

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

Кафедра математической кибернетики и компьютерных наук

ТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ НОВОСТЕЙ
КУРСОВАЯ РАБОТА

студента 3 курса 351 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Кондрашова Даниила Владиславовича

Научный руководитель
доцент, к. ф.-м. н.

С. В. Папшев

Заведующий кафедрой
к. ф.-м. н.

С. В. Миронов

Саратов 2024

СОДЕРЖАНИЕ

| | |
|---|----|
| ВВЕДЕНИЕ | 4 |
| 1 Математические основы тематического моделирования | 5 |
| 1.1 Основная гипотеза тематического моделирования | 5 |
| 1.2 Аксиоматика тематического моделирования | 5 |
| 1.3 Задача тематического моделирования | 6 |
| 1.4 Решение обратной задачи | 7 |
| 1.4.1 Лемма о максимизации функции на единичных симплексах | 7 |
| 1.4.2 Сведение обратной задачи к задаче максимизации функ- ционала | 8 |
| 1.4.3 Аддитивная регуляризация тематических моделей | 9 |
| 1.4.4 E-M алгоритм | 9 |
| 1.5 Регуляризаторы в тематическом моделировании | 10 |
| 1.5.1 Дивергенция Кульбака-Лейблера | 10 |
| 1.5.2 Регуляризатор сглаживания | 11 |
| 1.5.3 Регуляризатор разреживания | 12 |
| 1.5.4 Регуляризатор декоррелирования тем | 12 |
| 1.6 Оценка качества моделей тематического моделирования | 13 |
| 1.6.1 Правдоподобия и перплексия | 14 |
| 1.6.2 Когерентность | 14 |
| 1.6.3 Разреженность | 15 |
| 1.6.4 Чистота темы | 15 |
| 1.6.5 Контрастность темы | 15 |
| 2 Тематическое моделирование новостей | 16 |
| 2.1 Предобработка текстов | 16 |
| 2.1.1 Токенизация, перевод в нижний регистр и удаление неал- фавитных символов | 16 |
| 2.1.2 Удаление стоп-слов | 17 |
| 2.1.3 Лемматизация | 18 |
| 2.1.4 Создание N-грамм | 18 |
| 2.2 Статистика по данным | 19 |
| 2.2.1 Создание тематической модели с помощью библиотеки BigARTM | 20 |
| 2.3 PLSA (модель без регуляризаторов) | 22 |

| | | |
|--|--|----|
| 2.4 | LDA (модель с регуляризатором сглаживания) | 22 |
| 2.5 | Модель с регуляризатором разреживания | 23 |
| 2.6 | Модель с регуляризатором декоррелирования | 24 |
| 2.7 | Выбор лучшей модели | 24 |
| ЗАКЛЮЧЕНИЕ | | 25 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | | 26 |

ВВЕДЕНИЕ

С ростом объёмов информации в современном мире умение классифицировать и структурировать данные становится необходимым для их эффективного поиска и изучения. Физически невозможно найти нужные сведения, просто перебирая все ресурсы подряд, поэтому возникает острая потребность в тематическом поиске и классификации данных.

Тематическое моделирование призвано решить эту проблему. Оно позволяет быстро и эффективно автоматически разбивать большие объёмы информации по темам, упрощая процесс поиска и анализа данных.

1 Математические основы тематического моделирования

1.1 Основная гипотеза тематического моделирования

Тематическое моделирование — это метод анализа текстовых данных, который позволяет выявлять семантические структуры в коллекциях документов.

Основная идея тематического моделирования заключается в том, что слова в тексте связаны не с конкретным документом, а с темами. Сначала текст разбивается на темы, и каждая из них генерирует слова для соответствующих позиций в документе. Таким образом, сначала формируется тема, а затем тема формирует терм.

Эта гипотеза позволяет проводить тематическую классификацию текстов на основе частоты и встречаемости слов [1–3].

1.2 Аксиоматика тематического моделирования

Каждый текст можно количественно охарактеризовать. Вот основные количественные характеристики, используемые при тематическом моделировании:

- W — конечное множество термов;
- D — конечное множество текстовых документов;
- T — конечное множество тем;
- $D \times W \times T$ — дискретное вероятностное пространство;
- коллекция — i.i.d выборка $(d_i, w_i, t_i)_{i=1}^n$;
- $n_{dwt} = \sum_{i=1}^n [d_i = d][w_i = w][t_i = t]$ — частота (d, w, t) в коллекции;
- $n_{wt} = \sum_d n_{dwt}$ — частота термина w в документе d ;
- $n_{td} = \sum_w n_{dwt}$ — частота термов темы t в документе d ;
- $n_t = \sum_{d,w} n_{dwt}$ — частота термов темы t в коллекции;
- $n_{dw} = \sum_t n_{dwt}$ — частота термина w в документе d ;
- $n_W = \sum_d n_{dw}$ — частота термина w в коллекции;
- $n_d = \sum_w n_{dw}$ — длина документа d ;
- $n = \sum_{d,w} n_{dw}$ — длина коллекции.

