

- Уровень символов:
 - ▶ Токенизация: разбиение текста на слова
 - ▶ Разбиение текста на предложения
- Уровень слов – морфология:
 - ▶ Разметка частей речи
 - ▶ Снятие морфологической неоднозначности
- Уровень предложений – синтаксис:
 - ▶ Синтаксический разбор
- Уровень смысла – семантика и дискурс:
 - ▶ Разрешение кореферентных связей
 - ▶ Анализ дискурсивных связей
 - ▶ Анализ семантических ролей
 - ▶ Выделение синонимов

Сколько слов в этом предложении?

“На дворе трава, на траве дрова, не руби дрова на траве двора.”

12 **токенов**: На, дворе, трава, на, траве, дрова, не, руби, дрова, на, траве, двора

8 - 9 **типов**: Н/на, дворе, трава, траве, дрова, не, руби, двора.

5 **лексем**: на, двор, трава, дрова, рубить

Токен и тип

Тип – уникальное слово из текста

Токен – тип и его позиция в тексте

N = число токенов

V – словарь (все типы)

$|V|$ = количество типов в словаре

Как связаны N и $|V|$?

Закон Ципфа

В любом достаточно большом тексте ранг типа обратно пропорционален его частоте:

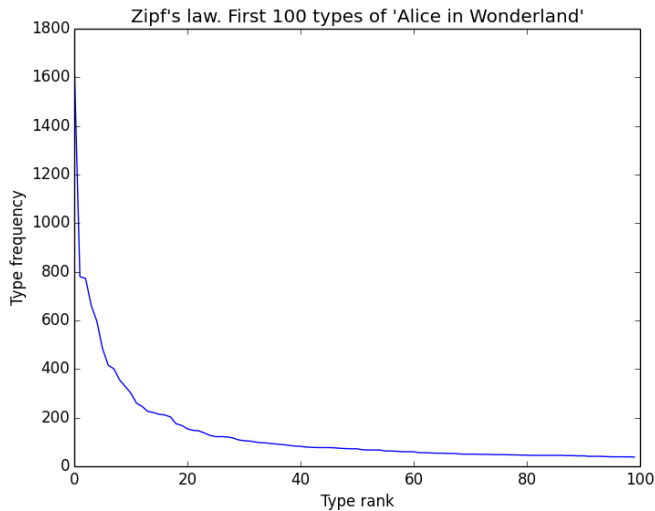
$$f = \frac{a}{r}$$

f – частота типа

r – ранг типа

a – параметр, для славянских языков – около 0.07

Закон Ципфа: пример



С увеличением длины текста (количества токенов), количество типов увеличивается в соответствии с законом :

$$|V| = K * N^b$$

N = число токенов

$|V|$ = количество типов в словаре

K, b – параметры, обычно $K \in [10, 100]$, $b \in [0.4, 0.6]$

- Разбиение текста на отдельные слова
- Сегментация предложений

Почему сложна токенизация?

- Простой пример: “В Нью-Йорке хороший маффин стоит 3.88\$”.
 - ▶ “.” – это токен?
 - ▶ 3.88\$ – это один токен или несколько?
 - ▶ Нью-Йорк – это один токен или несколько?
- В реальных данных много шума: html-разметка, ссылки, лишние знаки пунктуации.
- В реальных данных много опечаток: аптом она сказала
- Точка не всегда означает конец предложения: р. Москва, к.т.н., 20-ые гг.

Токенизация с помощью регулярных выражений

Задаем шаблоны, описывающие токены.

Регулярные выражения на Python

```
In[1]: import re
In[2]: prog = re.compile('[A-Za-z]+')
In[3]: prog.findall("Words, words, words.")
Out[1]: ['Words', 'words', 'words']
```


Сегментация предложений (1)

Как определить границы предложения?

- “?”, “!” как правило, однозначны
- Точка “.” не всегда означает конец предложения
- Прямая речь: — Кто там? — спросил дядя Фёдор. — Это я!

Бинарный классификатор для сегментации предложений: для каждой точки “.” определить, является ли она концом предложения или нет.

