



# ТЕХНОСФЕРА

## Лекция 7

# Классификация текстов и Naive Bayes

Николай Анохин

3 ноября 2015 г.

# План занятия

Обработка текстов

Naive Bayes

## Обработка текстов



# Data Mining vs Text Mining

Data Mining:

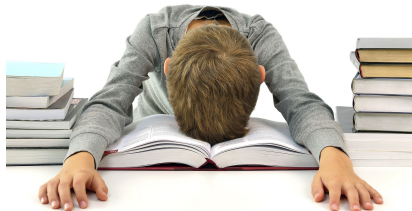
извлечение *неочевидной* информации

Text Mining:

извлечение *очевидной* информации

Трудности

- ▶ Огромные объемы
- ▶ Отсутствие структуры



# Задачи Text Mining

- ▶ Суммаризация текста  
*агрегация новостей*
- ▶ Классификация и кластеризация документов  
*категоризация, антиспам, sentiment analysis, opinion mining*
- ▶ Извлечение метаданных  
*определение языка, автора, тегирование*
- ▶ Выделение сущностей  
*места, люди, компании, почтовые адреса*

# Этапы (простой) обработки текста

Коллекция документов



Декодирование

Токенизация

Удаление  
стоп-слов

Нормализация

Стемминг

Обучающая  
выборка



# Декодирование

## Def.

перевод последовательности байт в последовательность символов

- ▶ Распаковка

*plain/.zip/.gz/...*

- ▶ Кодировка

*ASCII/utf-8/Windows-1251/...*

- ▶ Формат

*csv/xml/json/doc...*

Кроме того: что такое документ?

# Разбиение на токены

## Def.

разбиение последовательности символов на части (токены), возможно, исключая из рассмотрения некоторые символы

*Наивный подход:* разделить строку пробелами и выкинуть знаки препинания

*Трисия любила **Нью-Йорк**, поскольку любовь к Нью-Йорку могла положительно повлиять на ее карьеру.*

*Проблемы:*

- ▶ n.anokhin@corp.mail.ru, 127.0.0.1
- ▶ C++, C#
- ▶ York University vs New York University
- ▶ Зависимость от языка (“Lebensversicherungsgesellschaftsangestellter”, “l’amour”)

*Альтернатива:* n-граммы



## Разбиение на токены

```
>>> from nltk.tokenize import RegexpTokenizer
>>> tokenizer = RegexpTokenizer('\w+|[\^\w\s]+')
>>> s = u'Трисия любила Нью-Йорк, поскольку любовь \
... к Нью-Йорку могла положительно повлиять на ее карьеру.'
>>> for t in tokenizer.tokenize(s)[:7]: print t + " ::",
...
Трисия :: любила :: Нью :: - :: Йорк :: , :: поскольку ::
```

# Стоп-слова

Def.

Наиболее частые слова в языке, не содержащие никакой информации о содержании текста

```
>>> from nltk.corpus import stopwords
>>> for sw in stopwords.words('russian')[1:20]: print sw,
...
в во не что он на я с со как а то все она так его но да ты
```

Проблема: "To be or not to be"

# Нормализация

## Def.

Приведение токенов к единому виду для того, чтобы избавиться от поверхностной разницы в написании

## Подходы

- ▶ сформулировать набор правил, по которым преобразуется токен

*Нью-Йорк → нью-йорк → ньюйорк → ньюиорк*

- ▶ явно хранить связи между токенами (WordNet – Princeton)

*машина → автомобиль, Windows ↗ window*

# Нормализация

```
>>> s = u'Нью-Йорк'
>>> s1 = s.lower()
>>> print s1
нью-йорк
>>> s2 = re.sub(ur"\W", "", s1, flags=re.U)
>>> print s2
ньюйорк
>>> s3 = re.sub(ur"й", u"и", s2, flags=re.U)
>>> print s3
ньюиорк
```

# Стемминг и Лемматизация

## Def.

Приведение грамматических форм слова и однокоренных слов к единой основе (lemma):

- ▶ Stemming – с помощью простых эвристических правил
  - ▶ Porter (Cambridge – 1980)  
5 этапов, на каждом применяется набор правил, таких как

$sses \rightarrow ss$  (caresses  $\rightarrow$  caress)

$ies \rightarrow i$  (ponies  $\rightarrow$  poni)

- ▶ Lovins (1968)
  - ▶ Paice (1990)
  - ▶ еще 100500
- ▶ Lemmatization – с использованием словарей и морфологического анализа

# Стемминг

```
>>> from nltk.stem.snowball import PorterStemmer
>>> s = PorterStemmer()
>>> print s.stem('tokenization'); print s.stem('stemming')
token
stem
>>> from nltk.stem.snowball import RussianStemmer
>>> r = RussianStemmer()
>>> print r.stem(u'Авиация'); print r.stem(u'национальный')
авиац
национальн
```

