Тематическое моделирование: теория, инструменты, приложения

Мурат Апишев

НИУ ВШЭ, МГУ им. Ломоносова, Яндекс, ШАД

25 сентября, 2017

Тематическое моделирование

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

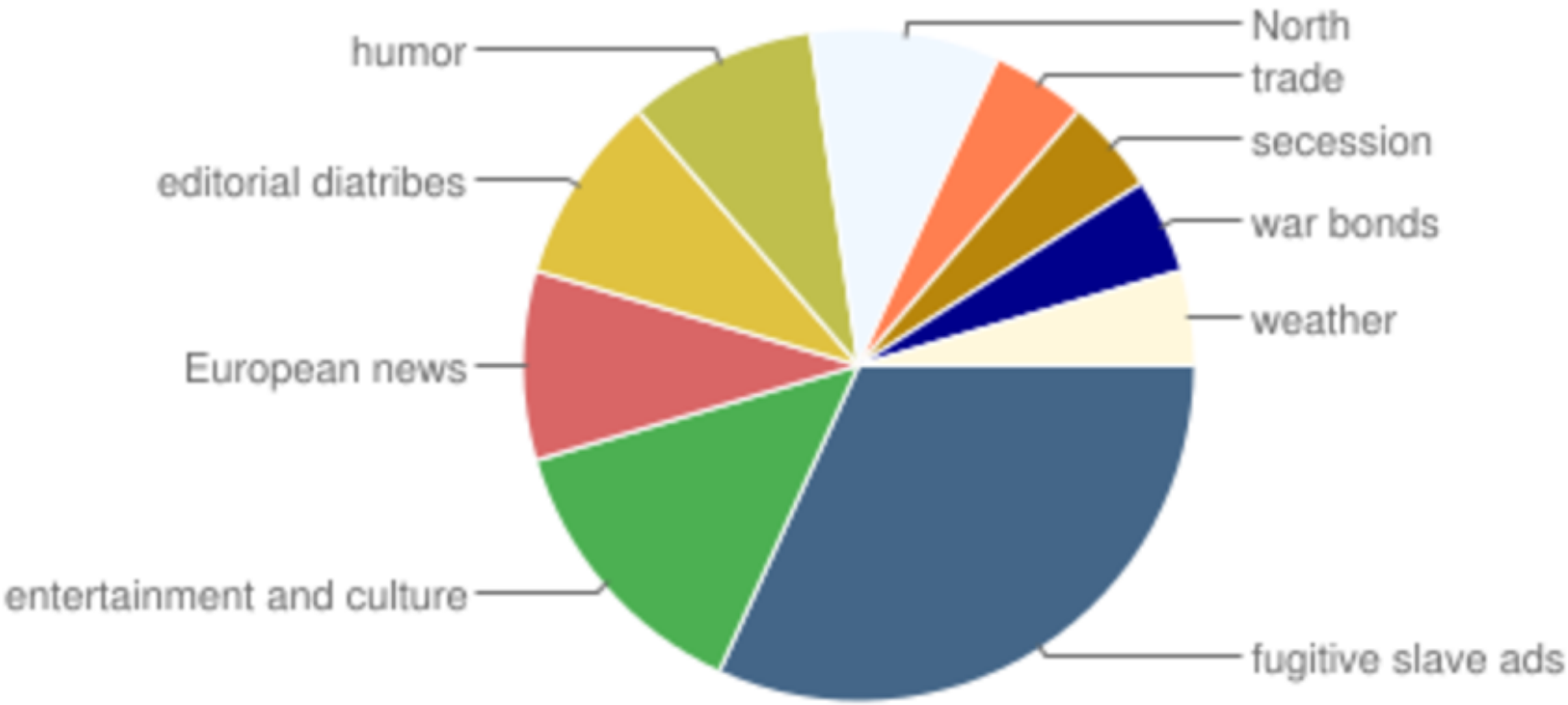
Тема — терминология предметной области, набор терминов (униграм или $n-$грам) часто встречающихся вместе в документах.

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.