Также в тематическом моделировании используются следующие гипотезы и аксиомы:

- Независимость слов от порядка в документе: порядок слов в документе не важен;
- Независимость от порядка документов в коллекции: порядок документов

в коллекции не важен;

- Зависимость термина от темы: каждый терм связан с соответствующей темой и порождается ей;
- Гипотеза условной независимости: $p(w|d, t) = p(w|t)$.

Вышеперечисленные характеристики, гипотезы и аксиомы составляют основу тематического моделирования и являются достаточными для построения тематической модели. [1–4].

1.3 Задача тематического моделирования

Как уже говорилось ранее, документ порождается следующим образом:

1. для каждой позиции в документе генерируется тема $p(t|d)$;
2. для каждой сгенерированной темы в соответствующей позиции генерируем терм $p(w|d, t)$.

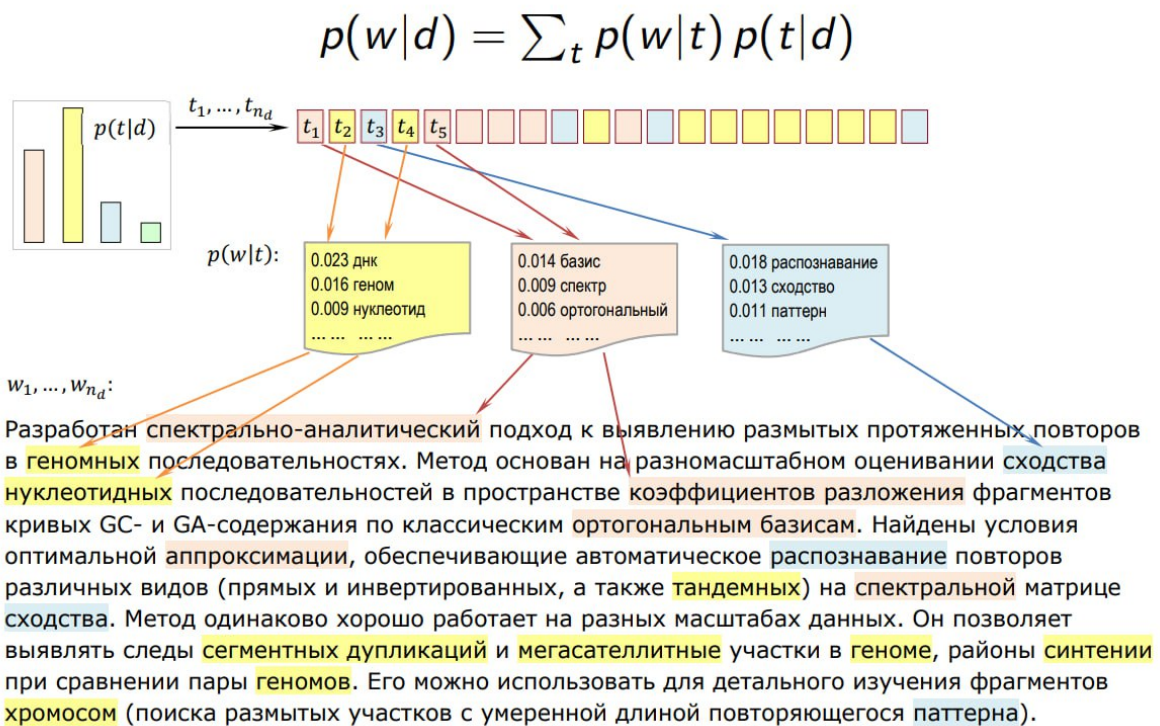


Рисунок 1 – Алгоритм формирования документа

Тогда вероятность появления слова в документе можно описать по формуле полной вероятности:

$$p(w|d) = \sum_{t \in T} p(w|d, t) p(t|d) = \sum_{t \in T} p(w|t) p(t|d) \quad (1)$$

Такой алгоритм является прямой задачей порождения текста. Тематическое моделирование призвано решить обратную задачу:

1. для каждого термина w в тексте найти вероятность появления в теме t (найти $p(w|t) = \phi_{wt}$);
2. для каждой темы t найти вероятность появления в документе d (найти $p(t|d) = \theta_{td}$).

Обратную задачу можно представить в виде стохастического матричного разложения 2.

