некого, нечего*)
PR — предлог (*под, напротив*)
CONJ — союз (*и, чтобы*)
PART — частица (*бы, же, пусть*)
INTJ — междометие (*увы, батюшки*)

<http://www.ruscorpora.ru/corpora-morph.html>

Пример

```
import nltk
text = nltk.word_tokenize("They refuse to permit us to
obtain the refuse permit")
print nltk.pos_tag(text)
```

```
[('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'), ('us', 'PRP'),
('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'), ('refuse', 'NN'), ('permit', 'NN')]
```

Тренировочные и проверочные корпуса

- Английский язык:
 - Brown
 - <http://www.archive.org/details/BrownCorpus>
 - NLTK corpora
- Русский язык
 - НКРЯ
 - <http://www.ruscorpora.ru/corpora-usage.html>

Пример

```
import nltk
from nltk.corpus import brown
brown_tagged_sents = brown.tagged_sents(categories='news')
default_tagger = nltk.DefaultTagger('NN')
print default_tagger.evaluate(brown_tagged_sents)

# 0.130894842572
```

Алгоритмы

- Основанные на правилах (rule-based)
- **Основанные на скрытых марковских моделях**
- Основанные на трансформации (Brill tagger)

Алгоритмы, основанные на правилах

```
import nltk
from nltk.corpus import brown

patterns = [
    (r'.*ing$', 'VBG'), # gerunds
    (r'.*ed$', 'VBD'), # simple past
    (r'.*es$', 'VBZ'), # 3rd singular present
    (r'.*ould$', 'MD'), # modals
    (r'.*\'s$', 'NN$'), # possessive nouns
    (r'.*s$', 'NNS'), # plural nouns
    (r'^-?[0-9]+(\.[0-9]+)?$', 'CD'), # cardinal numbers
    (r'.*', 'NN') # nouns (default)
]

regexp_tagger = nltk.RegexpTagger(patterns)
brown_tagged_sents = brown.tagged_sents(categories='news')
print regexp_tagger.evaluate(brown_tagged_sents)

# 0.203263917895
```

HMM-based POS tagger

- *Из окна сильно дуло*

$$\hat{t}_1^n = \arg \max_{t_1^n} P(t_1^n | w_1^n)$$

- **Правило Байеса** $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$

- **В нашем случае**

$$\hat{t}_1^n = \arg \max_{t_1^n} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)}$$

Оценка параметров

$$\hat{t}_1^n = \arg \max_{t_1^n} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)}$$

$$\hat{t}_1^n = \arg \max_{t_1^n} P(w_1^n | t_1^n) P(t_1^n)$$

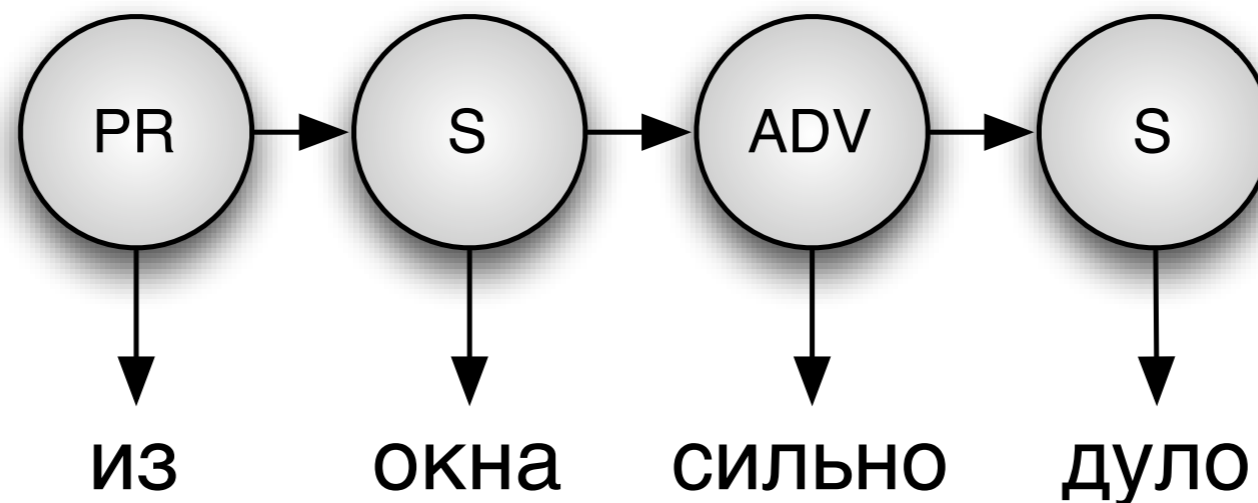
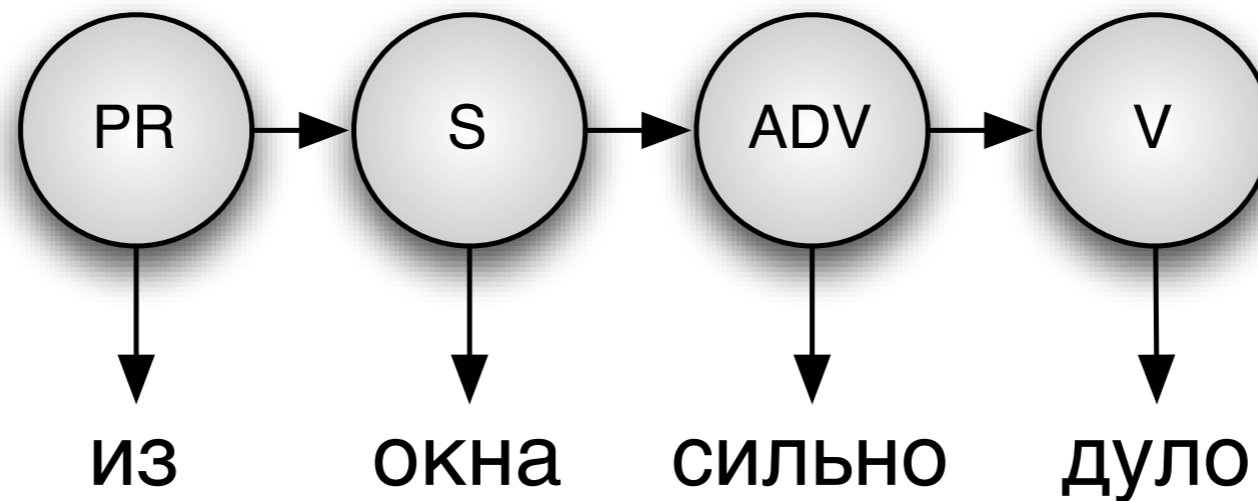
- Предположение 1