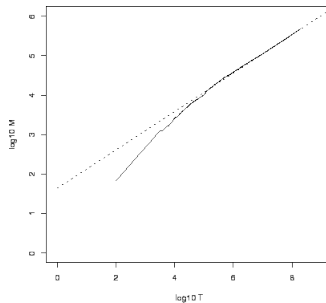
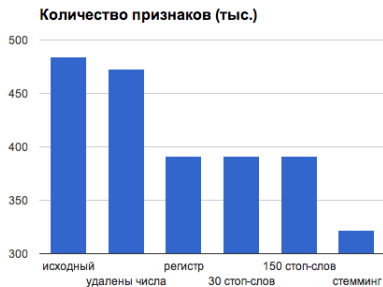
## Наблюдение

для сложных языков лучше подходит лемматизация

# Heap's law

$M = kT^b$ ,  $M$  – размер словаря,  $T$  – количество слов в корпусе

$$30 \leq k \leq 100, b \approx 0.5$$



# Представление документов

**Boolean Model.** Присутствие или отсутствие слова в документе

**Bag of Words.** Порядок токенов не важен

*Погода была ужасная, принцесса была прекрасная.  
Или все было наоборот?*

Координаты

- ▶ Мультиномиальные: количество токенов в документе
- ▶ Числовые: взвешенное количество токенов в документе



# Zipf's law

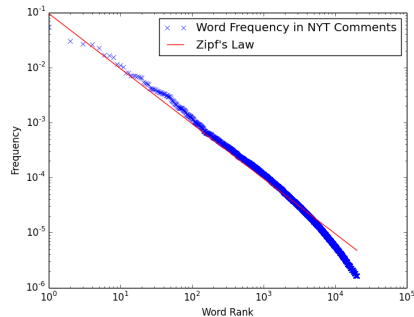
$t_1, \dots, t_N$  – токены, отсортированные по убыванию частоты

$f_1, \dots, f_N$  – соответствующие частоты

## Закон Ципфа

$$f_i = \frac{c}{i^k}$$

Что еще? Посещаемость сайтов,  
количество друзей, население городов...



# Задача

Дана коллекция, содержащая  $10^6$  (не уникальных) токенов. Предполагая, что частоты слов распределены по закону

$$f_i = \frac{c}{(i + 10)^2},$$

оцените

- ▶ количество вхождений наиболее часто встречающегося слова
- ▶ количество слов, котоые встречаются минимум дважды

Подсказка:  $\sum_{i=11}^{\infty} \frac{1}{i^2} \approx 0.095$

## BoW & TF-IDF

Количество вхождений слова  $t$  в документе  $d$

$$TF_{t,d} = \text{term-frequency}(t, d)$$

Количество документов из  $N$  возможных, где встречается  $t$

$$DF_t = \text{document-frequency}(t)$$

$$IDF_t = \text{inverse-document-frequency}(t) = \log \frac{N}{DF_t}$$

TF-IDF

$$TF-IDF_{t,d} = TF_{t,d} \times IDF_t$$

# Naive Bayes



# Байесовский классификатор

Дано

$\mathbf{x} \in X$  – описание документа  $d$  из коллекции  $D$

$C_k \in C$ ,  $k = 1, \dots, K$  – целевая переменная

Теорема Байеса

$$P(C_k | \mathbf{x}) = \frac{p(\mathbf{x} | C_k)p(C_k)}{p(\mathbf{x})} \propto p(\mathbf{x} | C_k)p(C_k)$$

Принцип Maximum A-Posteriori

$$C_{MAP} = \arg \max_k p(C_k | \mathbf{x})$$

# Naive Bayes

$x_j$  – слово на  $j$ -м месте в документе  $\mathbf{x}$ ,  $w^i \in V$  – слово из словаря  $V$

Предположения

1. conditional independence

$$p(x_i = w^s, x_j = w^r | C_k) = p(x_i = w^s | C_k) p(x_j = w^r | C_k)$$

2. positional independence

$$P(x_i = w^s | C_k) = P(x_j = w^s | C_k) = P(x = w^s | C_k)$$

Получаем

$$p(\mathbf{x} | C_k) = p(x_1 = w^{s_1}, \dots, x_{|\mathbf{x}|} = w^{s_{|\mathbf{x}|}} | C_k) = \prod_{i=1}^{|\mathbf{x}|} p(x = w^{s_i} | C_k)$$

Почему NB хорошо работает?