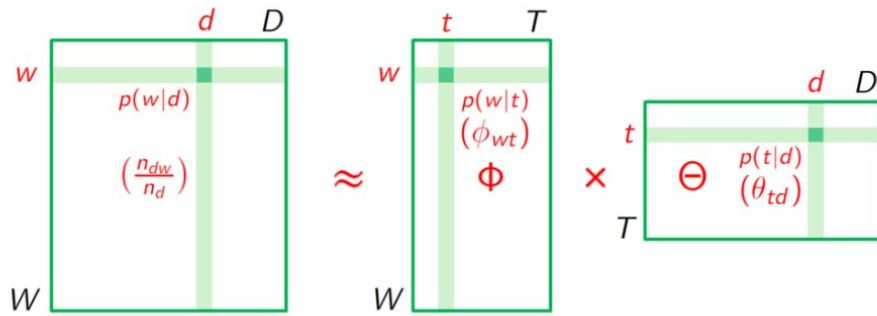


Рисунок 2 – Стохастическое матричное разложение

Таким образом, тематическое моделирование ищет величину $p(w|d)$ [1–4].

1.4 Решение обратной задачи

Для решения задачи тематического моделирования необходимо найти величину $p(w|d)$, сделать это можно с помощью метода максимального правдоподобия.

1.4.1 Лемма о максимизации функции на единичных симплексах

Перед тем как перейти к решению обратной задачи, сформулируем лемму, которая поможет нам в этом процессе.

Введём операцию нормировки вектора:

$$p_i = \left(\begin{matrix} x_i \\ \sum_{k \in I} \max(x_k, 0) \end{matrix} \right) = \frac{\max(x_i, 0)}{\sum_{k \in I} \max(x_k, 0)} \quad (2)$$

Лемма о максимизации функции на единичных симплексах:

Пусть функция $f(\Omega)$ непрерывно дифференцируема по набору векторов $\Omega = (w_i)_{i \in J}$, $w_j = (w_{ij})_{i \in I_j}$ различных размерностей $|I_j|$. Тогда векторы w_j

локального экстремума задачи

$$\begin{cases} f(\Omega) \rightarrow \max_{\Omega} \\ \sum_{i \in I_j} w_{ij} = 1, \quad j \in J \\ w_{ij} \geq 0, \quad i \in I_j, j \in J \end{cases}$$

при условии 1^0 : $(\exists i \in I_j) w_{ij} \frac{\partial f}{\partial w_{ij}} > 0$ удовлетворяют уравнениям

$$w_{ij} = \underset{i \in I_j}{\text{norm}} \left(w_{ij} \frac{\partial f}{\partial w_{ij}} \right), \quad i \in I_j; \quad (3)$$

при условии 2^0 : $(\forall i \in I_j) w_{ij} \frac{\partial f}{\partial w_{ij}} \leq 0$ и $(\exists i \in I_j) w_{ij} \frac{\partial f}{\partial w_{ij}} < 0$ удовлетворяют уравнениям

$$w_{ij} = \underset{i \in I_j}{\text{norm}} \left(-w_{ij} \frac{\partial f}{\partial w_{ij}} \right), \quad i \in I_j; \quad (4)$$

в противном случае (условие 3^0) — однородным уравнениям

$$w_{ij} \frac{\partial f}{\partial w_{ij}} = 0, \quad i \in I_j. \quad (5)$$

Данная лемма служит для оптимизации любых моделей, параметрами которых являются неотрицательные нормированные векторы [1–3].

1.4.2 Сведение обратной задачи к задаче максимизации функционала

Чтобы вычислить величину $p(w|d)$ воспользуемся принципом максимума правдоподобия, согласно которому будут подобраны параметры Φ, Θ такие, что $p(w|d)$ примет наибольшее значение.

$$\prod_{i=1}^n p(d_i, w_i) = \prod_{d \in D} \prod_{w \in d} p(d, w)^{n_{dw}} \quad (6)$$

Прологарифмировав правдоподобие, перейдём к задаче максимизации логарифма правдоподобия.

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \ln p(w|d) \underset{const}{\rightarrow \max} = n_{dw} \rightarrow \max \quad (7)$$

Данная задача эквивалентна задаче максимизации функционала

$$L(\Phi, \Theta) = \sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} \rightarrow \max_{\Phi, \Theta} \quad (8)$$

при ограничениях неотрицательности и нормировки

$$\phi_{wt} \geq 0; \quad \sum_{w \in W} \phi_{wt} = 1; \quad \theta_{td} \geq 0; \quad \sum_{t \in T} \theta_{td} = 1 \quad (9)$$

Таким образом, обратная задача сводится к задаче максимизации функции [1–4].

1.4.3 Аддитивная регуляризация тематических моделей

Задача [?] не соответствует критериям корректно поставленной задачи по Адамару, поскольку в общем случае она имеет бесконечное множество решений. Это свидетельствует о необходимости доопределения задачи.

Для доопределения некорректно поставленных задач применяется регуляризация: к основному критерию добавляется дополнительный критерий — регуляризатор, который соответствует специфике решаемой задачи.

Метод ARTM (аддитивная регуляризация тематических моделей) основывается на максимизации линейной комбинации логарифма правдоподобия и регуляризаторов $R_i(\Phi, \Theta)$ с неотрицательными коэффициентами регуляризации $t\tau_i$, $i = 1, \dots, k$.