Бинарный классификатор

Бинарный классификатор $f : X \Rightarrow 0, 1$ получает на вход таблицу X (каждая строчка – одна точка в тексте, столбцы – признаки) и решает EndOfSentence (0) или NotEndOfSentence (1).

Сегментация предложений (2)

Какие признаки использовать для классификации?

- Количество пробелов после точки
- Заглавная или строчная буква после точки
- Принадлежит ли точка к аббревиатуре
- и т.д.

Нужно много разметки!

Natural Language Toolkit

Natural Language Toolkit умеет все

NLTK tokenizers

```
In[1]: from nltk.tokenize import RegexpTokenizer,  
wordpunct_tokenize
```

```
In[2]: s = 'Good muffins cost $3.88 in New York. \n Please  
buy me two of them. \n Thanks.'
```

```
In[3]: tokenizer = RegexpTokenizer('\w+| \$ [\d \.]+ | S  
\+')
```

```
In[4]: tokenizer.tokenize(s)
```

```
Out[1]: ['Good', 'muffins', 'cost', '$3.88', 'in', 'New',  
'York', '.', 'Please', 'buy', 'me', 'two', 'of', 'them',  
'.', 'Thanks', '.']
```

```
In[5]: wordpunct_tokenize(s)
```

```
Out[2]: ['Good', 'muffins', 'cost', '$', '3', '.', '88',  
'in', 'New', 'York', '.', 'Please', 'buy', 'me', 'two',  
'of', 'them', '.', 'Thanks', '.']
```

Обучение токенизации

`nltk.tokenize.punkt` – это инструмент для обучения токенизации по размеченным данным. Содержит обученный токенизатор `Punkt_tokenizer` для текстов на английском языке.

`Punkt_tokenizer`

```
In[1]: import nltk.data
In[2]: sent_detector =
nltk.data.load('tokenizers/punkt/english.pickle')
In[3]: sent_detector.tokenize(s)
Out[1]: ['Good muffins cost $3.88 in New York.', 'Please
buy me two of them.', 'Thanks.']
```

Задачи морфологического анализа:

- **Разбор слова** — определение нормальной формы (леммы) и грамматических характеристик слова
- **Синтез слова** — генерация слова по заданным грамматическим характеристикам

Морфологический процессор – инструмент морфологического анализа:

- Морфологический словарь
- Морфологический анализатор

Морфологический анализ

У каждого слова есть **лемма** (нормальная форма):

- кошке, кошку, кошкам, кошкой \Rightarrow кошка
- бежал, бежит, бегу \Rightarrow бежать
- белому, белым, белыми \Rightarrow белый

Часть речи [Manning, Schuetze, 1999]

Слова можно объединить в классы – части речи – в соответствии с их синтаксическими характеристиками. Основные части речи: существительные, предлоги и глаголы. Основные морфологические процессы: словоизменение (спряжение и склонение), словообразование (клуб – клубный), словосложение (земле+устройство).

Словоизменительная парадигма — список словоформ, принадлежащих одной лексеме и имеющих разные грамматические значения.

пальто-	плакать	рук-а
	плач-у	рук-и
	плач-ешь	рук-е
	плач-ет	рук-у
	плач-ем	рук-ой
	плач-ете	о рук-е
	плач-ут	

Грамматические характеристики [Попов, 1982]

- Существительное: род (м.р., ж.р., с.р.), число (ед., мн.), падеж (им., род., дат., вин., твор., пред.), одушевленность (од., неод.)
- Полное прилагательное и причастие: пассивность (пасс., акт.), время (прош., наст.), род (м.р., ж.р., с.р.), число (ед., мн.), падеж (им., род., дат., вин., твор., пред.), одушевленность (од., неод.), вид (сов., несов.)
- Краткое прилагательное и причастие: пассивность (пасс., акт.), время (прош., наст.), род (м.р., ж.р., с.р.), число (ед., мн.)
- Глагол: пассивность (пасс., акт.), время (прош., наст.) род (м.р., ж.р., с.р.), число (ед., мн.)
- Деепричастие: пассивность (пасс., акт.), время (прош., наст.), число (ед., мн.)
- Наречие: обстоятельственное (обст.), определительное (опред.)
- Количественное числительное: тип "1", "2", "5", дробное (дроб.), неопределенное (неопр.), именнованное (именнов.),
- Местоимение: класс: притяжательное (прит.), указательное (указ.), возвратное (возвр.), возвратно-аттрибутивное (возвр.-атр.), третьего лица (3 л.); род (м.р., ж.р., с.р.), число (ед., мн.), падеж (им., род., дат., вин., твор., пред.)
- Союз: сочинительный (соч.), подчинительный (подч.)
- Предлог: падеж (род., дат., вин., твор., пред.)
- Частица: вопросительная (вопр.), отрицательная (отр.)