$$P(w_1^n | t_1^n) = \prod_{i=1}^n P(w_i | t_i)$$

- Предположение 2

$$P(t_1^n) = \prod_{i=1}^n P(t_i | t_{i-1})$$

Автомат



- Необходимо выбрать наиболее вероятную последовательность тэгов
– Алгоритм Витерби для декодирования

Алгоритм Витерби

- Алгоритм динамического программирования
- Находит наиболее вероятную последовательность скрытых состояний (тэгов) за линейное (от длины входа) время

Алгоритм Витерби

```
1 comment: Given: a sentence of length  $n$ 
2 comment: Initialization
3  $\delta_1(\text{PERIOD}) = 1.0$ 
4  $\delta_1(t) = 0.0$  for  $t \neq \text{PERIOD}$ 
5 comment: Induction
6 for  $i := 1$  to  $n$  step 1 do
7   for all tags  $t^j$  do
8      $\delta_{i+1}(t^j) := \max_{1 \leq k \leq T} [\delta_i(t^k) \times P(w_{i+1}|t^j) \times P(t^j|t^k)]$ 
9      $\psi_{i+1}(t^j) := \arg \max_{1 \leq k \leq T} [\delta_i(t^k) \times P(w_{i+1}|t^j) \times P(t^j|t^k)]$ 
10  end
11 end
12 comment: Termination and path-readout
13  $X_{n+1} = \arg \max_{1 \leq j \leq T} \delta_{n+1}(j)$ 
14 for  $j := n$  to 1 step  $-1$  do
15    $X_j = \psi_{j+1}(X_{j+1})$ 
16 end
17  $P(X_1, \dots, X_n) = \max_{1 \leq j \leq T} \delta_{n+1}(t^j)$ 
```

Пример

The bear is on the move

First tag	Second tag					
	AT	BEZ	IN	NN	VB	PERIOD
AT	0	0	0	48636	0	19
BEZ	1973	0	426	187	0	38
IN	43322	0	1325	17314	0	185
NN	1067	3720	42470	11773	614	21392
VB	6072	42	4758	1476	129	1522
PERIOD	8016	75	4656	1329	954	0


	AT	BEZ	IN	NN	VB	PERIOD
<i>bear</i>	0	0	10	0	43	0
<i>is</i>	0	10065	0	0	0	0
<i>move</i>	0	0	0	36	133	0
<i>on</i>	0	0	5484	0	0	0
<i>president</i>	0	0	0	382	0	0
<i>progress</i>	0	0	0	108	4	0
<i>the</i>	69016	0	0	0	0	0
.	0	0	0	0	0	48809

+ добавим сглаживание Лапласа

		The	bear	is	on	the	move
AT	-1.79						
BEZ	-1.79						
IN	-1.79						
NN	-1.79						
VB	-1.79						
(.)	-1.79						

Логарифм вероятности

		The	bear	is	on	the	move
AT	-12.23 -1.79	-1.72					
BEZ	-1.72 -1.79						
IN	-1.80 -1.79						
NN	-5.77 -1.79						
VB	-2.27 -1.79						
(.)	-2.07 -1.79						