Преобразуем задачу к ARTM виду:

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} + R(\Phi, \Theta) \rightarrow \max_{\Phi, \Theta}; \quad R(\Phi, \Theta) = \sum_{i=1}^k \tau_i R_i(\Phi, \Theta) \quad (10)$$

при ограничениях неотрицательности и нормировки 9.

Регуляризатор (или набор регуляризаторов) выбирается в соответствии с решаемой задачей [1–3].

1.4.4 Е-М алгоритм

Из представленных ограничений 9 следует, что столбцы матриц можно считать неотрицательными единичными векторами. Таким образом, задача сводится к максимизации функции на единичных симплексах.

Воспользуемся леммой о максимизации функции на единичных симплексах 1.4.1 и перепишем задачу.

Пусть функция $R(\Phi, \Theta)$ непрерывно дифференцируема. Тогда точка (Φ, Θ) локального экстремума задачи с ограничениями, удовлетворяет системе уравнений с вспомогательными переменными $p_{tdw} = p(t|d, w)$, если из решения исключить нулевые столбцы матриц Φ и Θ :

$$\begin{cases} p_{tdw} = \underset{t \in T}{\text{norm}}(\phi_{wt}\theta_{td}) \\ \phi_{wt} = \underset{w \in W}{\text{norm}}\left(n_{wt} + \phi_{wt} \frac{\partial R}{\partial \phi_{wt}}\right); \\ \theta_{td} = \underset{t \in T}{\text{norm}}\left(n_{td} + \theta_{td} \frac{\partial R}{\partial \theta_{td}}\right) \end{cases} \quad (11)$$

Полученная модель соответствует Е-М алгоритму, где первая строка системы уравнений соответствует Е шагу, а вторая и третья строки — М шагу.

Решив полученную систему уравнений, методом простых итерации получим искомые матрицы Φ и Θ [1–3].

1.5 Регуляризаторы в тематическом моделировании

В этом разделе будут рассмотрены некоторые возможные варианты регуляризаторов.

1.5.1 Дивергенция Кульбака-Лейблера

Перед тем как перейти к регуляризаторам необходимо ввести меру оценки близости тем.

Чтобы оценить близость тем можно воспользоваться дивергенцией Кульбака-Лейблера (KL или KL-дивергенция). KL-дивергенция позволяет оценить степень вложенности одного распределения в другое, в случае тематического моделирования будет оцениваться вложенность матриц.

Определим KL-дивергенцию:

Пусть $P = (p_i)_{i=1}^n$ и $Q = (q_i)_{i=1}^n$ некоторые распределения. Тогда дивергенция Кульбака-Лейблера имеет следующий вид:

$$KL(P||Q) = KL_i(p_i||q_i) = \sum_{i=1}^n p_i \ln \frac{p_i}{q_i}. \quad (12)$$

Свойства KL-дивергенции:

1. $KL(P||Q) \geq 0$;

$$2. KL(P||Q) = 0 \Leftrightarrow P = Q;$$

3. Минимизация KL эквивалентна максимизации правдоподобия:

$$KL(P||Q(\alpha)) = \sum_{i=1}^n p_i \ln \frac{p_i}{q_i(\alpha)} \rightarrow \min_{\alpha} \Leftrightarrow \sum_{i=1}^n p_i \ln q_i(\alpha) \rightarrow \max_{\alpha};$$

4. Если $KL(P||Q) < KL(Q||P)$, то P сильнее вложено в Q , чем Q в P .

Теперь можно перейти к рассмотрению регуляризаторов [1, 5, 6].

1.5.2 Регуляризатор сглаживания

Сглаживание предполагает семантическое сближение тем, это может быть полезно в следующих случаях:

1. Темы могут быть похожи между собой по терминологии, например, основы теории вероятностей и линейной алгебры обладают рядом одинаковых терминов;
2. При выделении фоновых тем важно максимально вобрать в них слова, следовательно, сглаживание поможет решить эту задачу.

Определим регуляризатор сглаживания:

Пусть распределения ϕ_{wt} близки к заданному распределению β_w и пусть распределения θ_{td} близки к заданному распределению α_t . Тогда в форме KL-дивергенции 1.5.1 выразим задачу сглаживания:

$$\sum_{t \in T} KL(\beta_w || \phi_{wt}) \rightarrow \min_{\Phi}; \quad \sum_{d \in D} KL(\alpha_t || \theta_{td}) \rightarrow \min_{\Theta}. \quad (13)$$

Согласно свойству 3 KL-дивергенции перейдём к задаче максимизации правдоподобия:

$$R(\Phi, \Theta) = \beta_o \sum_{t \in T} \sum_{w \in W} \beta_w \ln \phi_{wt} + \alpha_o \sum_{d \in D} \sum_{t \in T} \alpha_t \ln \theta_{td} \rightarrow \max. \quad (14)$$

Перепишем ЕМ-флгоритм 11 в соответствии с полученной формулой:

$$\begin{cases} p_{tdw} = \text{norm}_{t \in T}(\phi_{wt} \theta_{td}) \\ \phi_{wt} = \text{norm}_{w \in W}(n_{wt} + \beta_o \beta_w); \\ \theta_{td} = \text{norm}_{t \in T}(n_{td} + \alpha_o \alpha_t) \end{cases} \quad (15)$$

Таким образом был получен модифицированный ЕМ-алгоритм соответствующий модели LDA [1, 4–6].