Типы существительных [Попов, 1982]

морфологические типы существительных

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Им. ед.	Ø	Ø	Ø	а	я, ъя	я	я	ь, й	ь, й	й	о	о	я	е	е, ъе	е
Род. ед.	а	а	а	ы, и	и, ъи	и	и	и	я	я	а	а	и	а	и, ъи	я
Дат. ед.	у	у	у	е	е, ъе	е	и	и	ю	ю	у	у	и	у	ю, ъю	ю
Вин. ед.	ом, ом	ом	ом	у	ю, ъю	ю	ю	ь	ю	ю	ом	ом	ом	ом	ем, ъем	ом
Тв. ед.	ом, ом	ом	ом	ей, ой	ей, ъей	ей	ей	ю	ом	ом	ом	ом	ом	ем	ем, ъем	ом
Предл. ед.	е	е	е	е	е, ъе	е	и	и	е	и	е	е	и	е	е, ъе	и
Им. мн.	ы, и, ъя	ы, и	а	ы, и	и, ъи	и	и	и	и	и	а	и, ъи	а	а	и, ъе	я
Род. мн.	ов, ев, ей, ъев	Ø	ов	Ø	ей, ъей	й, ъ, Ø	й	ей	ей, ев	ев	Ø, ов	Ø, ов, ъев	Ø	Ø, ев	ий, ей	й
Дат. мн.	ам, ъам	ам	ам	ам	ям, ъям	ям	ям	ям	ям	ям	ам	ам, ъам	ам	ам	ям, ъям	ям
Вин. мн.	амн, ъамн	амн	амн	амн	ямн, ъямн	ямн	ямн	ямн	ямн	ямн	амн	ямн, ъамн	амн	амн	ямн, ъямн	ямн
Тв. мн.	амн, ъамн	амн	амн	амн	ямн, ъямн	ямн	ямн	ямн	ямн	ямн	амн	ямн, ъамн	амн	амн	ямн, ъямн	ямн
Предл. мн.	ах, ъах	ах	ах	ах	ях, ъях	ях	ях	ях	ях	ях	ах	ях, ъях	ах	ах	ях, ъях	ях
Примеры	Процесс, брат	Грамм	Номер	Матрица	Ступень, статья	Идея	Линия	Машень	Забой, уголь	Салатный	Тело	Дерев	Время	Полотенце	Поле, устье	Ванна

Слова состоят из морфем: *word = stem + affixes*. Стемминг позволяет отбросить аффиксы. Чаще всего используется алгоритм Портера.

1-ый вид ошибки: белый, белка, белье \Rightarrow бел

2-ой вид ошибки: трудность, трудный \Rightarrow трудност, труд, кротость \Rightarrow кротост, крот

3-ий вид ошибки: быстрый, быстрее \Rightarrow быст, побыстрее \Rightarrow побыст

Алгоритм Портера состоит из 5 циклов команд, на каждом цикле – операция удаления / замены суффикса. Возможны вероятностные расширения алгоритма.

