Математические методы анализа текстов

Тематическое моделирование. Анализ тональности.

Мурат Апишев

24 апреля, 2017

Тематическое моделирование

Тематическое моделирование (*Topic Modeling*) — приложение машинного обучения к статистическому анализу текстов.

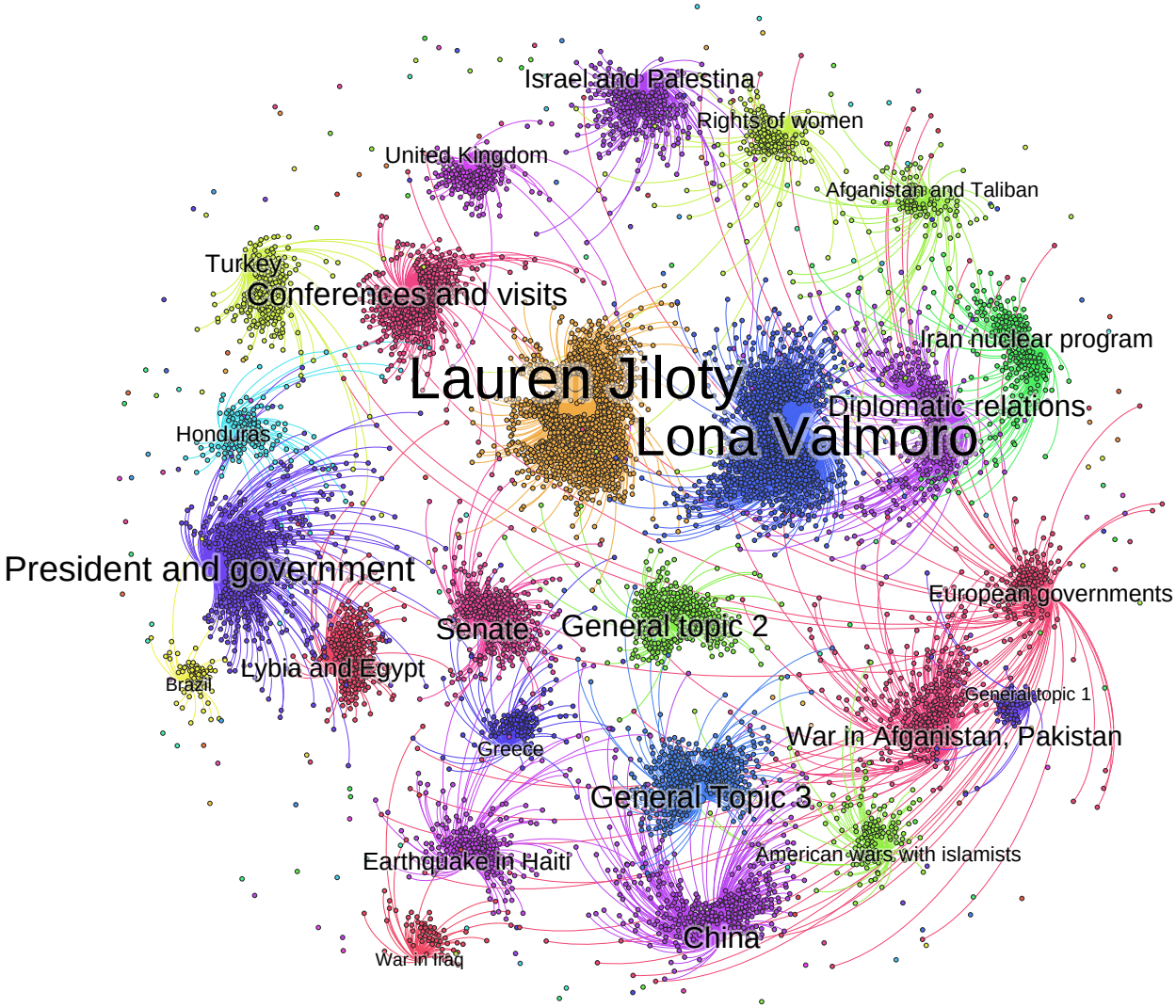
Тема — терминология предметной области, набор терминов (униграм или $n-$грам) часто встречающихся вместе в документах.

Тематическая модель исследует скрытую тематическую структуру коллекции текстов:

- *тема* t — это вероятностное распределение $p(w|t)$ над терминами w
- *документ* d — это вероятностное распределение $p(t|d)$ над темами t

Нестрого говоря, тема — это набор слов, глядя на которые можно сказать, какую предметную область они описывают.

Кластеризация и классификация документов



Анализ социальных медиа

- Извлечение из корпуса сообщений социальных медиа информации об обсуждаемых там темах.
- Исследование значимости тех или иных тем, определение интересов аудитории
- Выделение методами TM специфических тем из интересующей предметной области:
  - Национальные/этнические вопросы
  - Разного рода незаконная деятельность
  - Политические и экономические проблемы
- Отслеживание распространённости интересных тем в пространстве и времени

Информационный поиск

Информационный (разведочный) поиск — парадигма поиска, отличная от современных поисковых систем, направленная на изучение предметной области, связанной с запросом.

Поисковый запрос — документ или коллекция документов.

Поисковая потребность — информация о предметной области.

Концепция особенно полезна в ситуации, когда неясно, каким образом задавать короткий поисковый запрос:

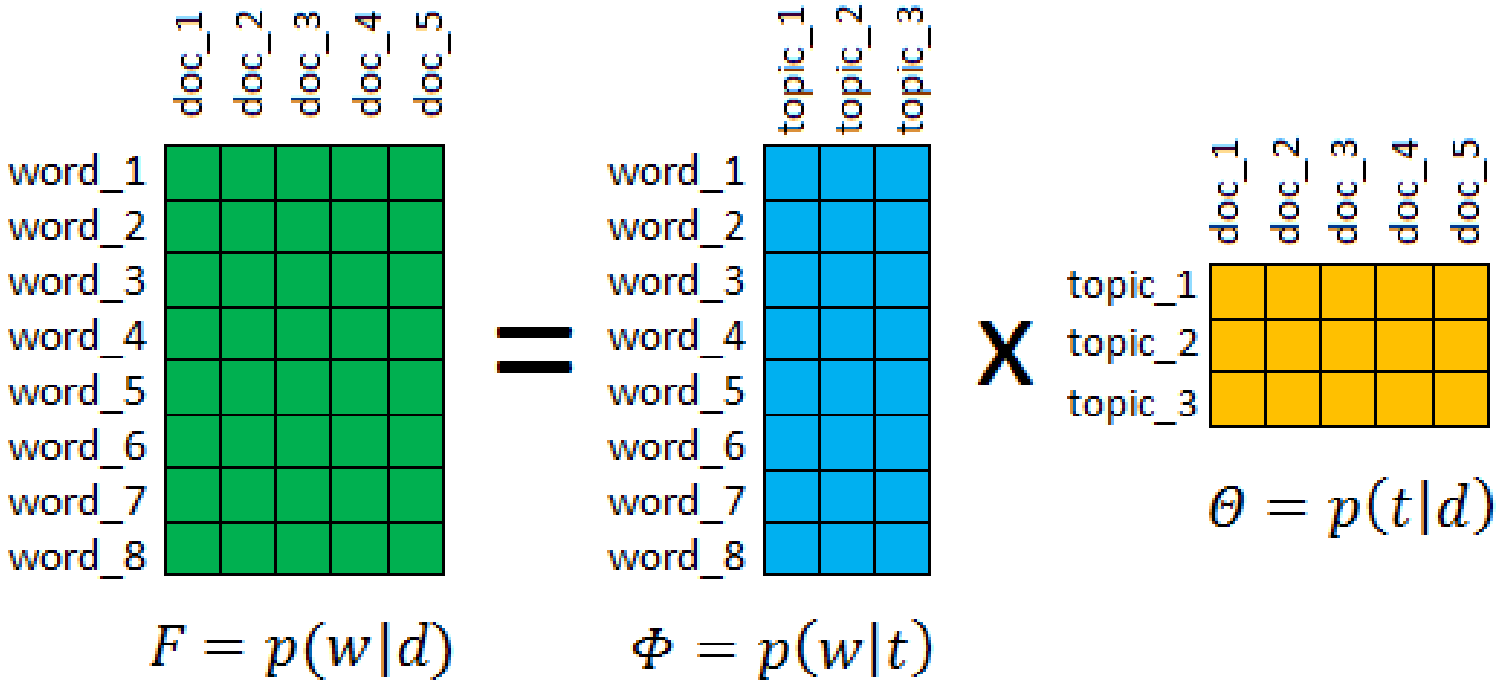
- отсутствие подходящих слов
- проблема омонимии

Особенно полезным тематический поиск может быть для научного и инженерного сообществ.

PLSA

Probabilistic Latent Semantic Analysis:

$$p(w|d) = \sum_{t \in T} p(w|t)p(t|d)$$



PLSA

Дано: W — словарь терминов (униграм или $n-$биграм),
D — коллекция текстовых документов $d \subset W$,
n_{dw} — счётчик частоты появления слова w в документе d.

Найти: модель $p(w|d) = \sum_{t \in T} \phi_{wt} \theta_{td}$ с параметрами $\underset{W \times T}{\Phi}$ и $\underset{T \times D}{\Theta}$:
$\phi_{wt} = p(w|t)$ — вероятности терминов w в каждой теме t,
$\theta_{td} = p(t|d)$ — вероятности тем t в каждом документе d.

Критерий максимизация логарифма правдоподобия:

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} \;\to\; \max_{\phi, \theta};$$

$$\phi_{wt} \geqslant 0; \quad \sum_w \phi_{wt} = 1; \qquad \theta_{td} \geqslant 0; \quad \sum_t \theta_{td} = 1.$$

PLSA

Максимизация логарифма правдоподобия:

$$\sum_{d\in D}\sum_{w\in d} n_{dw} \ln \sum_{t} \phi_{wt}\theta_{td} \;\to\; \max_{\Phi,\Theta}$$

EM-алгоритм: метод простых итерация для решения системы уравнений

$$\begin{cases} p_{tdw} = \underset{t\in T}{\operatorname{norm}}\left(\phi_{wt}\theta_{td}\right) \\ \phi_{wt} = \underset{w\in W}{\operatorname{norm}}\left(n_{wt}\right), \quad n_{wt} = \sum_{d\in D} n_{dw}p_{tdw} \\ \theta_{td} = \underset{t\in T}{\operatorname{norm}}\left(n_{td}\right), \quad n_{td} = \sum_{w\in d} n_{dw}p_{tdw} \end{cases}$$

E-шаг:

M-шаг:

где $\underset{i\in I}{\operatorname{norm}}\, x_i = \frac{\max\{x_i,0\}}{\sum_{j\in I}\max\{x_j,0\}}$

PLSA и перплексия

Величина, характеризующая степень сходимости модели с заданным словарём W — *перплексия*:

$$P(D) = \exp\left(-\frac{1}{n}\sum_{d\in D}\sum_{w\in d} n_{dw} \ln \sum_{t\in T} \phi_{wt}\theta_{td}\right), \quad n = \sum_{d} n_d.$$

Она построена на основе логарифма правдоподобия и характеризует степень качества описания коллекции моделью. Чем ниже — тем лучше. Алгоритм оптимизирует именно её.

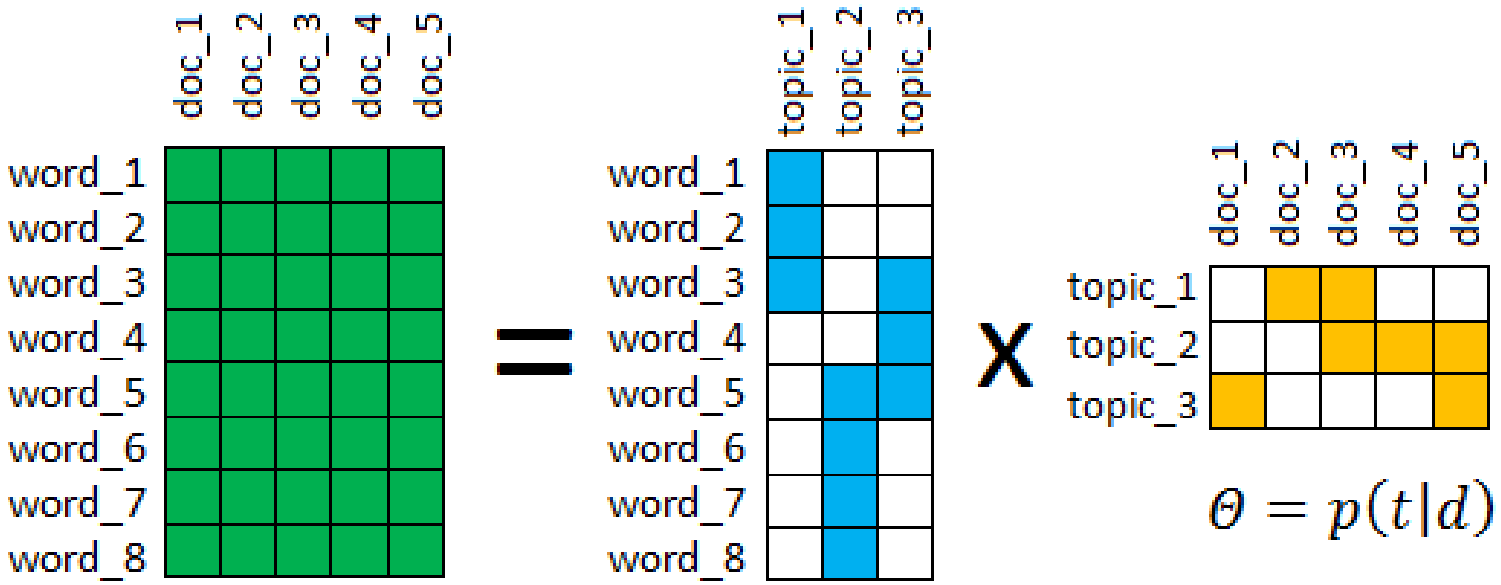
Но! Матричное разложение $F \approx \Phi \times \Theta$ имеет бесконечное множество решений: $\Phi\Theta = (\Phi S)(S^{-1}\Theta) = \Phi'\Theta' \Rightarrow$ можно подобрать Φ и Θ подходящего вида.