1.5.3 Регуляризатор разреживания

Разреживание подразумевает разделение тем и документов, исключая общие слова из них. Этот тип регуляризации основывается на предположении, что темы и документы в основном являются специфичными и описываются относительно небольшим набором терминов, которые не встречаются в других темах.

Определим регуляризатор разреживания:

Пусть распределения ϕ_{wt} далеки от заданного распределения β_w и пусть распределения θ_{td} далеки от заданного распределения α_t . Тогда в форме KL-дивергенции 1.5.1 выразим задачу сглаживания:

$$\sum_{t \in T} KL(\beta_w || \phi_{wt}) \rightarrow \max_{\Phi}; \quad \sum_{d \in D} KL(\alpha_t || \theta_{td}) \rightarrow \max_{\Theta}. \quad (16)$$

Согласно свойству 3 KL-дивергенции перейдём к задаче максимизации правдоподобия:

$$R(\Phi, \Theta) = -\beta_o \sum_{t \in T} \sum_{w \in W} \beta_w \ln \phi_{wt} - \alpha_o \sum_{d \in D} \sum_{t \in T} \alpha_t \ln \theta_{td} \rightarrow \max. \quad (17)$$

Перепишем ЕМ-алгоритм 11 в соответствии с полученной формулой:

$$\begin{cases} p_{tdw} = \underset{t \in T}{\text{norm}}(\phi_{wt} \theta_{td}) \\ \phi_{wt} = \underset{w \in W}{\text{norm}}(n_{wt} - \beta_o \beta_w); \\ \theta_{td} = \underset{t \in T}{\text{norm}}(n_{td} - \alpha_o \alpha_t) \end{cases} \quad (18)$$

Таким образом был получен модифицированный ЕМ-алгоритм, разреживающий матрицы Φ и Θ [1, 4–6].

1.5.4 Регуляризатор декоррелирования тем

Декоррелятор тем — это частный случай разреживания, призванный выделить для каждой темы лексическое ядро — набор термов, отличающий её от других тем:

Определим регуляризатор декоррелирования:

Минимизируем ковариации между вектор-столбцами ϕ_t :

$$R(\Phi) = -\frac{\tau}{2} \sum_{t \in T} \sum_{s \in T \setminus t} \sum_{w \in W} \phi_{wt} \phi_{ws} \rightarrow \max. \quad (19)$$

Перепишем ЕМ-алгоритм 11 в соответствии с полученной формулой:

$$\begin{cases} p_{tdw} = \underset{t \in T}{\text{norm}}(\phi_{wt} \theta_{td}) \\ \phi_{wt} = \underset{w \in W}{\text{norm}} \left(n_{wt} - \tau \phi_{wt} \sum_{t \in T \setminus t} \phi_{ws} \right); \\ \theta_{td} = \underset{t \in T}{\text{norm}} \left(n_{td} + \theta_{td} \frac{\partial R}{\partial \theta_{td}} \right) \end{cases} \quad (20)$$

Таким образом был получен модифицированный ЕМ-алгоритм, декоррелирующий темы [1, 5, 6].

1.6 Оценка качества моделей тематического моделирования

После обучения модели, очевидно, нужно оценить её качество.

Перечислим основные критерии оценки качества тематических моделей:

1. Внешние критерии (оценка производится экспертами):
 - а) Полнота и точность тематического поиска;
 - б) Качество ранжирования при тематическом поиске;
 - в) Качество классификации / категоризации документов;
 - г) Качество суммаризации / сегментации документов;
 - д) Экспертные оценки качества тем.
2. Внутренние критерии (оценка производится программно):
 - а) Правдоподобие и перплексия;
 - б) Средняя когерентность (согласованность тем);
 - в) Разреженность матриц Φ и Θ ;
 - г) Различность тем;
 - д) Статистический тест условной независимости.

Поскольку оценка по внешним критериям невозможна в рамках данной работы, сосредоточимся на внутренних критериях оценки, которые можно вычислять автоматически [1, 7, 8].