Корректная оценка дает правильное предсказание, но правильное предсказание *не* требует корректной оценки

# Варианты NB

## MAP

$$\begin{aligned} C_{MAP} &= \arg \max_k \prod_{i=1}^{|\mathbf{x}|} p(x = w^{s_i} | C_k) p(C_k) = \\ &= \arg \max_k \left[ \log p(C_k) + \sum_{i=1}^{|\mathbf{x}|} \log p(x = w^{s_i} | C_k) \right] \end{aligned}$$

Априорные вероятности

$$p(C_k) = N_{C_k} / N$$

Likelihood  $p(x = w^{s_i} | C_k)$

- ▶ BernoulliNB  $p(x = w^{s_i} | C_k) = D_{w^{s_i}, C_k} / D_{C_k}$ ,  $D$  – кол-во документов
- ▶ MultinomialNB  $p(x = w^{s_i} | C_k) = T_{w^{s_i}, C_k} / T_{C_k}$ ,  $T$  – кол-во токенов
- ▶ GaussianNB  $p(x = w^{s_i} | C_k) = \mathcal{N}(\mu_k, \sigma_k^2)$ , параметры из MLE

## Обучение NB

```
1 function nb_train(D,C):
2     V = dictionary of tokens
3     N = number of documents
4     for Ck in C: # iterate over all classes
5         N_Ck = number of documents in class Ck
6         p(Ck) = N_Ck / N # Class prior
7         D_Ck = Documents in class Ck
8         for w_i in V:
9             # multinomial, bernoulli, gaussian
10             p(w_i|Ck) = count_likelihood(...)
11     return V, p(Ck), p(w_i|Ck)
```

Алгоритмическая сложность:  $O(|D|\langle|\mathbf{x}|\rangle + |C||V|)$



## Применение MultinomialNB

```
1 function nb_apply(d, C, V, p(Ck), p(w_i|Ck)):  
2     x = tokenize(d) # somehow  
3     for Ck in C: # iterate over all classes  
4         score(Ck|x) = log p(Ck) # use class prior  
5         # use likelihoods  
6         for i in 1..|x|:  
7             score(Ck|x) += log p(x_i|Ck)  
8     return arg max score(Ck|x)
```

Алгоритмическая сложность:  $O(|C||x|)$

# Задача

d	Текст	Класс
1	котики такие мокрые	мимими
2	пушистые котики няшки	мимими
3	морские котики	не мимими
4	мокрые морские свинки	не мимими
5	котики мокрые	???

С помощью алгоритма MultinomialNB вычислить  $p(\text{мимими} | d_5)$

## Сглаживание

Проблема:  $p(\text{свинки}|\text{мимими}) = 0$

Решение:

$$p(x = w_{s_i} | C_k) = \frac{T_{w_{s_i}, C_k} + \alpha}{T_{C_k} + \alpha |V|}$$

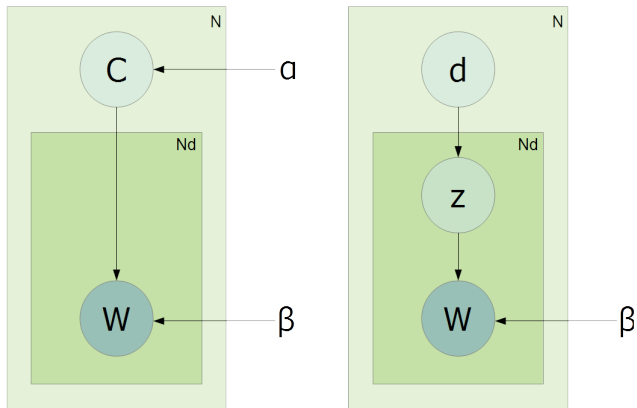
если  $\alpha \geq 1$  – сглаживание Лапласа, если  $0 \leq \alpha \leq 1$  – Лидстоуна

### Упражнение

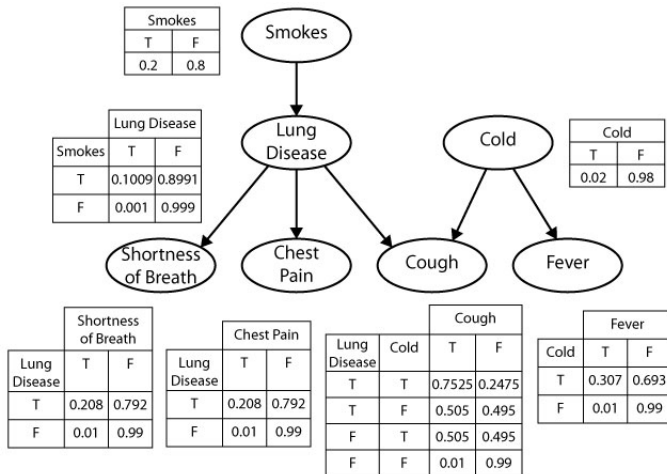
С учетом сглаживания вычислить

$p(\text{пушистые}|\text{не мимими}), p(\text{пушистые}|\text{мимими}).$

# Генеративная модель



# Байесовские сети



# Итоги

- + Генеративная модель
- + (Удивительно) неплохо работает
- + Стабилен при смещении выборки (aka concept drift)
- + Оптимальный по производительности
- Наивные предположения
- Требуется отбор признаков

## Вопросы