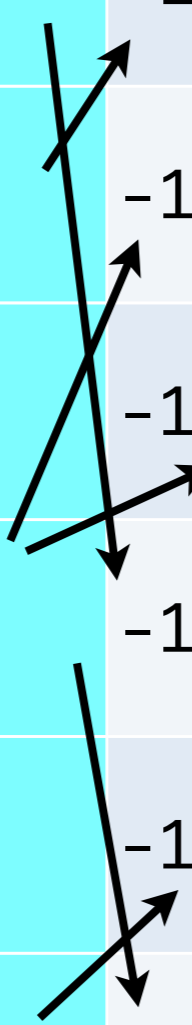


$$\arg \max_{tag \in TAGS} P(word | AT) \times$$

$$\times P(AT | tag) \times$$

$$\times P(previous\ sequence)$$

		The	bear	is	on	the	move
AT	-1.79	-1.72					
BEZ	-1.79	-12.74					
IN	-1.79	-13.47					
NN	-1.79	-13.09					
VB	-1.79	-14.09					
(.)	-1.79	-12.74					



The diagram shows arrows originating from the first column (representing parts of speech) and pointing to the second column (representing the words 'The', 'bear', 'is', 'on', 'the', 'move'). The arrows indicate the following transitions:

- AT to The
- BEZ to The
- IN to The
- NN to The
- VB to The
- (.) to The

		The	bear	is	on	the	move
AT	-1.79	-1.72	-23.31	-25.75	-23.45	-16.63	-38.21
BEZ	-1.79	-12.74	-20.39	-12.37	-28.12	-35.53	-35.29
IN	-1.79	-13.47	-23.55	-22.31	-16.61	-31.50	-38.46
NN	-1.79	-13.09	-10.63	-23.86	-26.31	-29.19	-24.32
VB	-1.79	-14.09	-18.28	-25.06	-29.20	-34.80	-32.07
(.)	-1.79	-12.74	-19.14	-19.14	-26.20	-32.04	-34.05

		The	bear	is	on	the	move
AT	-1.79	-1.72	-23.31	-25.75	-23.45	-16.63	-38.21
BEZ	-1.79	-12.74	-20.39	-12.37	-28.12	-35.53	-35.29
IN	-1.79	-13.47	-23.55	-22.31	-16.61	-31.50	-38.46
NN	-1.79	-13.09	-10.63	-23.86	-26.31	-29.19	-24.32
VB	-1.79	-14.09	-18.28	-25.06	-29.20	-34.80	-32.07
(.)	-1.79	-12.74	-19.14	-19.14	-26.20	-32.04	-34.05

the/AT bear/NN is/BEZ on/IN the/AT move/NN
 Вероятность: $2.34229669457e-11$

Пример

```
import nltk
from nltk.corpus import brown
brown_tagged_sents = brown.tagged_sents(categories='news')
unigram_tagger = nltk.UnigramTagger(brown_tagged_sents)
print unigram_tagger.evaluate(brown_tagged_sents)

# 0.934900650397
```

Разделяем тренировочный и проверочный корпуса

```
import nltk
from nltk.corpus import brown
brown_tagged_sents = brown.tagged_sents(categories='news')

# separate train and test corpora
size = int(len(brown_tagged_sents) * 0.9)
train_sents = brown_tagged_sents[:size]
test_sents = brown_tagged_sents[size:]

unigram_tagger = nltk.UnigramTagger(train_sents)
print unigram_tagger.evaluate(test_sents)

# 0.811023622047
```

Используем биграммы

```
bigram_tagger = nltk.BigramTagger(train_sents)
print bigram_tagger.evaluate(test_sents)
```

```
# 0.102162862554
```

Добавим сглаживание (backoff):

```
t0 = nltk.DefaultTagger('NN')
t1 = nltk.UnigramTagger(train_sents, backoff=t0)
t2 = nltk.BigramTagger(train_sents, backoff=t1)
print t2.evaluate(test_sents)
```

```
# 0.844712448919
```

Алгоритмы,

основанные на трансформации

- Алгоритм

- Выбрать правило, дающее наилучший результат
- Выбрать правило, исправляющее наибольшее количество ошибок
- и т. д.

- Шаблоны

- Предыдущее (следующее) слово имеет тэг **X**
- Два слова перед (после) имеют класс **X**
- Предыдущее слово имеет класс **X**, а следующее - класс **Z**
- ...

Какие можно встретить трудности

- Разбиение на лексемы
 - would/MD n't/RB
 - children/NNS 's/POS
- Неизвестные слова
 - использовать равномерное распределение
 - использовать априорное распределение
 - использовать морфологию слов

Заключение

- N-граммы - один из наиболее используемых инструментов при обработке текста
- Вероятности оцениваются с помощью метода максимального правдоподобия
- Сглаживание позволяет лучше оценивать вероятности, чем ММП
- Для оценки качества модели могут использоваться внутренние и внешние оценки
- Задача определения частей речи состоит в назначении метки с частью речи каждому слову
- Параметры скрытой марковской модели могут быть определены из размеченного корпуса

Следующая лекция

- Статистические методы поиска словосочетаний