Математические методы анализа текстов

Семинар 4

Тематическое моделирование

Мурат Апишев (great-mel@yandex.ru)

МГУ им. М. В. Ломоносова

23 марта, 2018

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.

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

Виды тематических моделей

- PLSA – простейшая модель без регуляризации
- LDA (+ модификации) – байесовская модель со сглаживанием, может дополняться одним-двумя проблемно-ориентированными регуляризаторами.
- ARTM – модель на основе PLSA, может дополняться любым числом регуляризаторов, позволяет использовать взвешенные модальности слов.

Основные предположения

- Порядок терминов в документе не важен (bag of words)
- Порядок документов в коллекции не важен (bag of docs)
- Каждый термин в документе связан с некоторой темой $t \in T$
- $D \times W \times T$ — дискретное вероятностное пространство
- Коллекция — это i.i.d. выборка $(d_i, w_i, t_i)_{i=1}^{n} \sim p(d, w, t)$
- d_i, w_i — наблюдаемые, темы t_i — скрытые
- гипотеза условной независимости: $p(w|d, t) = p(w|t)$

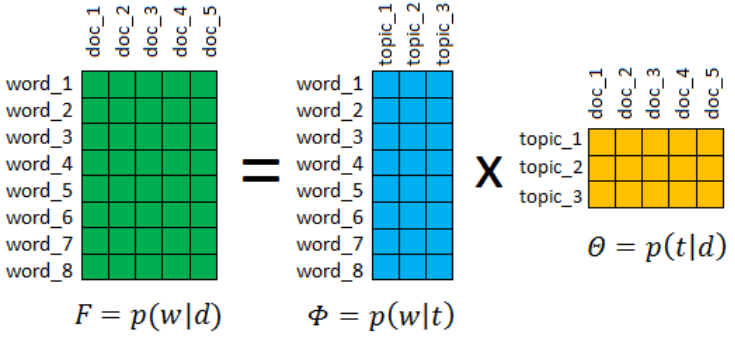
Предварительная обработка текста:

- Лемматизация (русский) или стемминг (английский)
- Выделение терминов (term extraction)
- Удаление стоп-слов и слишком редких слов

PLSA

Probabilistic Latent Semantic Analysis:

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



EM-алгоритм для модели PLSA

Максимизация логарифма правдоподобия (с ограничениями):

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

EM-алгоритм: метод простых итерация для решения системы уравнений

$$\begin{cases} \text{E-шаг:} & p_{tdw} = \underset{t\in T}{\operatorname{norm}}\big(\phi_{wt}\theta_{td}\big) \\ \text{M-шаг:} & \phi_{wt} = \underset{w\in W}{\operatorname{norm}}\big(n_{wt}\big), \quad n_{wt} = \sum_{d\in D} n_{dw}p_{tdw} \\ & \theta_{td} = \underset{t\in T}{\operatorname{norm}}\big(n_{td}\big), \quad n_{td} = \sum_{w\in d} n_{dw}p_{tdw} \end{cases}$$

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.

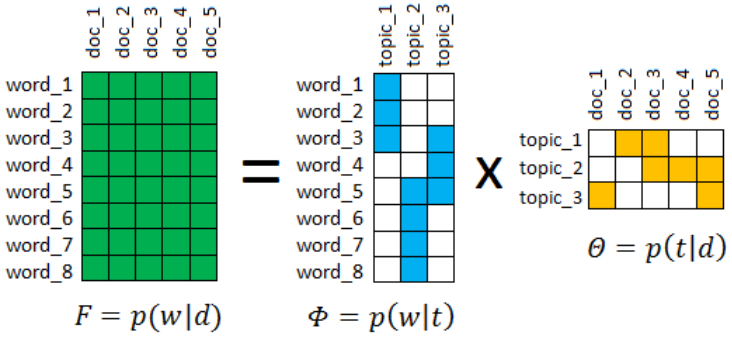
Но! Матричное разложение $F \approx \Phi \times \Theta$ имеет бесконечное множество решений: $\Phi\Theta = (\Phi S)(S^{-1}\Theta) = \Phi'\Theta' \Rightarrow$ можно подобрать Φ и Θ подходящего вида.