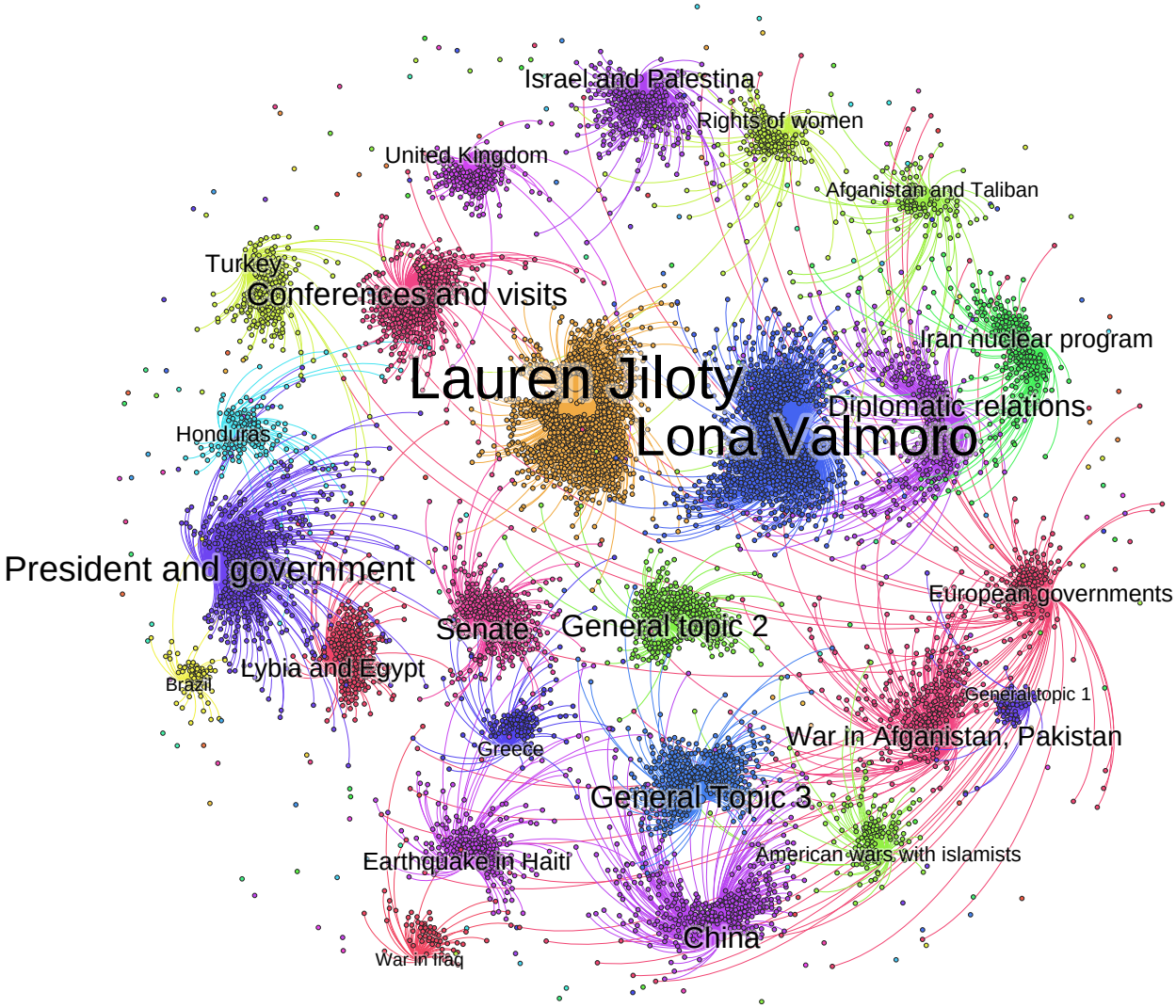
Определение тем и их соотношений



1

[1] http://dsl.richmond.edu/dispatch/pages/intro

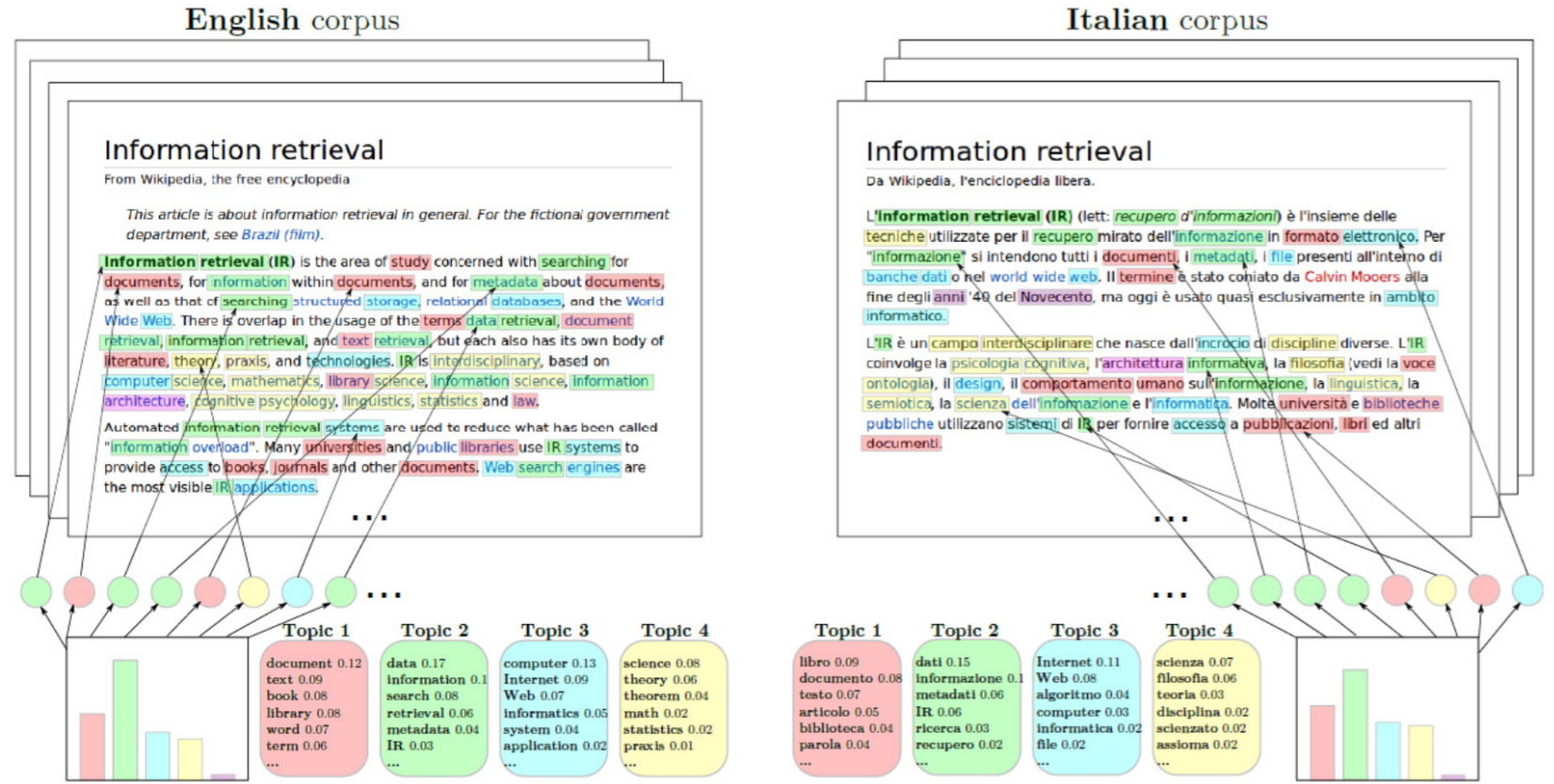
Кластеризация и классификация документов



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

Мультиязычные модели перевода



2

[2] I. Vulic, W. De Smet, J. Tang, M.-F. Moens. Probabilistic topic modeling in multilingual settings: a short overview of its methodology with applications // NIPS, 7–8 December 2012. — Pp. 1–11.

Приложения тематического моделирования

- Тематический поиск документов по тексту любой длины, или по любому объекту
- Поиск научных статей, экспертов, рецензентов, проектов
- Выявление трендов и фронтов исследования
- Суммаризация и аннотирование текстовых документов
- Анализ и агрегирование новостных потоков
- Рубрикация документов, видео, музыки
- Различные задачи биоинформатики

Требования к тематической модели

- Интерпретируемость выделяемых тем
- Обработка больших объёмов данных

Мешок слов

Мешок слов (Bag-Of-Words) — представление текстовых данных, в котором учитывается только частота встречаемости слов в документах. Порядок слов игнорируется.

Исходное предложение: `I can drink a milk can`

Его мешок слов:

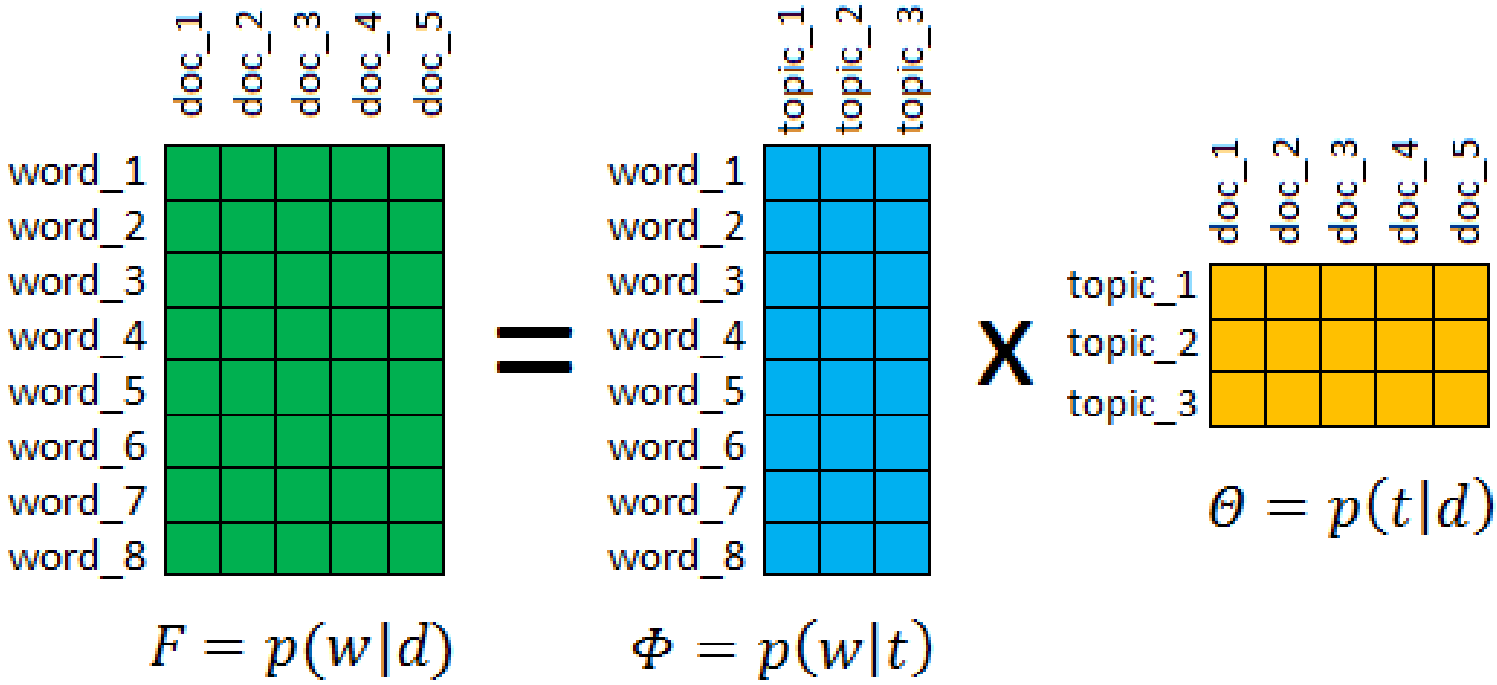
```
I: 1
can: 2
drink: 1
a: 1
milk: 1
```

Проще, но теряется много полезной информации.

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:

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

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

Интуиция EM-алгоритма

Есть текст

Pooh rubbed his nose again, and said that he hadn't thought of that. And then he brightened up, and said that, if it were raining already, the Heffalump would be looking at the sky wondering if it would clear up, and so he wouldn't see the Very Deep Pit until he was half-way down$\cdots$

Хотим оценить вероятности слов в темах $p(w|t)$ и тем в документах $p(t|d)$.

Интуиция EM-алгоритма

Если бы у нас были присваивания слов темам...