Существуют различные прикладные метрики качества. Они не оптимизируются моделью напрямую, но их значения хочется повышать. **Выход — регуляризация!**

Пример регуляризации

Логичные предположения:

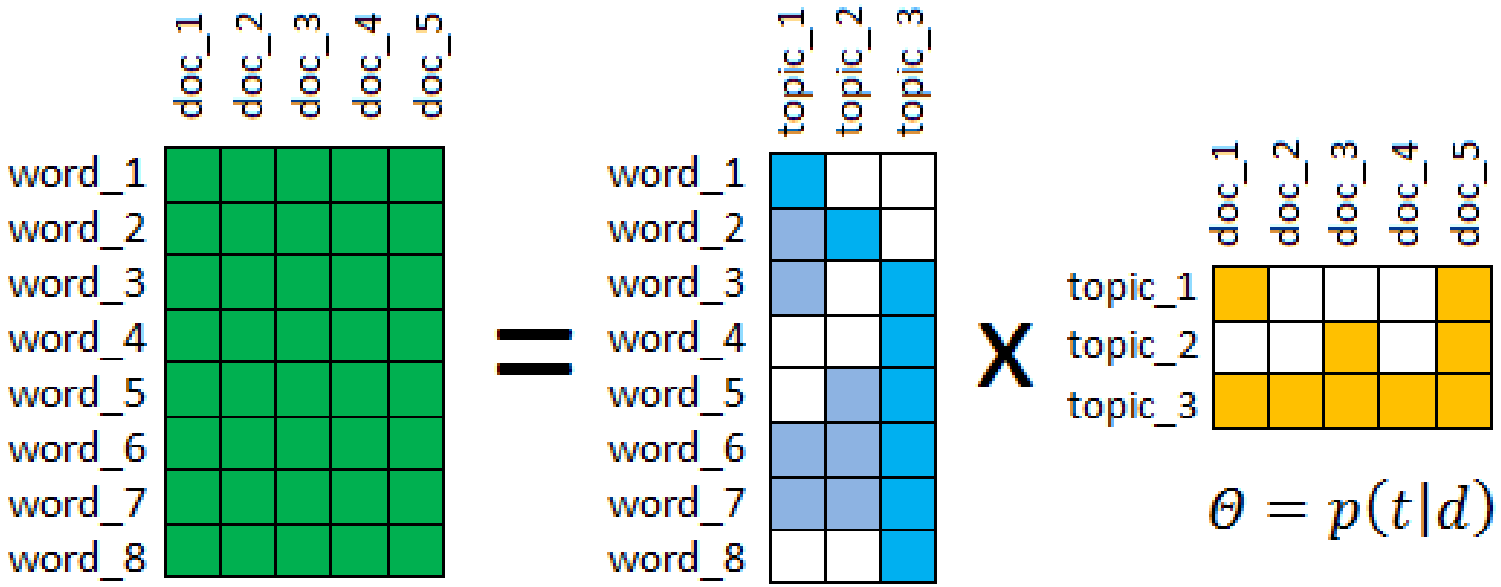
- темы должны состоять из небольшого числа слов, и эти множества слов не должны сильно пересекаться;
- каждый документ должен относиться к небольшому числу тем.



Пример регуляризации

Извлечение специфичной тематики по ключевым словам:

- хотим собрать темы около интересующих слов, а документы — около интересующих тем;
- прочие темы хотим сглаживать по неважным словам, чтобы собрать «мусор».



$F = p(w|d)$ $\quad \Phi = p(w|t)$ $\quad \Theta = p(t|d)$

LDA

Latent Dirichlet Allocation:

$$p(S, Z, \Theta, \Phi; \alpha, \beta) = \prod_{t=1}^{T} p(\Phi_t; \beta) \prod_{d=1}^{D} p(\Theta_d; \alpha) \times$$
$$\times \prod_{k=1}^{N} p(Z_{dk}|\Theta_d) p(S_{dk}|\Phi_{Z_{dk}})$$

- N — суммарная длина коллекции в словах
- S — N-мерный вектор всех слов коллекции
- Z — N-мерный вектор присваиваний тем для каждого слова из S
- Параметры Θ_d и Φ_t являются распределениями Дирихле с параметрами α и β соответственно — **регуляризатор**.

LDA

Методы обучения LDA:

1. Вариационный байесовский вывод
2. Сэмплирование Гиббса
3. ...

Недостатки LDA:

1. Сложный вывод
2. Сложность комбинирования регуляризаторов
3. Лингвистическая необоснованность регуляризатора сглаживания Дирихле

ARTM

Additive Regularization of Topic Models:

Максимизация логарифма правдоподобия с дополнительными аддитивными регуляризаторами R:

$$\sum_{d\in D}\sum_{w\in d} n_{dw} \ln \sum_{t} \phi_{wt}\theta_{td} + R(\Phi,\Theta) \;\to\; \max_{\Phi,\Theta}$$