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

Пример регуляризации

Логичные предположения:

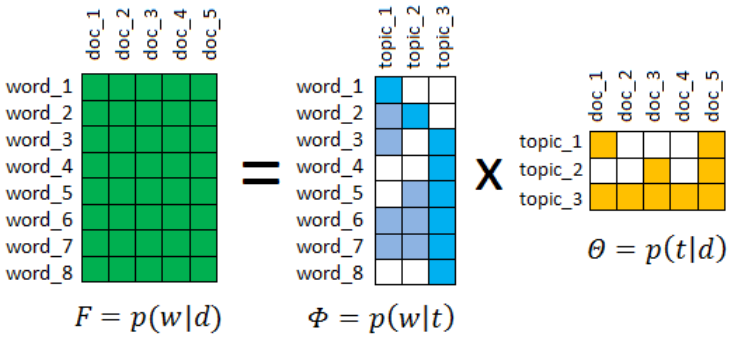
- темы должны состоять из небольшого числа слов, и эти множества слов не должны сильно пересекаться;
- каждый документ должен относиться к небольшому числу тем.



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».



ARTM

Additive Regularization of Topic Models:

Максимизация логарифма правдоподобия с дополнительными аддитивными регуляризаторами R:

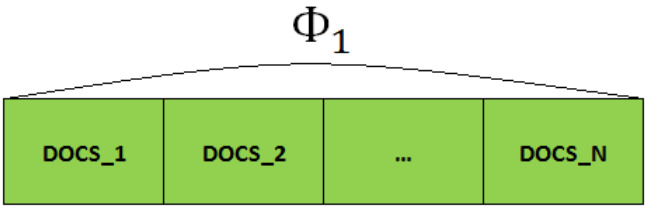
$$\sum_{d\in D}\sum_{w\in d} n_{dw} \ln \sum_t \phi_{wt}\theta_{td} + R(\Phi,\Theta) \;\to\; \max_{\Phi,\Theta}$$

EM-алгоритм: метод простых итераций для системы уравнений

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

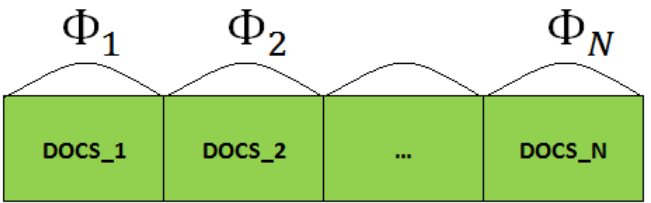
Оффлайн EM-алгоритм

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.



Онлайновый EM-алгоритм (BigARTM)

Вход: коллекция D, число тем $|T|$, параметры $i_{\max}$, $j_{\max}$, γ;
Выход: матрицы терминов тем Θ и тем документов Φ;

инициализировать $n_{wt} := 0$ и ϕ_{wt};
для всех $i = 1, \ldots, i_{\max}$ (для больших коллекций $i_{\max} = 1$)
  для всех документов $d \in D$
    инициализировать $\theta_{td} := \frac{1}{|T|}$;
    для всех $j = 1, \ldots, j_{\max}$ (итерации по документу)
      $n_{tdw} := n_{dw} \underset{t \in T}{\operatorname{norm}}\left(\phi_{wt}\theta_{td}\right)$ для всех $w \in d$;
      $\theta_{td} := \underset{t \in T}{\operatorname{norm}}\left(\sum_w n_{tdw} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}\right)$;
    $n_{wt} := \gamma n_{wt} + n_{tdw}$;
    если пора обновить матрицу Φ **то**
      $\phi_{wt} := \underset{w \in W}{\operatorname{norm}}\left(n_{wt} + \phi_{wt}\frac{\partial R}{\partial \phi_{wt}}\right)$;

Существующие реализации

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

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

```
1 lda = LatentDirichletAllocation(n_topics=20,
2                                 max_iter=50,
3                                 learning_method='batch')
4
5 lda.fit(cv.fit_transform(d))
6 tf_feature_names = cv.get_feature_names()
7 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

```
1  import gensim
2
3  corpus = gensim.matutils.Sparse2Corpus(cv.fit_transform(d),
4                                         documents\_columns=False)
5  id2word = {}
6  for index, token in enumerate(tf_feature_names):
7      id2word[index] = token
8
9  gensim.models.ldamodel.LdaModel(corpus=corpus,
10                                 id2word=id2word,
11                                 num_topics=\green{20})
12 lda.print_topics(lda.num_topics)
```

Vowpal Wabbit LDA

✗ Однопоточный

✗ Не совместим в векторайзерами sklearn

✓ Достаточно быстрый

Подготовка данных:

```
1  n_wd = array(corpus.sparse.todense())
2  num_docs = n_wd.shape[1]
3
4  with open('corpus.vw.txt', 'w') as fout:
5      for doc_id in xrange(num_docs):
6          fout.write('| ')
7          for token_id in xrange(1000):
8              value = n_wd[token_id][doc_id]
9              if value:
10                 fout.write('{}:{} '}}.format(id2word[token_id],
11                                                 value))
12         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