Pooh rubbed his nose again, and said that he hadn't thought of that. And then he brightened up, and said that, if it were raining already, the Heffalump would be looking at the sky wondering if it would clear up, and so he wouldn't see the Very Deep Pit until he was half-way down···

...мы могли бы просто посчитать:

$$p(w = sky|t) = \frac{n_{wt}}{\sum_w t} = \frac{1}{4} \qquad p(t = t|d) = \frac{n_{td}}{\sum_t n_{td}} = \frac{4}{54}$$

Но у нас есть только текст

Pooh rubbed his nose again, and said that he hadn't thought of that. And then he brightened up, and said that, if it were raining already, the Heffalump would be looking at the sky wondering if it would clear up, and so he wouldn't see the Very Deep Pit until he was half-way down···

Но у нас есть только текст

Pooh rubbed his nose again, and said that he hadn't thought of that. And then he brightened up, and said that, if it were raining already, the Heffalump would be looking at the sky wondering if it would clear up, and so he wouldn't see the Very Deep Pit until he was half-way down$\cdots$

Идея: попробуем оценить присваивания тем:

$$p(t|d,w) = \{\text{cond.rule}\} =$$

$$= \frac{p(w,t|d)}{p(w|d)} = \{\text{indep.} + \text{prod.rule}\} =$$

$$= \frac{p(w|t)p(t|d)}{p(w|d)}$$

Собираем всё вместе:

E-шаг:

$$p(t|d,w) = \frac{p(w|t)p(t|d)}{p(w|d)} = \frac{\phi_{wt}\theta_{td}}{\sum_t \phi_{wt}\theta_{td}}$$

M-шаг:

$$\phi_{wt} = \frac{n_{wt}}{\sum_w n_{wt}}, \quad n_{wt} = \sum_d n_{dw}\, p(t|d,w)$$

$$\theta_{td} = \frac{n_{td}}{\sum_t n_{td}}, \quad n_{td} = \sum_w n_{dw}\, p(t|d,w)$$

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.

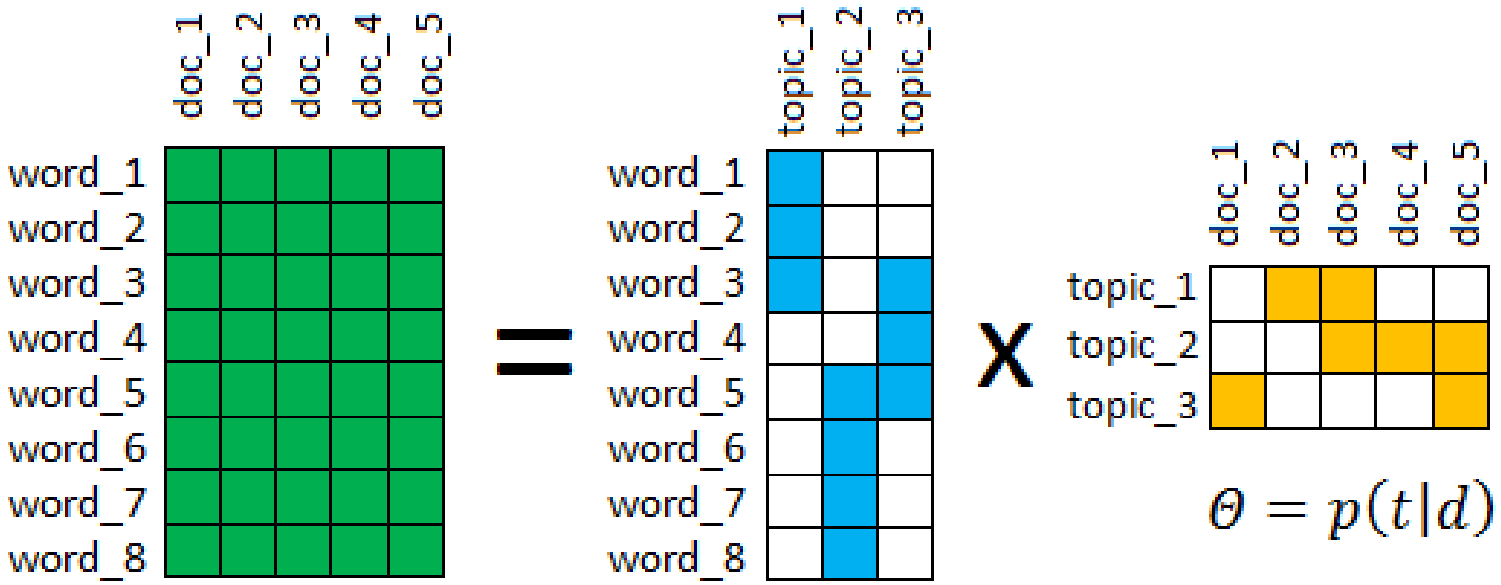
Но! Матричное разложение $F \approx \Phi \times \Theta$ имеет бесконечное множество решений: $\Phi\Theta = (\Phi S)(S^{-1}\Theta) = \Phi'\Theta' \Rightarrow$ можно подобрать Φ и Θ подходящего вида.

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

Пример регуляризации

Логичные предположения:

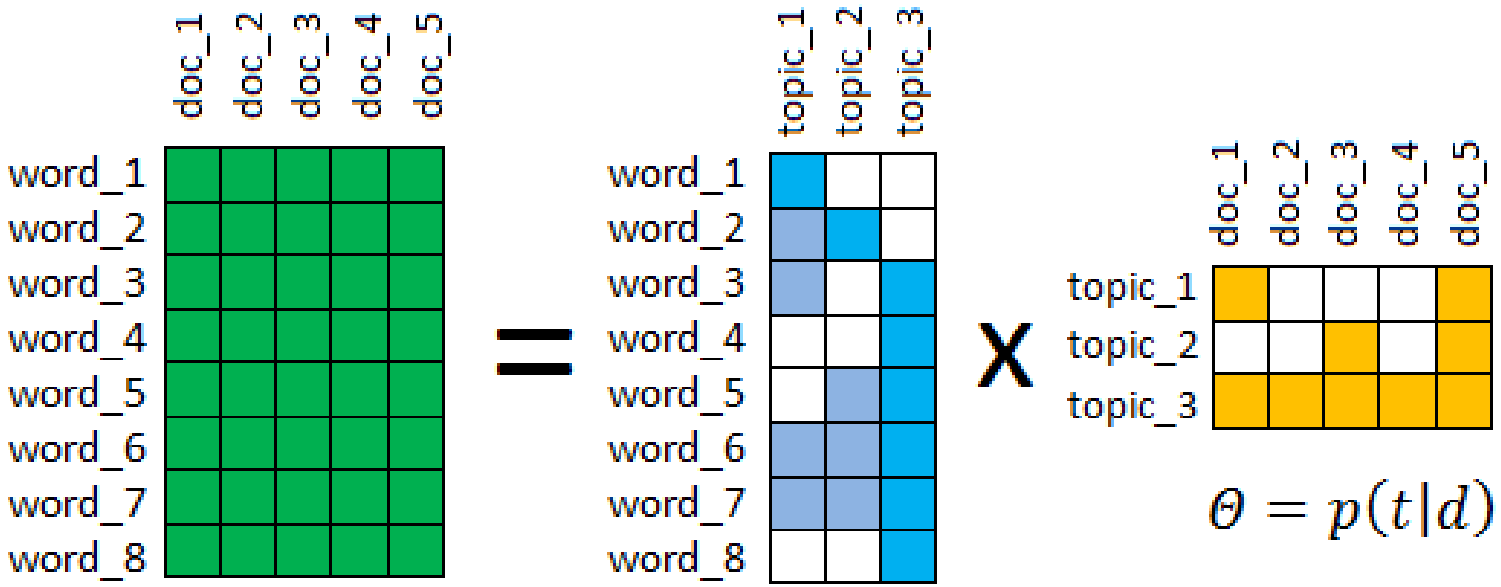
- темы должны состоять из небольшого числа слов, и эти множества слов не должны сильно пересекаться;
- каждый документ должен относиться к небольшому числу тем.



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».