EM-алгоритм: метод простых итераций для системы уравнений

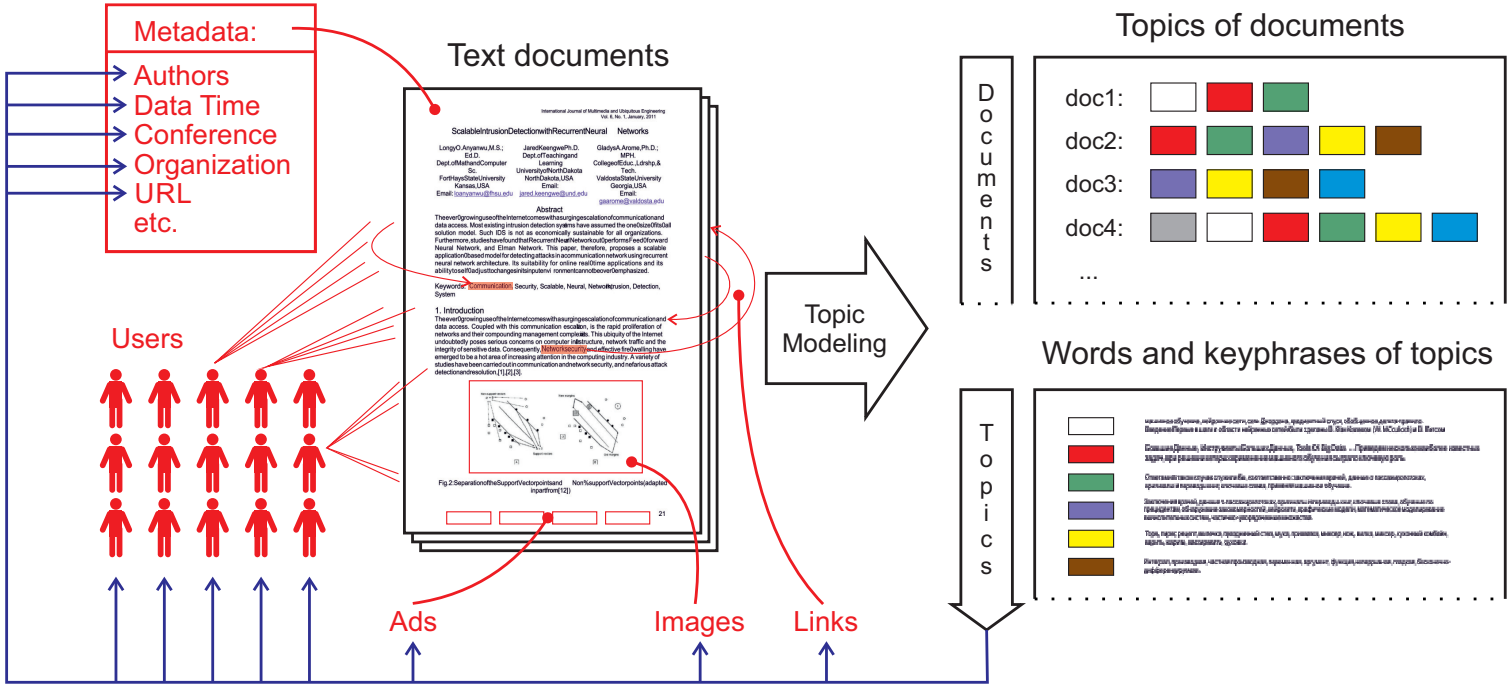
$$\begin{cases} \text{E-шаг:} & p_{tdw} = \underset{t\in T}{\text{norm}}\left(\phi_{wt}\theta_{td}\right) \\ \text{M-шаг:} & \phi_{wt} = \underset{w\in W}{\text{norm}}\left(n_{wt} + \phi_{wt}\dfrac{\partial R}{\partial \phi_{wt}} \right), \quad n_{wt} = \sum\limits_{d\in D} n_{dw} p_{tdw} \\ & \theta_{td} = \underset{t\in T}{\text{norm}}\left(n_{td} + \theta_{td}\dfrac{\partial R}{\partial \theta_{td}} \right), \quad n_{td} = \sum\limits_{w\in d} n_{dw} p_{tdw} \end{cases}$$

Примеры регуляризаторов

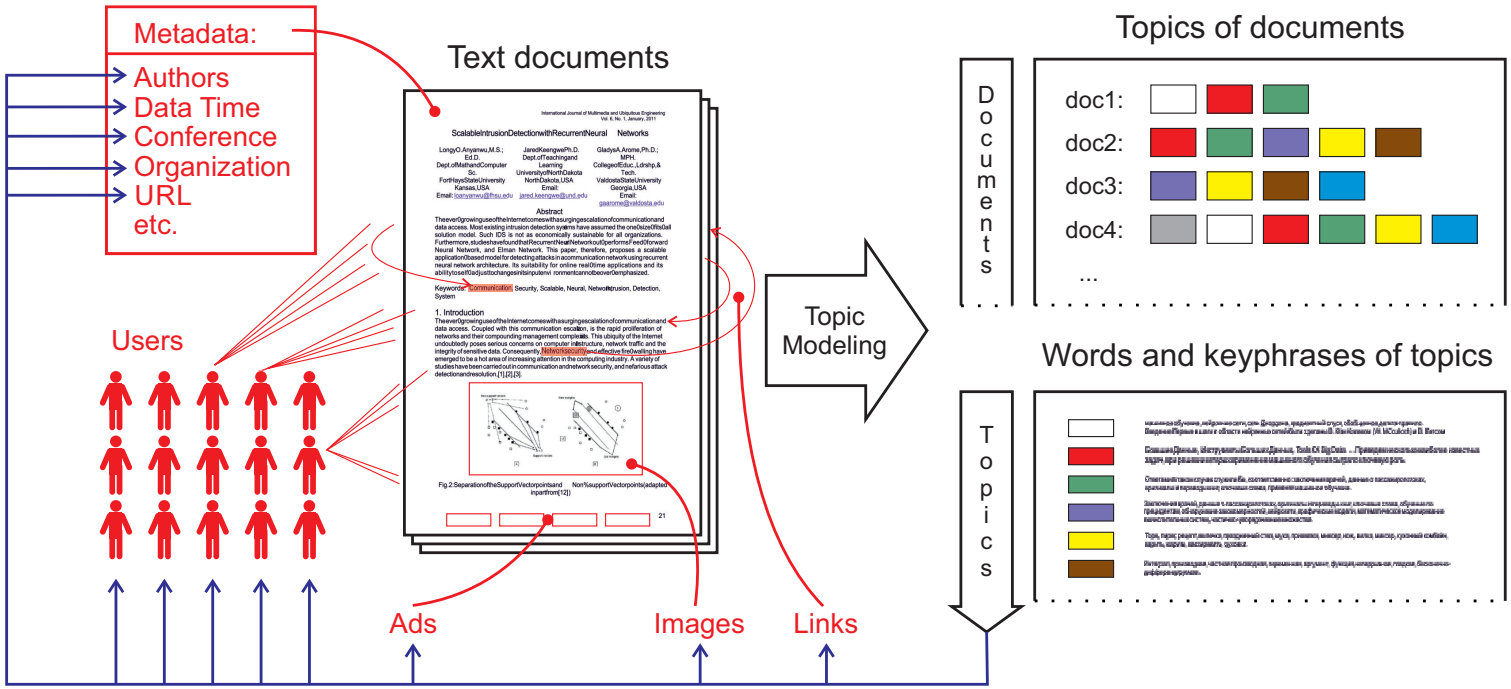
Существует много регуляризаторов. В ARTM наиболее простые и популярные следующие:

1. **Сглаживание** Φ / Θ — приводит к известной модели LDA.
2. **Разреживание** Φ / Θ — повышает разреженность тем и, как следствие, их интерпретируемость и различность.
3. **Декорреляция тем в** Φ — повышение различности тем (тоже разреживающая стратегия).
4. **Частичное обучение** — использование сглаживания/разреживания с предопределёнными словарями ключевых слов