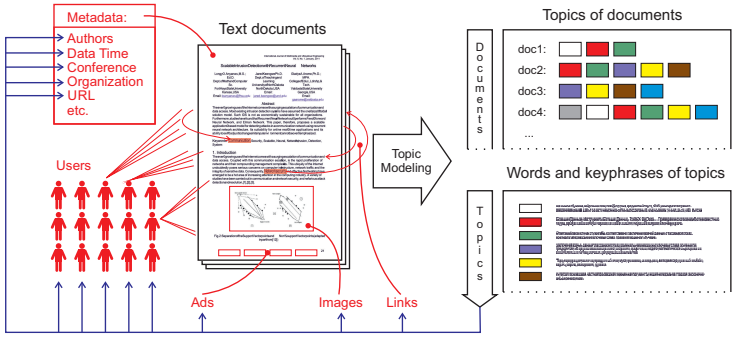
В Gensim есть обёртка на VW LDA, принимающая на вход данные в формате Gensim:

```
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'
```

Мультимодальная тематическая модель

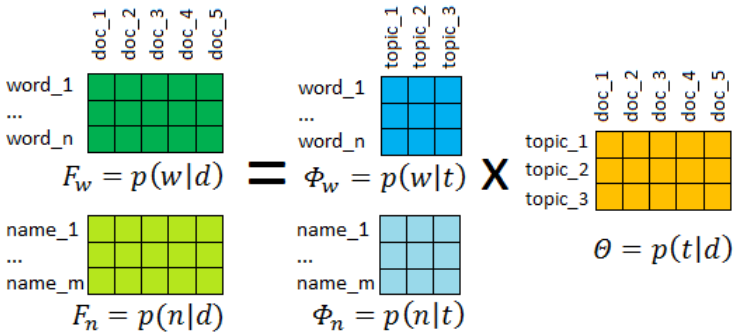


Мультимодальная TM строит распределения тем на терминах $p(w|t)$, авторах $p(a|t)$, метках времени $p(y|t)$, связанных документах $p(d'|t)$, рекламных баннерах $p(b|t)$, пользователях $p(u|t)$, и объединяет все эти модальности в одно тематическую модель.

Пример

Пусть у нас есть две модальности:

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



BigARTM

- ✓ Совместим в векторайзерами sklearn
- ✓ Позволяет строить сложные модели
- ✓ Параллельный, онлайновый, 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.

Формат VW: файл со строками вида (строка = документ)

doc1 Alpha Bravo:10 Charlie:5 |author Ola_Nordmann

doc2 Bravo:5 Delta Echo:3 |author Ivan_Ivanov

Запуск: `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:

```
1  import artm
2  n_wd = array(cv.fit_transform(d).todense()).T
3  vocabulary = cv.get_feature_names()
4
5  bv = artm.BatchVectorizer(data_format='bow_n_wd', n_wd=n_wd,
6                            vocabulary=vocabulary
7
8  model = artm.LDA(num_topics=20, dictionary=bv.dictionary)
9  model.fit_offline(bv, num_collection_passes=40)
10
11 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

Каждому регуляризатору соответствует свой класс в модуле artm:

```
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:

```
\texttt{m.score\_tracker[{\color{mymauve}'PERP\_SCR'}]}.value}
```

BigARTM vs. Gensim vs. VW LDA

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

	Gensim	VW-LDA	BigARTM	BigARTM (async)
P = 1, T= 50	142m (4945)	50m (5413)	42m (5117)	25m (5131)
P = 1, T= 100	287m (3969)	91m (4592)	52m (4093)	32m (4133)
P = 1, T= 200	637m (3241)	154m (3960)	83m (3347)	53m (3362)
P = 2, T= 50	89m (5056)		22m (5092)	13m (5160)
P = 2, T= 100	143m (4012)		29m (4107)	19m (4144)
P = 2, T= 200	325m (3297)		47m (3347)	28m (3380)
P = 4, T= 50	88m (5311)		12m (5216)	7m (5353)
P = 4, T= 100	104m (4338)		16m (4233)	10m (4357)
P = 4, T= 200	315m (3583)		26m (3520)	16m (3634)
P = 8, T= 50	88m (6344)		8m (5648)	5m (6220)
P = 8, T= 100	107m (5380)		10m (4660)	6m (5119)
P = 8, T= 200	288m (4263)		15m (3929)	10m (4309)

Framework/Topics	2000	5000
BigARTM	166m (2377)	399m (1942)
BigARTM (async)	119m (2645)	281m (2216)

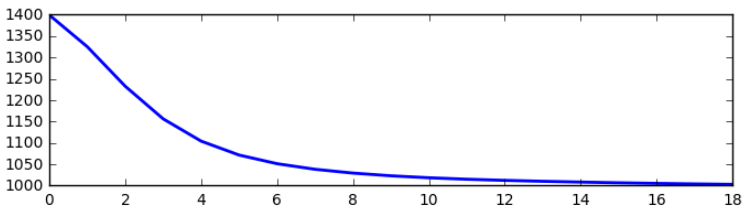
Пример: модель PLSA

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))
```

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
```

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

PLSA: 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).