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

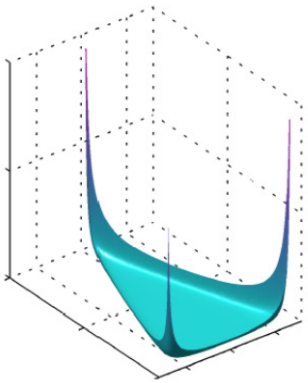
Модель LDA

Latent Dirichlet Allocation, [Blei, Ng, Jordan 2003]

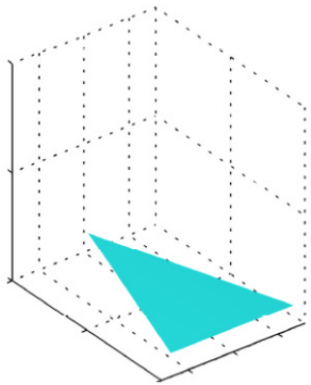
Гипотеза. Вектор-столбцы $\phi_t = (\phi_{wt})_{w \in W}$ и $\theta_d = (\theta_{td})_{t \in T}$ порождаются распределениями Дирихле, $\alpha \in \mathbb{R}^{|T|}$, $\beta \in \mathbb{R}^{|W|}$:

$$\mathrm{Dir}(\phi_t|\beta) = \frac{\Gamma(\beta_0)}{\prod_w \Gamma(\beta_w)} \prod_w \phi_{wt}^{\beta_w - 1}, \quad \beta_0 = \sum_w \beta_w, \ \beta_t \geqslant 0;$$

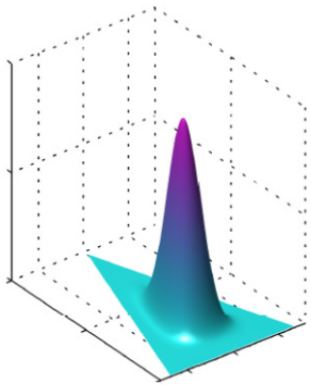
$$\mathrm{Dir}(\theta_d|\alpha) = \frac{\Gamma(\alpha_0)}{\prod_t \Gamma(\alpha_t)} \prod_t \theta_{td}^{\alpha_t - 1}, \quad \alpha_0 = \sum_t \alpha_t, \ \alpha_t \geqslant 0;$$



$\alpha_1 = \alpha_2 = \alpha_3 = 0.1$



$\alpha_1 = \alpha_2 = \alpha_3 = 1$



$\alpha_1 = \alpha_2 = \alpha_3 = 10$

Отличие LDA и PLSA в оценивании Φ и Θ

- в PLSA — точечные оценки максимума правдоподобия:

$$\phi_{wt} = \frac{n_{wt}}{n_t}, \qquad \theta_{td} = \frac{n_{td}}{n_d}$$

- в LDA — апостериорные распределения методами приближенного байесовского вывода (Variational Bayes, Gibbs Sampling):

$$\phi_{wt} \sim Dir\left(\phi_t | \frac{n_{wt} + \beta_w}{n_t + \beta_0}\right), \qquad \theta_{td} \sim Dir\left(\theta_d | \frac{n_{td} + \alpha_t}{n_d + \alpha_0}\right).$$

Asuncion A., Welling M., Smyth P., Teh Y. W. On smoothing and inference for topic models. Int'l Conf. on Uncertainty in Artificial Intelligence, 2009.

Potapenko A. A., Vorontsov K. V. Robust PLSA Performs Better Than LDA. ECIR-2013, Moscow, Russia, 24-27 March 2013. LNCS, Springer. Pp. 784–787.

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

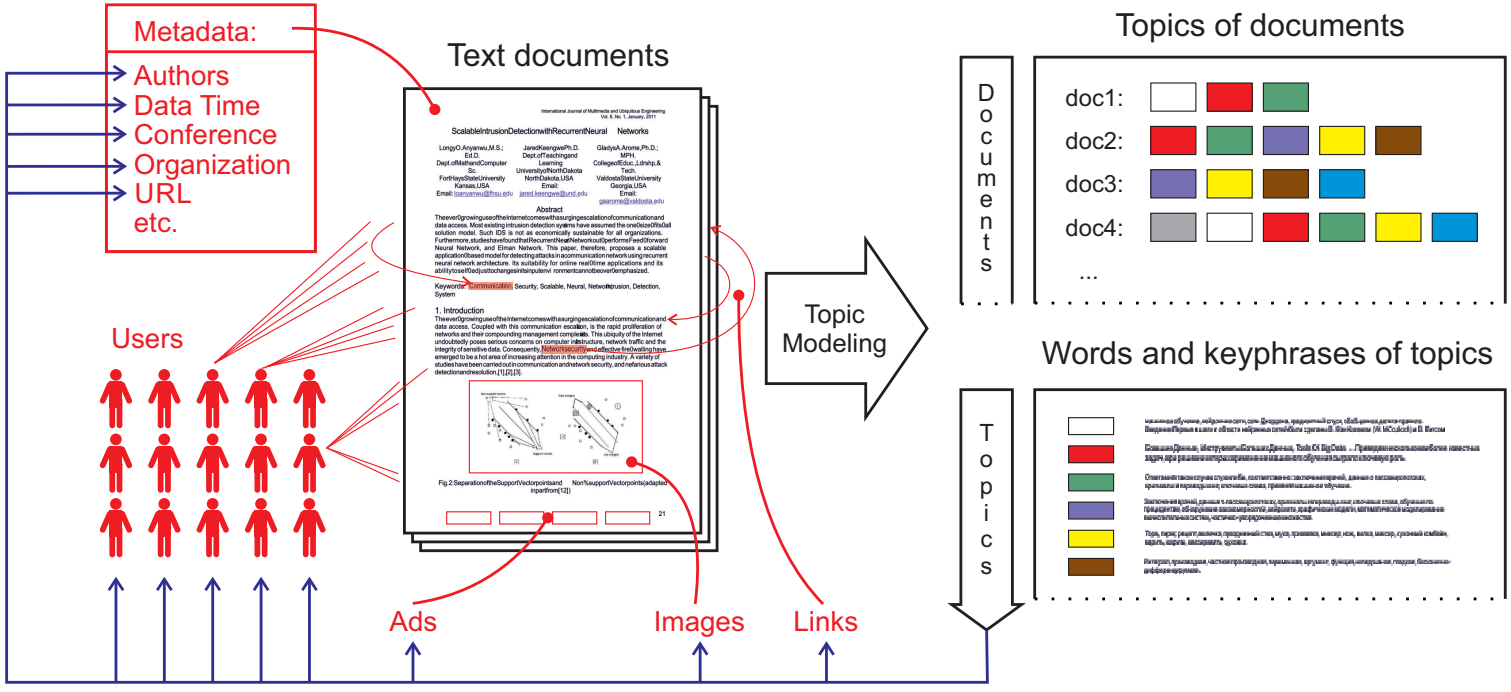
$$\begin{cases} \text{E-шаг:} & p_{tdw} = \underset{t\in T}{\text{norm}}\left(\phi_{wt}\theta_{td}\right) \\ \text{M-шаг:} & \phi_{wt} = \underset{w\in W}{\text{norm}}\left(n_{wt} + \phi_{wt}\dfrac{\partial R}{\partial \phi_{wt}} \right), \quad n_{wt} = \sum\limits_{d\in D} n_{dw} p_{tdw} \\ & \theta_{td}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

Примеры регуляризаторов

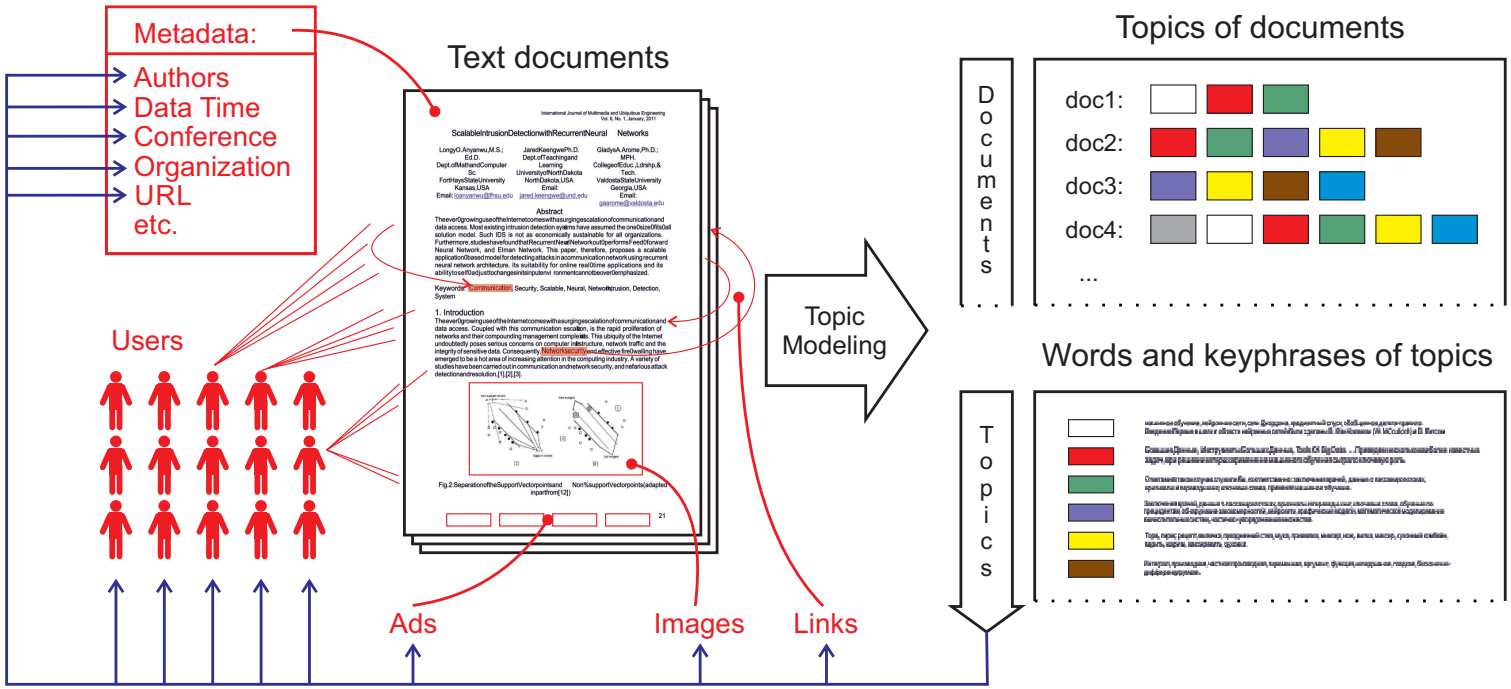
Существует много регуляризаторов. В ARTM наиболее простые и популярные следующие:

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

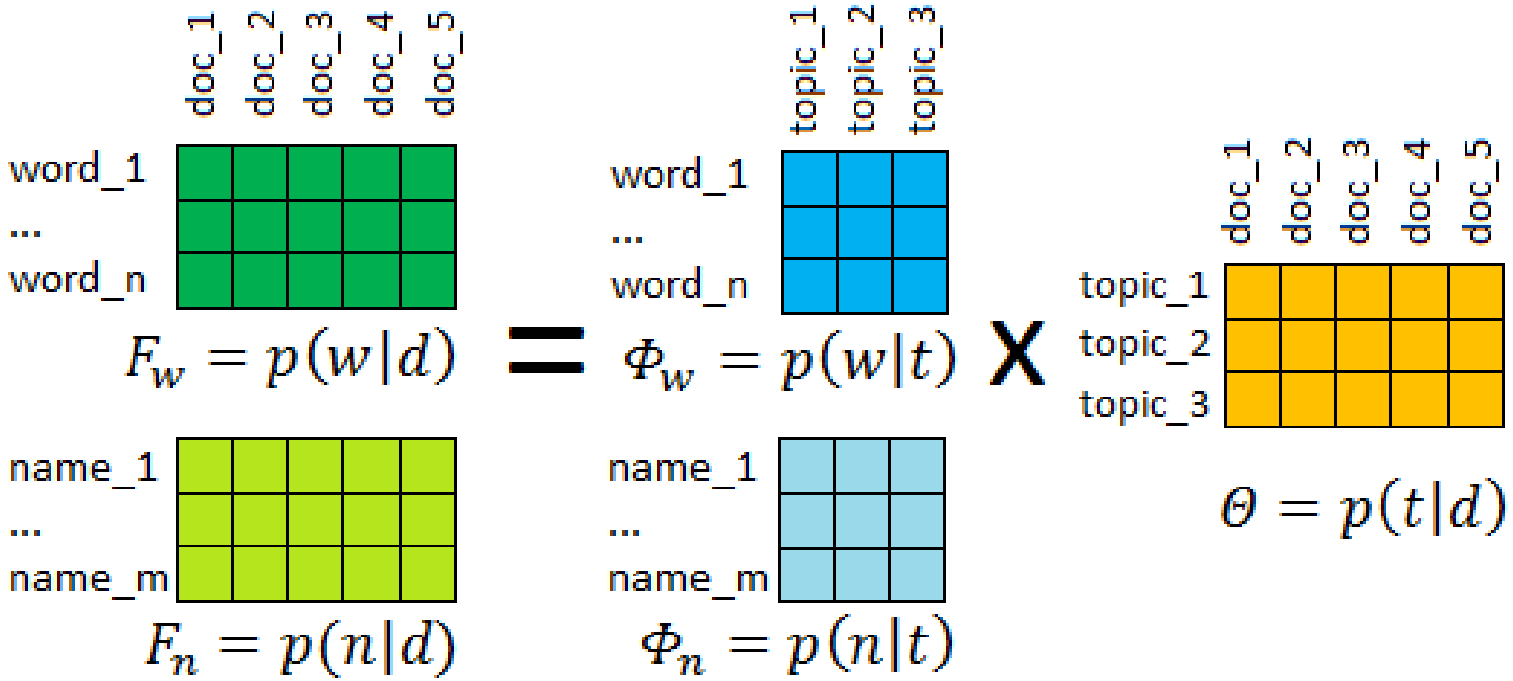


Мультимодальная TM строит распределения тем на терминах $p(w|t)$, авторах $p(a|t)$, метках времени $p(y|t)$, связанных документах $p(d'|t)$, рекламных баннерах $p(b|t)$, пользователях $p(u|t)$, и объединяет все эти модальности в одно тематическую модель.

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:

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

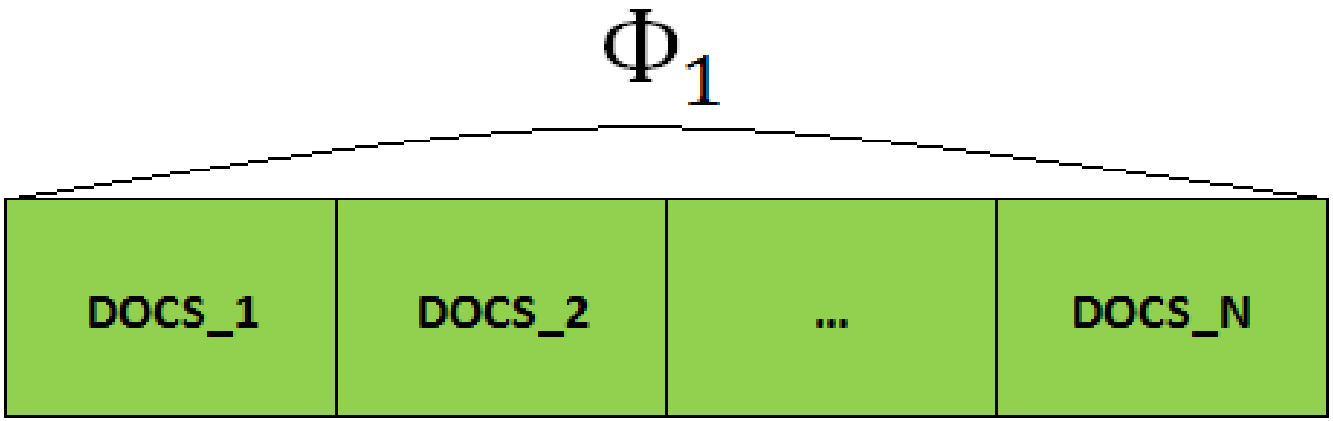
EM-алгоритм: метод простых итерация для системы уравнений

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

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

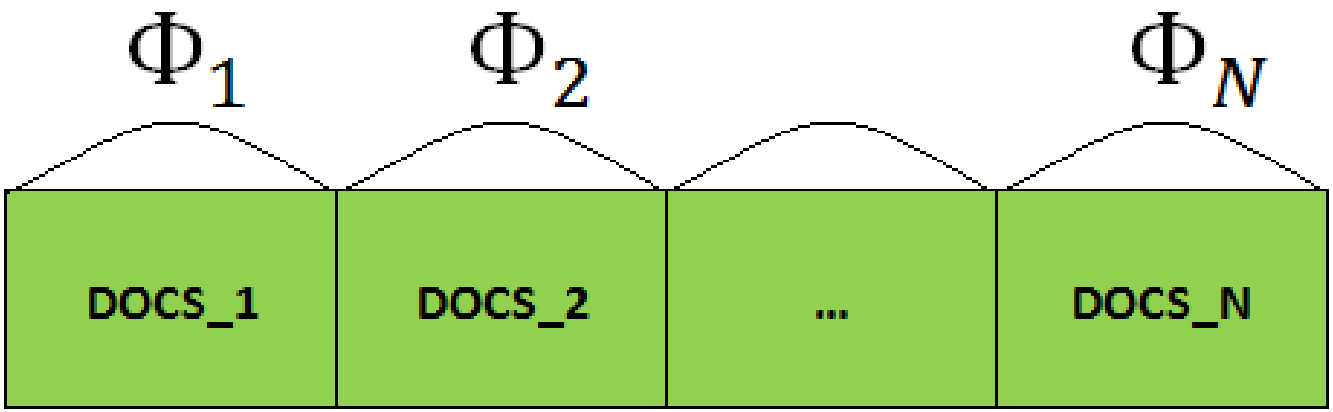
Оффлайн EM-алгоритм

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.