Модальности слов



Мультимодальная тематическая модель

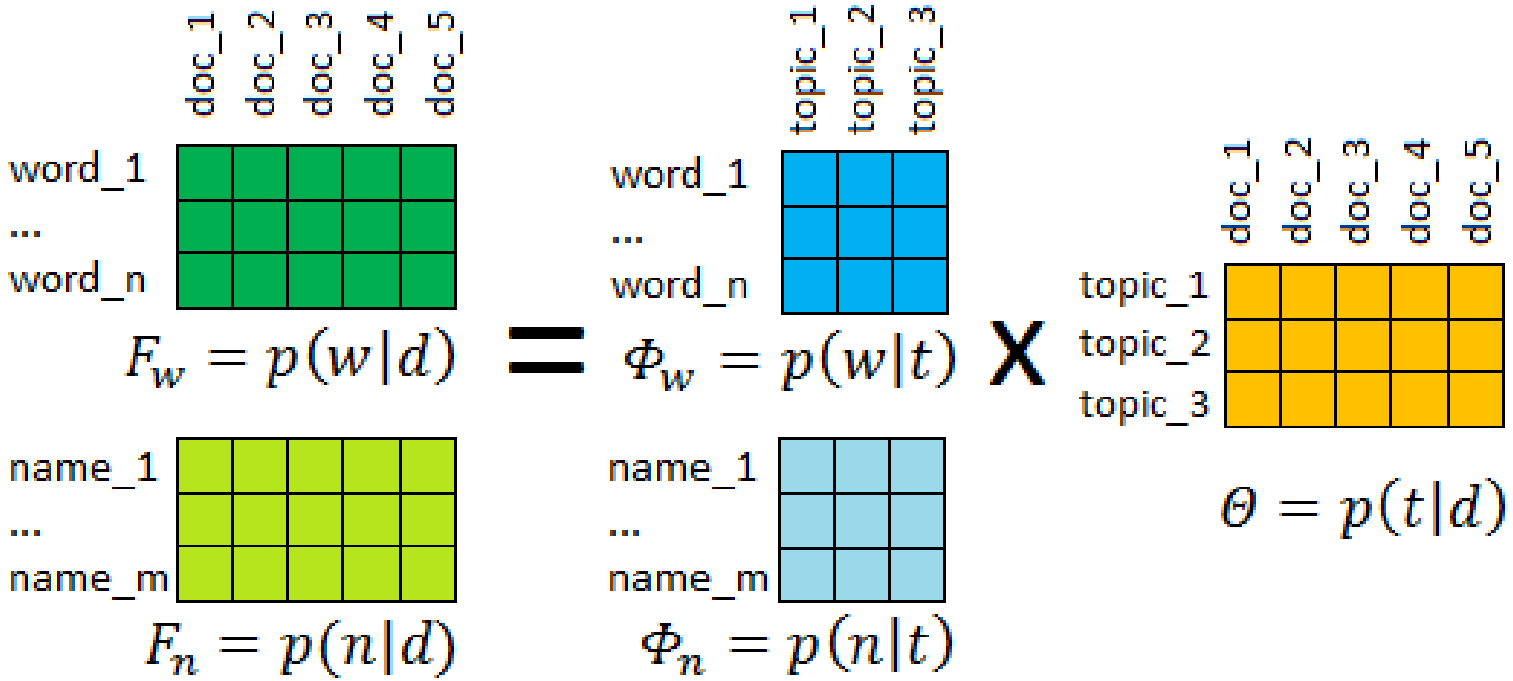


Мультимодальная TM строит распределения тем на терминах $p(w|t)$, авторах $p(a|t)$, метках времени $p(y|t)$, связанных документах $p(d'|t)$, рекламных баннерах $p(b|t)$, пользователях $p(u|t)$, и объединяет все эти модальности в одно тематическую модель.

Пример

Пусть у нас есть две модальности:

- обычные слова;
- слова-имена авторов (а можно, например, метки классов).



M-ARTM и EM-алгоритм

W^m — словарь терминов m-й модальности, $m \in M$,
$W = W^1 \sqcup W^m$ как объединение словарей всех модальностей.

Максимизация логарифма мультимодального правдоподобия с аддитивными регуляризаторами R:

$$\sum_{m \in M} \lambda_m \sum_{d \in D} \sum_{w \in W^m} n_{dw} \ln \sum_t \phi_{wt} \theta_{td} + R(\Phi, \Theta) \;\to\; \max_{\Phi, \Theta}$$

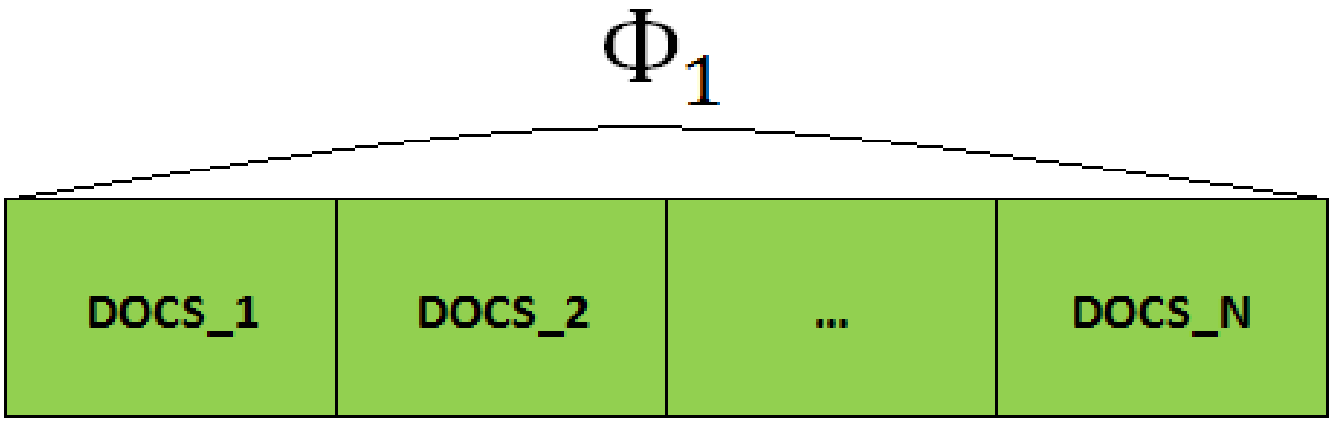
EM-алгоритм: метод простых итерация для системы уравнений

$$\text{E-шаг:} \quad p_{tdw} = \underset{t \in T}{\operatorname{norm}} \left(\phi_{wt} \theta_{td} \right)$$

$$\text{M-шаг:} \quad \begin{cases} \phi_{wt} = \underset{w \in W^m}{\operatorname{norm}} \left(n_{wt} + \phi_{wt} \dfrac{\partial R}{\partial \phi_{wt}} \right), \quad n_{wt} = \sum\limits_{d \in D} \lambda_{m(w)} n_{dw} p_{tdw} \\ \theta_{td} = \underset{t \in T}{\operatorname{norm}} \left(n_{td} + \theta_{td} \dfrac{\partial R}{\partial \theta_{td}} \right), \quad n_{td} = \sum\limits_{w \in d} \lambda_{m(w)} n_{dw} p_{tdw} \end{cases}$$