1.6.1 Правдоподобия и перплексия

Перплексия основывается на логарифме правдоподобия и является его некоторой модификацией.

$$P(D) = \exp \left(-\frac{1}{n} \sum_{d \in D} \sum_{w \in d} n_{dw} \ln p(w|d) \right), \quad n = \sum_{d \in D} \sum_{w \in d} n_{dw} \quad (21)$$

Не трудно заметить, что при равномерном распределении слов в тексте выполняется равенство $p(w|d) = \frac{1}{|W|}$. В этом случае значение перплексии равно мощности словаря $P = |W|$. Это позволяет сделать вывод, что перплексия является мерой разнообразия и неопределенности слов в тексте: чем меньше значение перплексии, тем более разнообразны вероятности появления слов.

Таким образом, чем меньше перплексия, тем больше слов с большей вероятностью $p(w|d)$, которые модель умеет лучше предсказывать, следовательно, чем меньше перплексия, тем лучше [1, 4, 7, 8].

1.6.2 Когерентность

Когерентность является мерой, коррелирующей с экспертной оценкой интерпретируемости тем.

Когерентность (согласованность) темы t по k топовым словам:

$$PNI_t = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^k PMI(w_i, w_j), \quad (22)$$

где w_i — i -ое слово в порядке убывания ϕ_{wt} , $PMI(u, v) = \ln \frac{|D|N_{uv}}{N_u N_v}$ — пото-
чечная взаимная информация, N_{uv} — число документов, в которых слова u, v хотя бы один раз встречаются рядом (расстояние определяется отдельно), N_u — число документов, в которых u встретился хотя бы один раз.

Гипотезу когерентности можно выразить так: когда человек говорит о какой-либо теме, то часто употребляет достаточно ограниченный набор слов, относящийся к этой теме, следовательно, чем чаще будут встречаться вместе слова этой темы, тем лучше её можно будет интерпретировать.

Сама когерентность берёт самые часто встречающиеся слова из тем, и вычисляет для каждой пары из них насколько они часто встречаются, соответ-

ственно, чем выше будет значение взаимовстречаемости, тем лучше [1, 7, 8].

1.6.3 Разреженность

Разреженность — доля нулевых элементов в матрицах Φ и Θ .

Разреженность играет ключевую роль в выявлении различий между темами. Каждая тема формируется на основе ограниченного набора слов, в то время как остальные слова должны встречаться реже, что отражается в нулевых элементах матриц. Оптимальный уровень разреженности должен быть высоким, но не чрезмерным: в таком случае темы будут четко различимы. Если разреженность слишком низка, темы могут сливаться, а если слишком высока — содержать недостаточное количество слов для адекватного представления.

1.6.4 Чистота темы

Чистота темы:

$$\sum_{w \in W_t} p(w|t), \quad (23)$$

где W_t — ядро темы: $W_t = \{w : p(w|t) > \alpha\}$, где α подбирается по разному, напр

Данная характеристика показывает как вероятноотносится ядро темы к фоновым словам темы, следовательно, чем больше вероятность ядра, тем лучше.

1.6.5 Контрастность темы

Контрастность темы:

$$\frac{1}{|W_T|} \sum_{w \in W_t} p(t|w). \quad (24)$$

Данная характеристика показывает насколько часто слова из ядра темы встречаются в других темах, очевидно, что чем меньше ядро будет встречаться в других темах, тем лучше.

2 Тематическое моделирование новостей

В данном разделе будет выполнено тематическое моделирование новостей новостного сайта ВШЭ.

Датасет был получен методами парсинга с помощью языка python и библиотек BeautifulSoup4 и selenium.

2.1 Предобработка текстов

Перед любым моделированием данные нужно подготовить. Вот стандартный набор предобработки текстов для тематического моделирования:

- токенизация;
- перевод текста в нижний регистр;
- удаление неалфавитных символов;
- удаление стоп слов;
- лемматизация.

2.1.1 Токенизация, перевод в нижний регистр и удаление неалфавитных символов

Токенизация — это разделение текста на составные части — токены (предложения и слова).

Провести токенизацию можно с помощью средств языка python, библиотека nltk. За токенизацию отвечают команды:

```
# разделить текст на предложения
nltk.sent_tokenize(<sentences>)
# разделить предложение на слова
nltk.word_tokenize(<sentence>)
```

После того как текст поделен на слова, нужно перевести все слова в нижний регистр, так как семантическое значение слов, чаще всего, не зависит от регистра. Перевод в нижний регистр можно с помощью стандартных средств языка python:

```
# перевести текст в нижний регистр
<text>.lower()
```

После перевода в нижний регистр нужно удалить все семантически незначимые символы, в данном случае будем рассматривать в качестве таких симво-

лов все символы, не совпадающие с символами русского и английского алфавитов. Чтобы провести удаление неалфавитных символов достаточно средств языка python:

```
new_word = ''
# перебираем символы некоторого слова
for symbol in word:
    # если символ принадлежит русскому или английскому алфавитам
    if (symbol >= 'а' and symbol <= 'z' or symbol >= 'A' and symbol <= 'Я')
        # добавляем символ в новое слово
        new_word += symbol
```

2.1.2 Удаление стоп-слов

Стоп-слова — это слова, которые не несут смысловой нагрузки в рамках, некоторой темы.