Про словари в BigARTM

В текстовом виде Dictionary представляет собой набор строк, каждая строка (кроме первой заголовочной) соответствует одному уникальному слову из словаря коллекции.

Строка имеет следующий формат:

`token  modality  value  tf  df`

1. Первые два элемента — это само слово в виде строки и его модальность, последние два — значения tf и df данного слова. Все эти значения считаются библиотекой в процессе парсинга.

2. Поле `value` тоже считается при парсинге, и представляет собой нормированное значение tf. Но его можно переопределять. Оно используется в регуляризаторе `SmoothSparsePhi` как дополнительный коэффициент регуляризации для данного слова.

Про словари в BigARTM

Словари в BigARTM играют огромную роль, они используются:

- инициализации тематической модели;
- в работе некоторых метрик качества;
- в работе некоторых регуляризаторов.

Словарь в Python можно сохранить на диск методом `artm.Dictionary.save_text(filename)`, отредактировать и загрузить обратно двойственным методом `load_text()`.

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

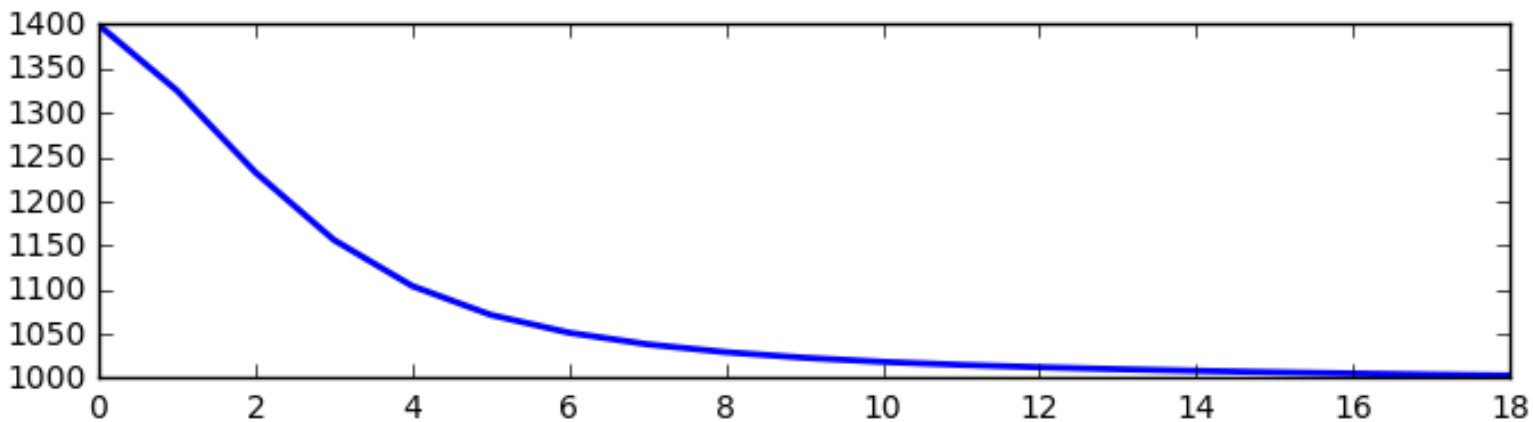
Пример: модель PLSA

Обучим модель PLSA с помощью оффлайн-алгоритма:

```
import artm

bv = artm.BatchVectorizer(data_path='vw.mmro.txt',
                          data_format='vowpal_wabbit',
                          batch_size=1000,
                          target_folder='my_batches',
                          gather_dictionary=True)

model = artm.ARTM(num_topics=10,
                  cache_theta=True,
                  dictionary=bv.dictionary)

model.scores.add(
    artm.PerplexityScore(name='perp',
                         dictionary=bv.dictionary))
```

Модель PLSA – обучение

```
%matplotlib inline
import matplotlib.pyplot as plt

model.fit_offline(batch_vectorizer=bv,
                  num_collection_passes=20)

perp_values = model.score_tracker['perp'].value[1: ]
plt.figure(figsize=(8,2))
plt.plot(xrange(len(perp_values)), perp_values)
```



Пример: модель PLSA – обучение

```
model.scores.add(
    artm.TopTokensScore(name='top_tok',
                        num_tokens=8))

model.fit_offline(batch_vectorizer=bv,
                  num_collection_passes=30)

tokens = model.score_tracker['top_tok'].last_tokens
for topic_name in model.topic_names:
    print '{}: '.format(topic_name),
    for token in tokens[topic_name]:
        print token,
    print
```

Пример: модель PLSA – оценивание

topic_0:	объект признак класс распознавание множество классификация образ пространство
topic_1:	последовательность слово метод признак который быть текст структура
topic_2:	алгоритм оценка выборка множество ошибка обучение метод обучать
topic_3:	система дать модель решение анализ который метод задача
topic_4:	задача решение множество алгоритм вектор число последовательность который
topic_5:	функция метод параметр преобразование система оценка быть значение
topic_6:	сигнал дать анализ метод быть результат обработка время

Пример: проблемы

В целом, темы дают представление о том, о чём говорится в коллекции (это тексты статей конференции ММРО).

Но! Никакой конкретики. По топ-словам тем нельзя с уверенностью сказать о том, о чём именно они.

Почему так получилось:

1. Документов в коллекции довольно мало (1062).
2. Коллекция не была должным образом фильтрована (слово «быть» — стоп-слово).
3. В модели слишком мало тем.
4. Было проделано недостаточное количество итераций EM-алгоритма.
5. Не были использованы регуляризаторы.
6. Использовались только униграммы (без качественных биграмм).

Пример: модель ARTM – обучение

Обучим модель тем же кодом, изменив три вещи:

1. зададим большее количество тем (50);
2. добавим регуляризатор декорреляции тем ($\tau = 10^5$);
3. увеличим количество итераций по коллекции (100).

```
model.regularizers.add(
    artm.DecorrelatorPhiRegularizer(name='decor',
                                    tau=1e+5))
```

Задача подбора регуляризаторов, их коэффициентов и траектории регуляризации до сих пор является открытой проблемой.

Как правило, это требует некоторого опыта, чутья и использования накопленных эвристик и соображений.

Пример: модель ARTM – оценивание

Темы получились несколько лучше. Здесь приведены наиболее качественные, но и это далеко не предел. Использование биграмм могло бы улучшить результат колоссально.

кластер кластеризация тренд центр кластерный нормировка средний тест

анализ время ряд временной момент основа интервал особенность

алгоритм оценка выборка множество ошибка обучение метод обучать

диагностика заболевание больной диагностический врач медицинский жизнь пациент

платон идеальный математика внимание идеал ступень идея ткань

цена рынок участник индекс мировой финансовый фондовый торг

сигнал спектр частота источник активность магнитный мозг частотный

Что нужно, кроме BigARTM

Помимо самого пакета BigARTM, установленного и настроенного для работы из Python, желательно уметь пользоваться следующими инструментами:

- Jupyter Notebook
- Лемматизаторы (pymorphy2, pymystem)
- Базовые средства обработки текстов из nltk
- Модули numpy, pandas, re и matplotlib
- Программы для просмотра больших текстовых файлов (Windows: emeditor, Linux/MacOS: less)

Постановка реальной задачи

Дано:

- Коллекция постов сайта LiveJournal.
- Словарь этнонимов (слов, связанных с различными этничностями).

Задача: выявить как можно большее количество качественных тем, связанных с этно-проблемами.

Метрика качества: оценки асессоров.

Параметры коллекции

Параметры коллекции:

- 1.58 млн. документов в виде «мешка слов»;
- 860 тыс. слов словаре;
- коллекция прошла лемматизацию.

Особенности:

- много слов с ошибками;
- коллекция русскоязычная, но присутствую термины на английском, украинском;
- много жаргонных слов и терминов специфических областей — сложно понимать и интерпретировать темы!

Подготовка данных

Парсим данные в формат Vowpal Wabbit, подходящий для BigARTM.
Сохраним только те слова, которые:

1. содержат только символы киррилицы и дефис;
2. содержат не более одного дефиса
   (есть слова вроде --, ----);
3. имеют длину не менее 3-х символов
   (есть слова вроде 'а', 'ж');
4. встречаются в коллекции не менее 20 раз;

Объём итогового словаря: 90 тыс слов.

В таких случаях бывают полезны регулярные выражения.