Разрешение морфологической неоднозначности

- Существительное или глагол: стали, стекло, течь, белила, падали
- Прилагательное или существительное: мороженое, простой
- Существительное или существительное: черепах

N-граммные морфологические анализаторы:

- unigram tagging: выбирает самый частый / вероятный разбор
- ngram tagging: анализирует контекст текущего слова – *n* предыдущих слов (HMM, CRF, нужно много данных для обучения)
- Временные затраты VS точность разбора VS количество неоднозначных слов \Rightarrow ансамбли морфологических анализаторов

Классификация методов извлечения ключевых слов и словосочетаний

Ключевые слова и словосочетания сложно определить формально. Поскольку определений ключевых слов и словосочетаний множество, существует масса методов их извлечения:

- с учителем VS без учителя
- частотные VS по-сложнее
- из одного текста VS из коллекции текстов
- слова (униграммы) VS биграммы VS N -граммы
- термины VS именованные сущности VS коллокации
- последовательные слова VS с использованием окна

Методы извлечения ключевых слов и словосочетаний с учителем

Построим бинарный классификатор с целевым признаком KeyWord (1) NotKeyWord (0). Возможные признаки для классификации:

- Слово употребляется в начале предложения
- Слово написано с заглавной буквы
- Частота слова
- Слово используется в качестве названия статьи или категории в Википедии
- Слово – именованная сущность
- Слово – термин
- И т.д.

Но нужны размеченные коллекции текстов.

Методы извлечения ключевых слов и словосочетаний без учителя

- Морфологические шаблоны
- Меры ассоциации биграмм: PMI, T-Score, LLR
- Графовые методы: TextRank [Mihalcea, Tarau, 2004]
- Синтаксические шаблоны

Меры ассоциации биграмм (1)

w_1, w_2 – два слова

$f(w_1), f(w_2)$ – их частоты

$f(w_1, w_2)$ – частота биграммы $w_1 w_2$

$$PMI(w_1, w_2) = \log \frac{f(w_1, w_2)}{f(w_1)f(w_2)}$$

Pointwise Mutual Information [Manning, Shuetze, 1999]

$PMI(w_1, w_2)$ показывает сколько информации о появлении одного слова содержится в появлении другого слова.

Меры ассоциации биграмм (2)

w_1, w_2 – два слова

$f(w_1), f(w_2)$ – их частоты

$f(w_1, w_2)$ – частота биграммы $w_1 w_2$

N – число слов

$$T - score(w_1, w_2) = \frac{f(w_1, w_2) - f(w_1) * f(w_2)}{f(w_1, w_2) / N}$$

T-Score [Manning, Shuetze, 1999]

$T - score(w_1, w_2)$ – это статистический $t - test$, используемый для извлечения ключевых словосочетаний. $t - test$ измеряет разницу между ожидаемым и наблюдаемым средним значением, нормированную стандартным отклонением. Считается лучшей мерой ассоциацией биграмм. Уровень значимости при анализе, как правило, не используется.

Меры ассоциации биграмм (3)

w_1, w_2 – два слова

$O_{11} = f(w_1, w_2)$ – **наблюдаемая** частота биграммы $w_1 w_2$

$E_{11} = \frac{O_{11} + O_{12}}{N} \times \frac{O_{11} + O_{21}}{N} \times N$ – **ожидаемая** частота биграммы $w_1 w_2$

	w_2	not w_2
w_1	O_{11}	O_{12}
not w_1	O_{21}	O_{22}

$$\begin{aligned} \chi^2 &= \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} = |i \in 1, 2, j \in 1, 2| = \\ &= \frac{N(O_{11}O_{22} - O_{12}O_{21})^2}{(O_{11} + O_{12})(O_{11} + O_{21})(O_{12} + O_{22})(O_{21} + O_{22})} \end{aligned}$$

Меры ассоциации биграмм в NLTK

NLTK BigramCollocationFinder

```
In[1]: from nltk.collocations import *  
In[2]: bigram_measures =  
nltk.collocations.BigramAssocMeasures()  
In[3]: finder = BigramCollocationFinder.from_words(tokens)  
In[4]: finder.apply_freq_filter(3)  
In[5]: for i in finder.nbest(bigram_measures.pmi, 20): ...
```

Меры ассоциации биграмм в NLTK:

- `bigram_measures.pmi`
- `bigram_measures.student_t`
- `bigram_measures.chi_sq`
- `bigram_measures.likelihood_ratio`

TextRank: использование мер центральности графов для извлечения ключевых слов и словосочетаний (1)

[Mihalcea, Tarau, 2004]