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.

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: один файл, на каждый документ по одной строке вида
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.

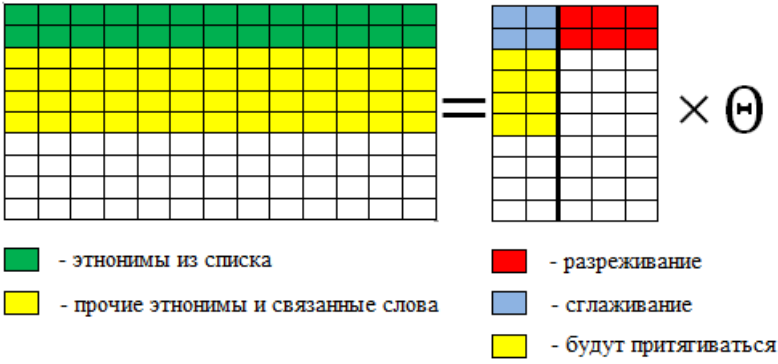
В таких случаях бывают полезны регулярные выражения.

Примеры этнонимов

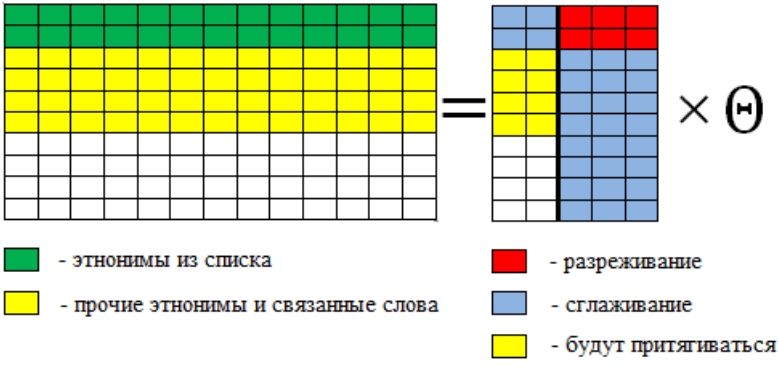
османский
восточноевропейский
эвенк
швейцарская
аланский
саамский
латыш
литовец
цыганка
ханты-мансийский
карачаевский
кубинка
гагаузский

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

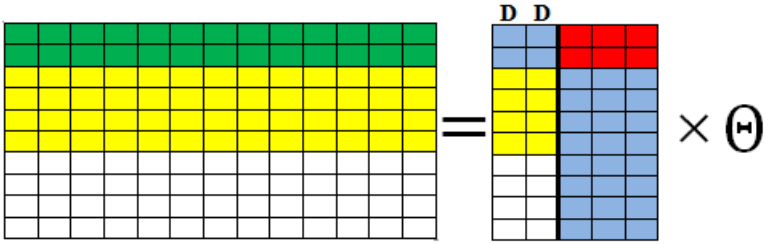
Сглаживание/разреживание этнонимов



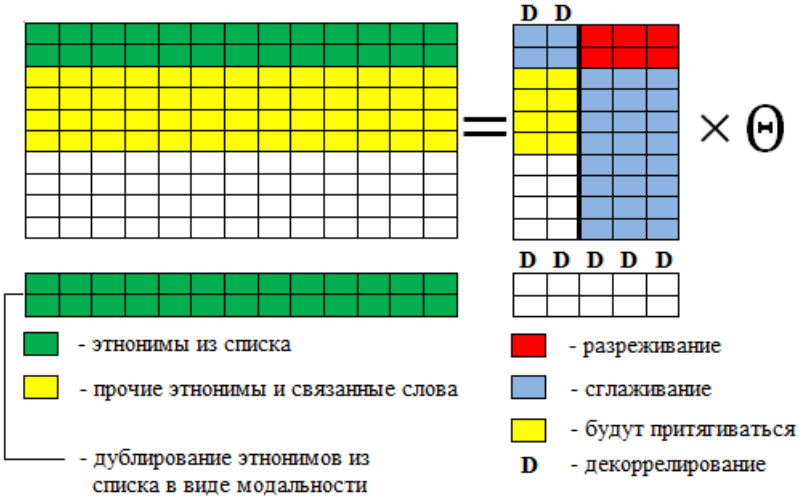
+ сглаживание обычных слов



+ 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.

Успехов!