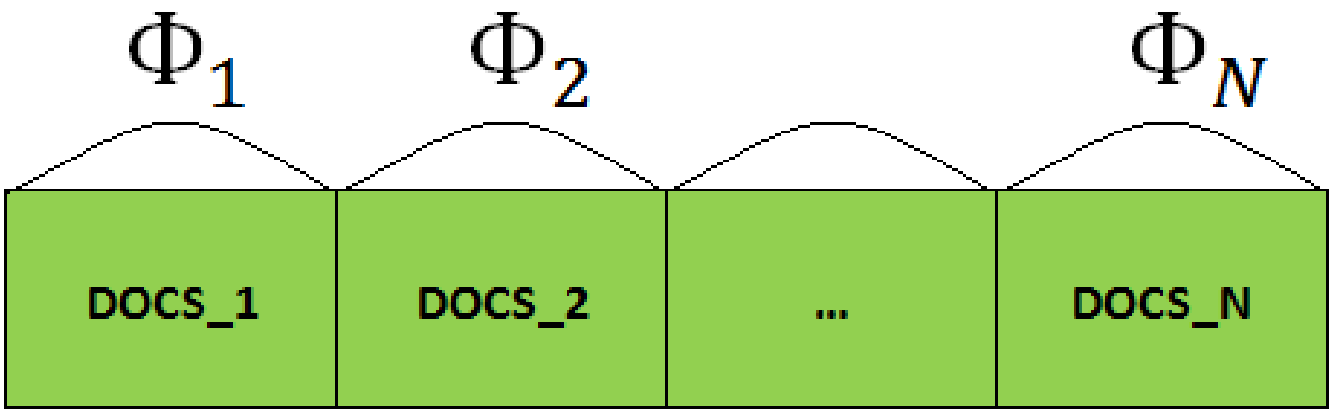
Оффлайн EM-алгоритм

1. Многократное итерирование по коллекции.
2. Однократный проход по документу.
3. Необходимость хранить матрицу Θ.
4. Φ обновляется в конце каждого прохода по коллекции.
5. Применяется при обработке небольших коллекций.



Онлайн EM-алгоритм

1. Однократный проход по коллекции.
2. Многократное итерирование по документу.
3. Нет необходимости хранить матрицу Θ.
4. Φ обновляется через определённое число обработанных документов.
5. Применяется при обработке больших коллекций в потоковом режиме.



Существующие реализации

1. **Gensim** (Online Variation LDA, Python, parallel)
2. **BigARTM** (Online ARTM, C++/Python, parallel)
3. **Vowpal Wabbit LDA** (Online Variation LDA, C++)
4. **Scikit-learn LDA** (Online Variation LDA, Python)

Данные

Опишем вспомогательный код:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.datasets import fetch_20newsgroups

d = fetch_20newsgroups(
       remove=('headers', 'footers', 'quotes')).data

cv = CountVectorizer(max_features=1000,
                             stop_words='english')

def print_top_words(model, feature_names, n_top_words):
    for topic_idx, topic in enumerate(model.components_):
        print 'Topic {}:'.format(topic_idx)
        print ' '.join([feature_names[i]
            for i in topic.argsort()[:-n_top_words - 1:-1]])
        print
```

Scikit-learn LDA

- ✗ Однопоточный, неэффективный.
- ✓ Совместим в векторайзерами sklearn.
- ✓ Легко использовать.

```
lda = LatentDirichletAllocation(n_topics=20,
                                max_iter=50,
                                learning_method='batch')
lda.fit(cv.fit_transform(d))

tf_feature_names = cv.get_feature_names()
print_top_words(lda, tf_feature_names, n_top_words)
```

Scikit-learn LDA

Фрагмент вывода:

Topic 1:
key chip scsi encryption keys clipper bit use bus security
Topic 4:
drive card dos disk mac pc hard video memory drives
Topic 6:
law government people israel state right rights israeli war jews
Topic 7:
god jesus bible christian church christ christians faith life man
Topic 8:
mr president stephanopoulos going congress tax know clinton think house
Topic 12:
new research 1993 national university information april center health years
Topic 14:
window windows use using program application display server set motif

Gensim

- ✗ Неэффективный.
- ✓ Совместим в векторайзерами sklearn.
- ✓ Легко использовать.
- ✓ Поддерживает несколько входных форматов.

```
import gensim
corpus = gensim.matutils.Sparse2Corpus(
                cv.fit_transform(d),
                documents_columns=False)

id2word = {}
for index, token in enumerate(tf_feature_names):
    id2word[index] = token

gensim.models.ldamodel.LdaModel(corpus=corpus,
                                id2word=id2word,
                                num_topics=20)
lda.print_topics(lda.num_topics)
```

Vowpal Wabbit LDA

- ✗ Однопоточный.
- ✗ Не совместим в векторайзерами sklearn.
- ✓ Достаточно быстрый.

Подготовка данных:

```
n_wd = array(corpus.sparse.todense())
num_docs = n_wd.shape[1]

with open('corpus.vw.txt', 'w') as fout:
    for doc_id in xrange(num_docs):
        fout.write('| ')
        for token_id in xrange(1000):
            value = n_wd[token_id][doc_id]
            if value:
                fout.write('{}:{} '.format(
                                id2word[token_id], value))
        fout.write('\n')
```

Vowpal Wabbit LDA

Пример строки данных:

```
| addition:1 body:1 called:1 car:4 day:1 early:1
engine:1 history:1 info:1 know:1 late:1 looked:1
looking:1 mail:1 model:1 really:1 rest:1 saw:1 small:1
years:1
```

Запуск CLI:

```
/usr/local/bin/vw -d corpus.vw.txt --lda 20 --lda_D
12000 --readable_model lda.model.vw -b 10 -c -k
--passes 10
```

Результат:

В итоге получается получается таблица хэшей на темы. Чтобы получить из неё темы, нужно пошаманить.

Можно сделать всё проще!

Vowpal Wabbit LDA из Gensim

В Gensim есть обёртка на VW LDA, принимающая на вход данные в формате Gensim:

```
lda = gensim.models.wrappers.LdaVowpalWabbit(
                          '/usr/local/bin/vw',
                          corpus=corpus,
                          num_topics=20,
                          id2word=id2word)

lda.print_topics(lda.num_topics)
```

Фрагмент вывода:

```
u'0.044*god + 0.014*evidence + 0.013*bible +
0.013*people + 0.013*believe + 0.013*does +
0.010*say + 0.010*jesus + 0.010*christian +
0.010*think'
```

BigARTM

- ✓ Совместим в векторайзерами sklearn.
- ✓ Позволяет строить сложные модели.
- ✓ Параллельный, эффективный.
- ✓ Легко использовать.
- ✓ Поддерживает несколько входных форматов.
- ✗ Непросто настраивать сложные модели.

Интерфейсы:

1. Command Line Interface (модель ARTM)
2. Python API (модели LDA, ARTM и hARTM)

BigARTM — CLI

CLI работает с файлами в формате UCI и Vowpal Wabbit.

Запустим на том же файле, который был создан для VW LDA:

```
bigartm --read-vw-corpus corpus.vw.txt --save-batches
my_batches --save-dictionary my.dict

bigartm --use-batches my_batches
--num_collection_passes 40 -t 20 --save-model my.model

bigartm --load-model my.model --write-model-readable
my.model.txt
```

BigARTM — CLI

Фрагмент выходного файла (матрица Φ, в MS Excel):

token	class_id	topic_0	topic_1	topic_2	topic_3	topic_4
received	@default_class	5.57E-07	0.000153	0.00161	1.55E-12	0.000448
different	@default_class	2.20E-05	8.41E-06	0.000124	0.001541	0.004249
digital	@default_class	1.42E-10	0.002185	0	0	0
away	@default_class	1.11E-05	0.000173	1.56E-09	0.001393	0.001013
policy	@default_class	0	0.003284	0.003769	0.002034	2.13E-06
having	@default_class	0.007911	7.40E-07	0.000345	0.002905	0.001154
did	@default_class	4.97E-09	1.09E-06	0.003752	3.42E-05	0.006726

`@default_class` — имя модальности обычных слов, модальность по-умолчанию.

С регуляризаторами и модальностями можно работать из CLI. Но удобнее из Python.

BigARTM — Python API

LDA на результате CountVectorizer:

```
import artm
n_wd = array(cv.fit_transform(d).todense()).T
vocabulary = cv.get_feature_names()

bv = artm.BatchVectorizer(data_format='bow_n_wd',
                          n_wd=n_wd,
                          vocabulary=vocabulary)

model = artm.LDA(num_topics=20,
                 dictionary=bv.dictionary)
model.fit_offline(bv, num_collection_passes=40)

model.get_top_tokens()
```

Модель ARTM

```
m = artm.ARTM(num_topics=50,
              num_document_passes=10,
              regularizers=[],
              scores=[],
              dictionary=bv.dictionary,
              class_ids={'@default_class': 1.0},
              cache_theta=True)
```

`artm.Dictionary` — структура данных, формально состоящая из строк следующего вида:

```
token    class_id    tf    df    value
```

По одной строке на каждое слова из W.

Регуляризаторы

```
artm.SmoothSparsePhiRegularizer(
                name='SSP_REG',
                tau=1.0,
                class_ids=['@default_class'],
                topic_names=m.topic_names[0: 10],
                dictionary=bv.dictionary)
```

Новый регуляризатор можно добавить в любой момент:

```
m.regularizers.add(
                artm.SmoothSparsePhiRegularizer(
                        name='SSP_REG', tau=1.0))
```

Или редактировать добавленный ранее:

```
artm.regularizers['SSP_REG'].tau = -1.0
```

Метрики качества

С метриками качества всё аналогично:

```
artm.PerplexityScore(
                name='PERP_SCR',
                class_ids=['@default_class'],
                topic_names=m.topic_names[0: 10],
                dictionary=bv.dictionary)

m.scores.add(
      artm.PerplexityScore(
            name='PERP_SCR', dictionary=bv.dictionary))
```

Так можно получить список значений метрики по итерациям:

```
m.score_tracker['PERP_SCR'].value
```

BigARTM vs. Gensim vs. VW LDA

- 3.7M статей англ. Википедии, $|W|$=100k

Фреймворк	procs	обучение	вывод	перп.
BigARTM	1	35 min	72 sec	4000
LdaModel	1	369 min	395 sec	4161
VW.LDA	1	73 min	120 sec	4108
BigARTM	4	9 min	20 sec	4061
LdaMulticore	4	60 min	222 sec	4111
BigARTM	8	**4.5 min**	**14 sec**	4304
LdaMulticore	8	57 min	224 sec	4455

- *procs* = число параллельных потоков
- *вывод* = время подсчёта распределений θ_d для теста в 100k док-в

Задачи анализа тональности

Сантимент-анализ предназначен для выявление в тексте *отношений*.

Отношение (attitude) — это тройка

1. Источник — субъект отношения
2. Цель — объект отношения
3. Тип — категориальный список

Определяется либо для текста, либо для предложения.

Задачи сантимент-анализа

Три типа задач (по возрастанию сложности):

1. Полярная тональность (positive / negativ)
2. Ранжированная тональность («звёздочки» от 1 до 5)
3. Выявление источника / цели или более сложные типы

http://www.greenbookblog.org/2012/01/02/from-sentiment-analysis-to-enterprise-applications/

Этапы решения задачи

1. Токенизация — парсинг данных в удобное для работы представление
2. Извлечение признаков
3. Классификация (для демонстрации используем Naive Bayes)

При токенизации важно выявить и сохранить ригистр, смайлы, даты и телефоны. Для последних есть специальные регулярные выражения.

При извлечении признаков важно находить отрицании, определиться с используемыми частями речи (можно пробовать только прилигательные).

Учёт отрицаний и частот

Простой способ: добавить `NOT_` ко всему между отрицанием и следующим знаком препинания:

`didn't like this movie, but I`

$\Rightarrow$

`didn't NOT_like NOT_this NOT_movie, but I`

Таким образом можно почти удвоить словарь.

Для того, чтобы учесть важные, но редкие слова (типа fantastic), можно

- учитывать все уникальные слова документа со счётчиком 1
- использовать не счётчик частоты слова, а его логарифм

Сложности

- Отзывы могут иметь ясный смысл, но при этом не содержать позитивных или негативных слов:

  `Это фильм заставляет прочувствовать всю гамму эмоций от «А» до «Я».`

- Отзыв может содержать позитивные слова, но на самом деле выражает ожидание:

  `Это фильм должен был быть супер крутым. Но не был.`

Сантимент-лексикон

Существуют наборы слов с оценённой степенью позитивности/негативности, активности.пассивности и т.п.

Пример: MPQA subjectivity cues lexicon

	Strength	Length	Word	Part-of-speech	Stemmed	Polarity
1.	type=weaksubj	len=1	word1=abandoned	pos1=adj	stemmed1=n	priorpolarity=negative
2.	type=weaksubj	len=1	word1=abandonment	pos1=noun	stemmed1=n	priorpolarity=negative
3.	type=weaksubj	len=1	word1=abandon	pos1=verb	stemmed1=y	priorpolarity=negative
4.	type=strongsubj	len=1	word1=abase	pos1=verb	stemmed1=y	priorpolarity=negative
5.	type=strongsubj	len=1	word1=abasement	pos1=anypos	stemmed1=y	priorpolarity=negative
6.	type=strongsubj	len=1	word1=abash	pos1=verb	stemmed1=y	priorpolarity=negative
7.	type=weaksubj	len=1	word1=abate	pos1=verb	stemmed1=y	priorpolarity=negative
8.	type=weaksubj	len=1	word1=abdicate	pos1=verb	stemmed1=y	priorpolarity=negative
9.	type=strongsubj	len=1	word1=aberration	pos1=adj	stemmed1=n	priorpolarity=negative
10.	type=strongsubj	len=1	word1=aberration	pos1=noun	stemmed1=n	priorpolarity=negative
...						
8221.	type=strongsubj	len=1	word1=zest	pos1=noun	stemmed1=n	priorpolarity=positive

http://sentiment.christopherpotts.net/lexicons.html

Создание сантимент-лексикона

Это задача частичного обучения. Необходимо разметить часть примеров и поисать правила поплнения.

Пример: `bad and ugly; cruel but amazing`

Если два слова связаны and, и тональность одного нам известна, то второе тоже будет иметь такую тональность.

Если через but — то противоположную.

Можно задавать поисковые запросы вида `was nice and`, и поисковая система найдёт новые слова той же тональности для пополнения словаря.

О классификации

Naive Bayes:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}}\, p(c) \prod_{w \in d} \hat{p}(w|c)$$

$$\hat{p}(w|c) = \frac{count(w, c) + 1}{count(c) + |W|}$$

- $count(w, c)$ — число раз, когда слово w встретилось в документе с классом c;
- $count(c)$ — общее число слов в документах с классом c.

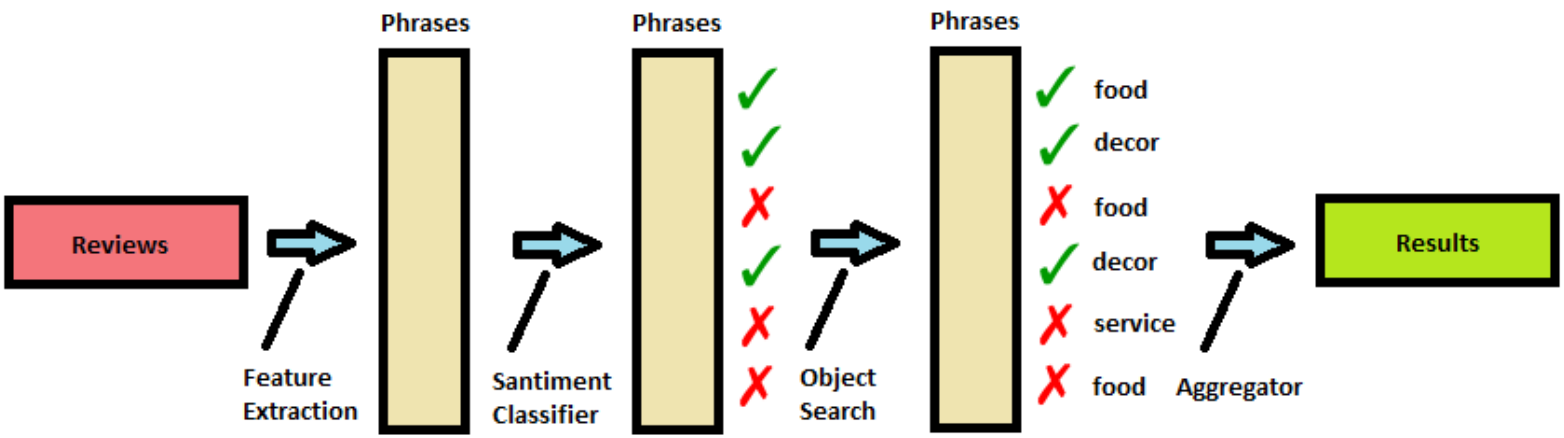
Можно нормировать $p(w|c)$ на $p(w)$.

Более сложные задачи

Поиск атрибутов и их объектов — в одном ревью могут по-разному оцениваться разные вещи.

Для этого ищем частые фразы, которые нам интересны, и смотрим на то, сколько раз они встречаются сразу после сантиментных слов (пример: `great fish tacos`).

Такую работу несложно проделать для ресторанов, отелей и т.п.: food, decor, service и синонимы.



Простой пример

```
import nltk.classify.util
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import movie_reviews

def get_feats(tokens):
    return dict([(token, True) for token in tokens])

def create_sample(tag, train_ratio):
    ids = movie_reviews.fileids(tag)
    feats = [(get_feats(movie_reviews.words(fileids=[f])),
                        tag) for f in ids]

idx = int(len(feats) * train_ratio)
train = feats[: idx]
text = feats[idx: ]

return train, test
```

Простой пример

```
train_pos, test_pos = create_sample('pos', 0.75)
train_neg, test_neg = create_sample('neg', 0.75)
train = train_pos + train_neg
test = test_pos + test_neg

print 'Train on {} docs, test on {} docs'.format(
                    len(train), len(test))

classifier = NaiveBayesClassifier.train(train)

print 'Accuracy: {}'.format(
     nltk.classify.util.accuracy(classifier, test))

classifier.show_most_informative_features()
```

Результаты

```
Train on 1500 documents, test on 4000 documents
Accuracy: 0.728
Most Informative Features
     magnificent = True      pos : neg = 15.0 : 1.0
     outstanding = True      pos : neg = 13.6 : 1.0
     insulting = True        neg : pos = 13.0 : 1.0
     vulnerable = True       pos : neg = 12.3 : 1.0
     ludicrous = True        neg : pos = 11.8 : 1.0
     avoids = True           pos : neg = 11.7 : 1.0
     uninvolving = True      neg : pos = 11.7 : 1.0
     astounding = True       pos : neg = 10.3 : 1.0
     fascination = True      pos : neg = 10.3 : 1.0
     idiotic = True          neg : pos = 9.8 : 1.0
```

Итоги занятия

- Тематическое моделирование — инструмент статистического анализа текстов, позволяющий быстро получать информацию о затрагиваемых в них темах.
- TM удобно использовать в качестве генератора признаков и кластеризатора.
- Существуют различные модели, наиболее общая — мультимодальная APTM.
- Наиболее быстрой реализацией TM для одной машины является BigARTM.
- Задача анализа тональности сводится к задаче классификации, основная проблема — выбор классификатора, предобработка и метод генерации признаков.

Спасибо за внимание!