- 1 Вершины графа: слова
- 2 Ребра графа могут определяться по следующим правилам:
 - ▶ Последовательные слова
 - ▶ Слова внутри левого или правого окна в $\pm 2-5$ слов;Ребра могут быть взвешенные или невзвешенные, направленные или ненаправленные.
- 3 Любая мера центральности графа используется для определения важности вершин в графе. Слова, соответствующие наиболее важным вершинам, считаются ключевыми.
- 4 Если две соседние вершины оказываются важными, соответствующие им слова формируют ключевое словосочетание.

http:

//web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf

TextRank (2)

Compatibility of systems of linear constraints over the set of natural numbers. Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the

600 ↗

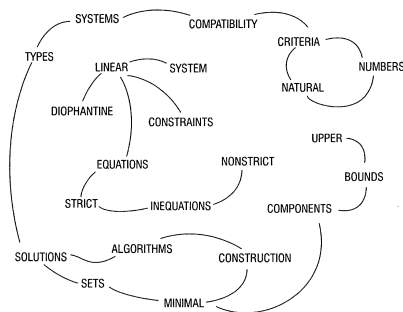


FIG. 6

TextRank (3)

$G = (V, E)$ – граф, V – вершины, E – ребра

$In(V_i)$ – множество исходящих ребер

$Out(V_i)$ – множество входящих ребер

Меры центральности

PageRank [Brin, Page, 1998]

$$PR(V_i) = (1 - d) + d \times \sum_{V_j \in In(V_i)} \frac{PR(V_j)}{|Out(V_j)|}$$

HITS [Kleinberg, 1999]

$$HITS_A(V_i) = \sum_{V_j \in In(V_i)} HITS_H(V_j)$$

$$HITS_H(V_i) = \sum_{V_j \in Out(V_i)} HITS_A(V_j)$$

TextRank (4)

Найденные ключевые слова и словосочетания

linear constraints; linear diophantine equations; natural numbers; nonstrict inequations; strict inequations; upper bounds

Меры контрастности для извлечения ключевых слов и словосочетаний

Рассмотрим некую коллекцию текстов. Требуется для данного текста определить слова и словосочетания, которые встречаются в нем **существенно** чаще, чем в других текстах.

Частота термина [Luhn, 1957]

Важность термина в тексте пропорциональная его частоте.

Обратная документная частота [Spaerck Jones, 1972]

Специфичность термина в тексте обратнопропорциональна числу текстов, в которых терм встречается.

$$tfidf(term, text, collection) = tf(term, document) \times idf(term, collection)$$

Используемые *TF* и *IDF* веса

$tf(term, document) =$

- бинарный вес: 1, если терм встречается в тексте, 0, иначе
- частота: $f_{t,d}$
- нормированная частота: $\log(1 + f_{t,d})$

$idf(term, collection) =$

- унарный вес: 1
- обратная частота: $\log \frac{N}{n_t}$, где N – число текстов в коллекции, n_t – число текстов, в которых встречается терм t
- сглаженная обратная частота: $\log \frac{N}{n_t+1}$

Самая популярная комбинация весов: $f_{t,d} \times \log \frac{N}{n_t+1}$

Использование меры $tf - idf$ для измерения сходства текстов

Пусть каждый текст d_i из коллекции текстов D представлен вектором $d_i = (w_1^i, w_2^i, \dots, w_{|V|}^i)$, где V – это словарь, $|V|$ – общее число термов (типов) в словаре. Получается векторное пространство размерности $|V|$, где каждое измерение задано одним термом. w_k^i – это вес термина k в тексте i – чаще всего, вычисленный по мере $tf - idf$. Сходство между двумя текстами может быть определено как косинус между соответствующими векторами.

Косинусная мера близости в векторном модели

Cosine similarity in Vector Space Model (VSM) [Salton et. al, 1975]

$$\cos(d_i, d_j) = \frac{d_i \times d_j}{||d_i|| ||d_j||} = \frac{\sum_k w_k^i \times w_k^j}{\sqrt{(\sum_k w_k^i)^2} \sqrt{(\sum_k w_k^j)^2}}$$

Векторная модель коллекции текстов