Любой текст содержит большое количество слов общей тематики — стоп-слов. Такие слова, для улучшения качества модели, можно удалить, так как такие слова не несут семантической нагрузки, то будут только сбивать модель.

Чтобы удалить стоп-слова можно воспользоваться библиотеки nltk языка python:

```
new_words = []
# перебираем список слов
for word in words:
    # проверяем какому алфавиту принадлежат символы слова
    if re.match('[а-я]', word):
        # если слово не принадлежит списку стоп слов
        if word not in (stopwords.words('russian')):
            # добавляем слово в новый список слов
            new_words.append(word)
    elif re.match('[a-z]', word):
        # если слово не принадлежит списку стоп слов
        if word not in stopwords.words('english'):
            # добавляем слово в новый список слов
            new_words.append(word)
```

2.1.3 Лемматизация

Лемматизация — процесс приведения слова к его начальной форме.

Так как семантическое значение слова для темы не зависит от его формы и падежа, то перед обучением модели важно привести все слова в начальную форму, сделать это можно с помощью библиотек `nltk` и `pymorphy2` языка `python`:

```
# создаём лемматизаторы
lemm_nltk = WordNetLemmatizer()
lemm_pymorphy2 = pymorphy2.MorphAnalyzer()

new_words = []
# перебираем список слов
for word in words:
    # проверяем какому алфавиту принадлежат символы слова
    if re.match(' [а-я] ', word):
        # лемматизируем слово на русском и добавляем его в новый список слов
        new_words.append(lemm_pymorphy2.parse(word)[0].normal_form)
    elif re.match(' [a-z] ', word):
        # лемматизируем слово на английском и добавляем его в новый список сл
        new_words.append(lemm_nltk.lemmatize(word))
```

2.1.4 Создание N-грамм

N-грамма — это склеивание слов в словосочетание, слов может быть несколько.

Часто слова в теме встречаются в парах или тройках подряд, тогда, если склеить слова в N-грамм, то качество и интерпретируемость модели может вырасти.

Сделать N-граммы можно средствами библиотеки `nltk` языка `python`:

```
n_gramms = []
# перебираем предложения и составляем список n-грамм
for sentence in sentences:
    # делаем n граммы и добавляем их в список n-грамм
    n_gramms.append(sentence.split(' '), <n>)
```

2.2 Статистика по данным

Чтобы корректнее строить тематические модели нужно знать количественные характеристики данных, получить такие данные можно удобно с помощью библиотек `pandas` и `pumpru` языка `python`.

Перечислим некоторые количественные характеристики, характеризующие наш датасет (перед вычислениями проводилась предобработка данных, исключая лемматизацию):

- количество новостей в датасете: 15768;
- средняя длина документа (в словах): 34.6;
- медианная длина документа (в словах): 29;
- двадцать наиболее популярных слов датасета:
 1. вшэ: 11437;
 2. ниу: 5559;
 3. экономики: 4783;
 4. россии: 2955;
 5. высшей: 2498;
 6. школы: 2293;
 7. гувшэ: 2107;
 8. вышки: 2100;
 9. года: 2070;
 10. развития: 2065;
 11. исследований: 1876;
 12. образования: 1858;
 13. году: 1737;
 14. программы: 1644;
 15. студентов: 1481;
 16. факультета: 1428;
 17. университета: 1399;
 18. института: 1307;
 19. школа: 1303;
 20. рамках: 1286.

По этим данным можно сделать следующие выводы:

- общий объём данных весьма не велик, что может усложнить построение тематической модели;

- короткая медианная длина документов тоже приведёт к снижению качества модели, так как тематическое моделирование происходит на текстах большей длины;
- среди 20 наиболее популярных слов датасета явно присутствуют слова общей лексики (стоп-слова), которые необходимо будет удалить на этапе удаления стоп-слов.

Программу вычисляющую количественные характеристики датасета можно найти в приложениях.

2.2.1 Создание тематической модели с помощью библиотеки BigARTM

Блок тематических моделей уже реализован в библиотеке BigARTM, которую можно использовать на языке python.

Модели BigARTM для своей работы требуют особого типа данных — `vowpal_wabbit`. Данный тип данных представляет из себя следующую конструкцию.