Составление словаря этнонимов

Описание проблемы:

- Имеется словарь из нескольких сотен этнонимов.
- Слова собраны в списки (например [абхаз, абхазец, абхазка])
- Пересечение слов этого словаря со словарём LJ непустое, но многие слова теряются.
- Нужно составить аналогичный словарь, специфичный для LJ.

Можно сделать вручную:

1. преобразовать списки всех слов в один линейный список;
2. пройтись по этому списку и для каждого слова найти все максимально похожие на него;
3. выбрать вручную в получившемся множестве все наиболее этничные слова, по 1-2 на каждый этноним исходного списка.

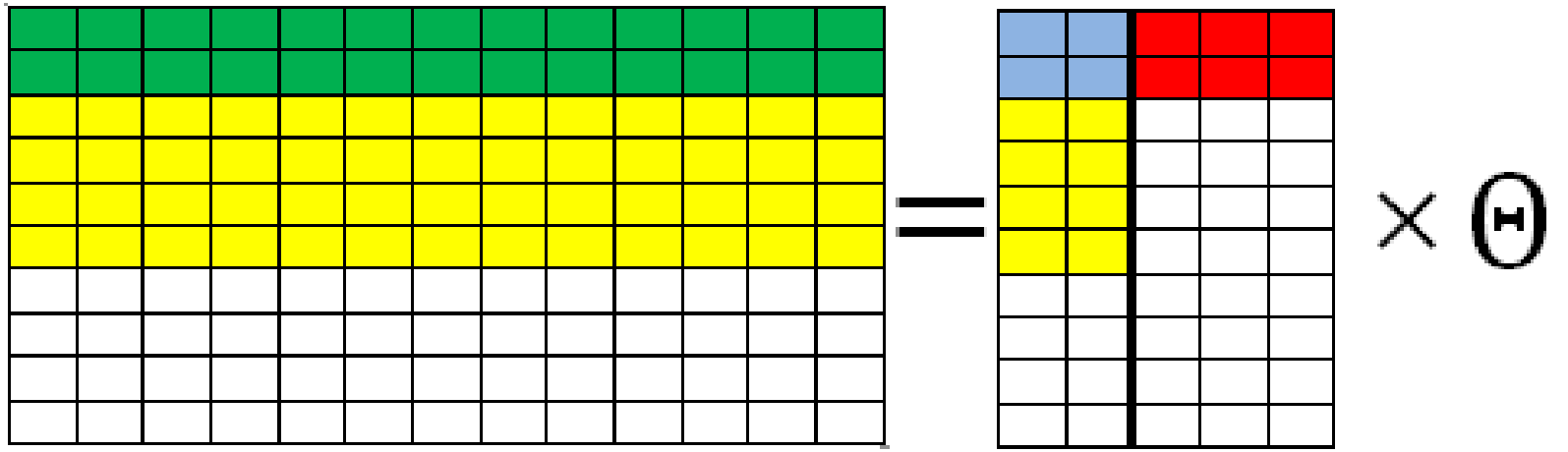
Объём итогового словаря этнонимов: 250 слов.

Примеры этнонимов

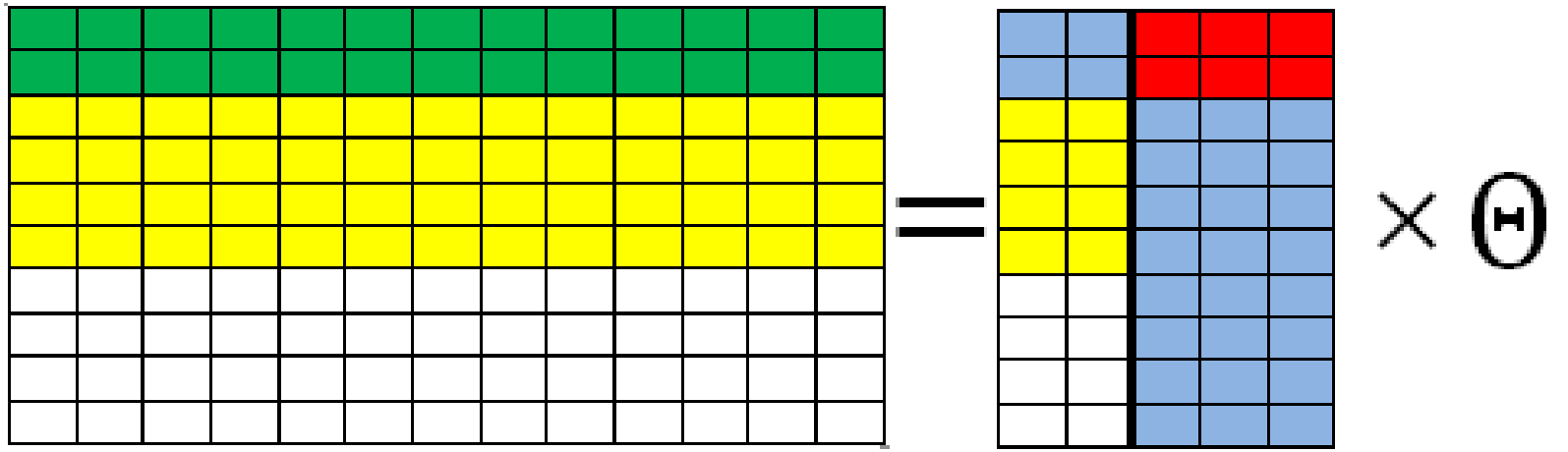
османский
восточноевропейский
эвенк
швейцарская
аланский
саамский
латыш
литовец
цыганка
ханты-мансийский
карачаевский
кубинка
гагаузский

русич
сингапурец
перуанский
словенский
вепсский
ниггер
адыги
сомалиец
абхаз
темнокожий
нигериец
лягушатник
камбоджиец

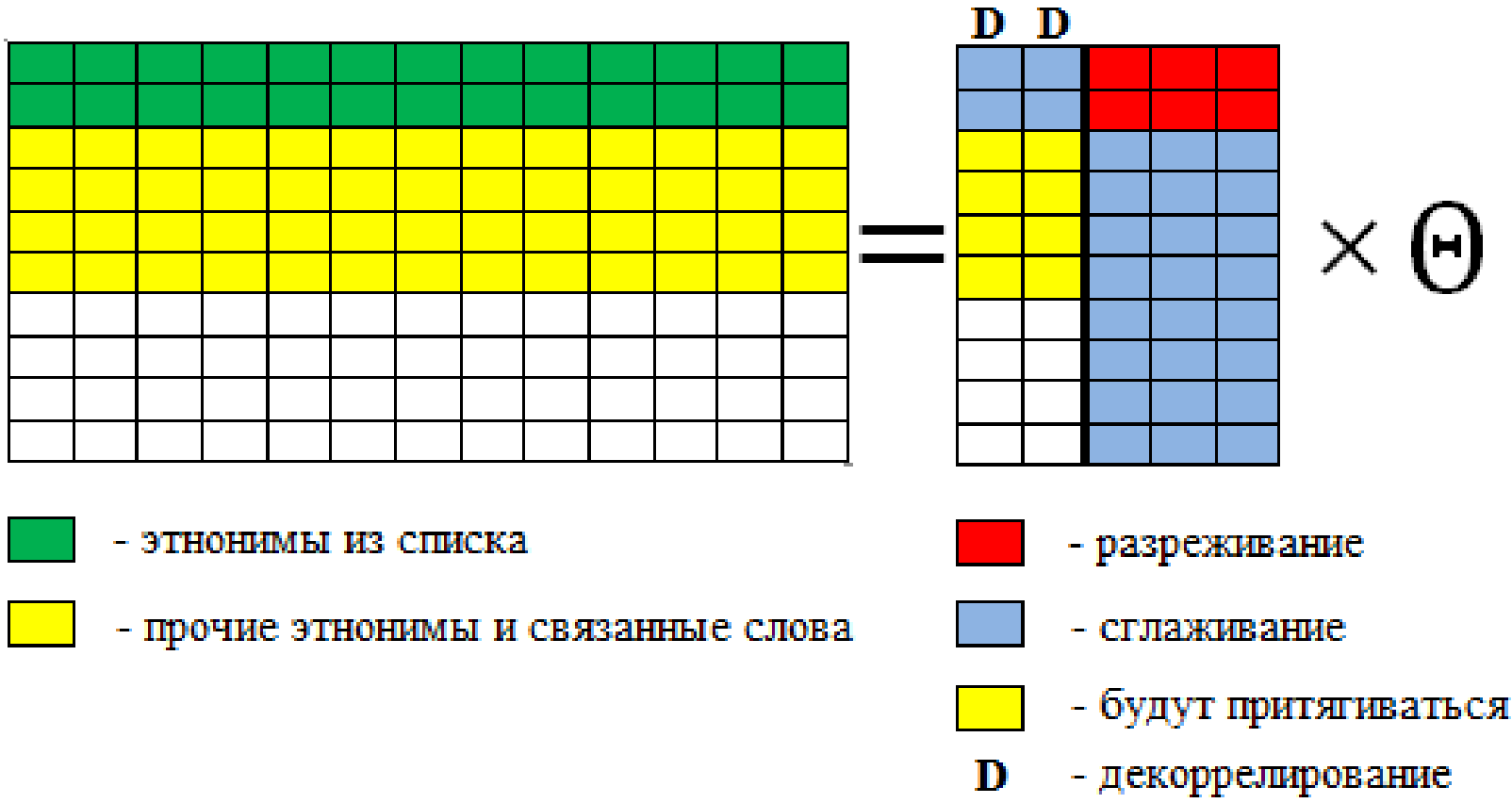
Сглаживание/разреживание этнонимов



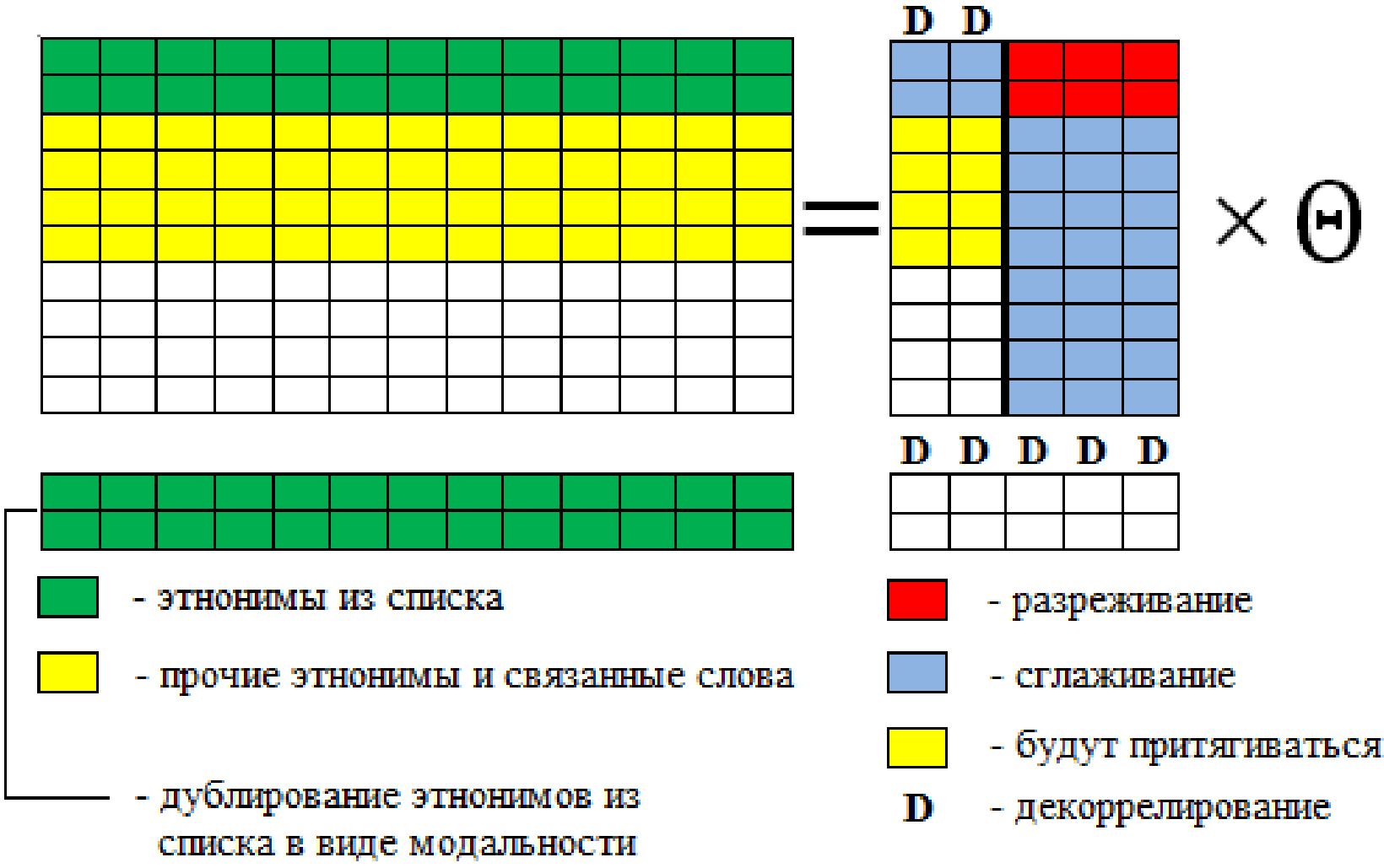
+ сглаживание обычных слов



+ декорреляция этничных тем



+ модальность этнонимов



Примеры лучших тем

(русские): акция, организация, митинг, движение, активный, мероприятие, совет, русский, участник, москва, оппозиция, россия, пикет, протест, проведение, националист, поддержка, общественный, проводить, участие,
(славяне, византийцы): славянский, святослав, жрец, древние, письменность, рюрик, летопись, византия, мефодий, хазарский, русский, азбука,
(сирийцы): сирийский, асад, боевик, район, террорист, уничтожать, группировка, дамаск, оружие, алесио, оппозиция, операция, селение, сша, нусра, турция,
(турки): турция, турецкий, курдский, эрдоган, стамбул, страна, кавказ, горин, полиция, премьер-министр, регион, курдистан, ататюрк, партия,
(иранцы): иран, иранский, сша, россия, ядерный, президент, тегеран, сирия, оон, израиль, переговоры, обама, санкция, исламский,
(палестинцы): террорист, израиль, терять, палестинский, палестинец, террористический, палестина, взрыв, территория, страна, государство, безопасность, арабский, организация, иерусалим, военный, полиция, газ,
(ливанцы): ливанский, боевик, район, ливан, армия, террорист, али, военный, хизбалла, раненый, уничтожать, сирия, подразделение, квартал, армейский,
(ливийцы): ливан, демократия, страна, ливийский, каддафи, государство, алжир, война, правительство, сша, арабский, али, муаммар, сирия,

(евреи): израиль, израильский, страна, израил, война, нетаньяху, тель-авив,

время, сша, сирия, египет, случай, самолет, еврейский, военный, ближний,

Некоторые результаты

Модель	Best	Good	Satisf.	Overall
PLSA (300)	9	11	18	38
PLSA (400)	12	15	17	44
С.Р.Д. (200+100)	18	33	20	71
С.Р.Д. (250+150)	21	27	20	68
С.Р.Д.М. (300+100)	28	23	23	74
С.Р.Д.М. (250+150)	22	25	33	80
С.Р.Д.М. (250+150) (tuned)	32	42	40	114

С – сглаживание, Р – разреживание,
Д - декорреляция, М – этномодальность

Что можно делать ещё?

- Эти эксперименты были продолжены на более крупной и сложной коллекции IQBuzz постов разных русскоязычных социальных медиа (в основном ВКонтакте).
- Был вручную собран новый, более полный и насыщенный существительными словарь этнонимов.
- Постановка задачи была усложнена: в дополнение для каждой релевантной темы требовалось исследовать её изменение в пространстве и времени.
- Для этого строились мультимодальные модели с дополнительными модальностями геотегов авторов, а также меток времени публикации сообщения.

Пример темы с привязкой ко времени и пространству

Топ-слова:
чеченский, чечня, кадыров, боевик, террорист, убийство, рамзан, грозный, спецназ, наемник, кавказ, погибать, операция, теракт, вооруженный, боевой, заложник, дудаев, лидер, командир

Топ-геотеги:
Москва, Санкт-Петербург, Чечня

Топ-метки времени:
Сосредоточены в начале и конце декабря 2014

Комментарий:
Совпадает с датой 20-тилетия начала войны в Чечне.

Данные IQBuzz охватывают период 2014-2015 годов.

Тем такого же качества - больше 10% от общего количества и примерно 30% от общего числа признанных этничными.

Итоги занятия

- Тематическое моделирование — инструмент статистического анализа текстов, позволяющий быстро получать информацию о затрагиваемых в них темах.
- Спектр задач, решаемых средствами ТМ, весьма широк.
- ТМ удобно использовать в качестве генератора признаков и кластеризатора.
- Существуют различные модели, наиболее общая — мультимодальная APTM.
- Наиболее быстрой реализацией ТМ для одной машины является BigARTM.

Спасибо за внимание!