Преобразовать excel таблицу с новостями к данному формату можно с помощью стандартных средств языка python и библиотеки pandas:

```
# считываем excel таблицу в pandas DataFrame
data = pd.read_excel('news.xlsx')
# открываем файл для записи vowpal_wabbit файла
f = open(<path>, 'w')
# проходимся по строкам DataFrame
for string in range(data.shape[0]):
    # записываем отдельную новость в файл как отдельный документ
    f.write('doc_{0}'.format(string)
          + data.loc[string, 'title']
          + ' '
          + data.loc[string, 'content']
          + '\n')
# после записи закрываем файл
f.close()
```

Чтобы передать данные из `vowpal_wabbit` файла на обучение необходимо создать батчи, они удобно будут постепенно загружаться в оперативную память

по мере необходимости и передаваться на обучение, кроме того батчи автоматически вычисляют для себя словарь, который также необходим при обучении. Создать батчи можно следующим образом:

```
# data_path - путь к vowpal_wabbit файлу
# data_format - формат загружаемого файла - vowpal_wabbit
# batch_size - количество документов в одном батче
# target_folder - папка, в которую батчи сохраняются
bv = artm.BatchVectorizer( data_path = 'vw.txt',
                           data_format = 'vowpal_wabbit',
                           batch_size=3000,
                           target_folder='batches' )
```

Наконец, можно создать саму модель, делается это следующим образом:

```
# num_topics - количество тем
# num_document_passes - количество проходов по каждому документу (новости)
# dictionary - словарь
# class_ids - веса для модальностей
# создание модели
model = artm.ARTM( num_topics=7,
                   num_document_passes=3,
                   dictionary=bv.dictionary,
                   class_ids={'@default_class': 1.0})
# добавление метрик
model.scores.add(artm.PerplexityScore(name='perplexity', dictionary=bv.dictionary))
# сохранения топа слов для каждой темы
model.scores.add(artm.TopTokensScore(name='top-tokens', num_tokens=10))
# добавление регуляризаторов, например, декоррелятора
# tau - коэффициент регуляризации
model.regularizers.add(artm.DecorrelatorPhiRegularizer(name='decorrelator', tau=
```

Метрик качества, а также регуляризаторов можно добавить сразу несколько.

После создания модели её нужно обучить, сделать это можно следующим образом:

```
for _ in range(<num_passes>):
    model.fit_offline(bv, num_collection_passes=1)
```

Чтобы оценить модель можно запросить значение метрик и список слов для тем:

```
# запрашиваем последнее значение перплексии
perplexity = model.score_tracker['perplexity'].last_value
# запрашиваем массив самых популярных слов для каждой темы
top_tokens = model.score_tracker['top-tokens'].last_value
```

2.3 PLSA (модель без регуляризаторов)

Модели PLSA соответствует EM-алгоритм без регуляризаторов. Данную модель можно создать средствами библиотеки BigARTM следующим образом:

```
model = artm.ARTM( num_topics=param1,
                  num_document_passes=param3,
                  dictionary=bv.dictionary,
                  class_ids={'@default_class': 1.0} )

model.scores.add(artm.PerplexityScore(name='perplexity', dictionary=bv.dictionary))
model.scores.add(artm.SparsityPhiScore(name='sparsity_phi_score'))
model.scores.add(artm.SparsityThetaScore(name='sparsity_theta_score'))
model.scores.add(artm.TopTokensScore(name='top-tokens', num_tokens=10))
```

Для оценки качества модели выбраны такие характеристика как перплексия и разреженность (по матрицам Φ и Θ).

На место параметров модели (param1, param3) в функции создания и обучения, которую можно увидеть в приложениях, будут подставляться значения из некоторого набора, затем модель будет обучаться param2 раз. После обучения будет выведена соответствующая таблица с результатами по моделям.

В результате получаем следующую таблицу:

2.4 LDA (модель с регуляризатором сглаживания)

Модели LDA соответствует EM-алгоритм с регуляризатором сглаживания. Создать модель можно следующим образом:

```

model = artm.ARTM( num_topics=param1,
                    num_document_passes=param3,
                    dictionary=bv.dictionary,
                    class_ids={ '@default_class': 1.0} )

model.scores.add(artm.PerplexityScore(name='perplexity', dictionary=bv.dictionary))
model.scores.add(artm.SparsityPhiScore(name='sparsity_phi_score'))
model.scores.add(artm.SparsityThetaScore(name='sparsity_theta_score'))
model.scores.add(artm.TopTokensScore(name='top-tokens', num_tokens=10))

```

Для оценки качества модели выбраны такие же характеристики как и у модели PLSA.

На место параметров модели (param1, param3) в функции создания и обучения, которую можно увидеть в приложениях, будут подставляться значения из некоторого набора, затем модель будет обучаться param2 раз. После обучения будет выведена соответствующая таблица с результатами по моделям.

В результате получаем следующую таблицу:

2.5 Модель с регуляризатором разреживания

В данном случае модели соответствует EM-алгоритм с регуляризатором разреживания. Создать модель можно следующим образом:

```

model = artm.ARTM( num_topics=param1,
                    num_document_passes=param3,
                    dictionary=bv.dictionary,
                    class_ids={ '@default_class': 1.0} )

model.scores.add(artm.PerplexityScore(name='perplexity', dictionary=bv.dictionary))
model.scores.add(artm.SparsityPhiScore(name='sparsity_phi_score'))
model.scores.add(artm.SparsityThetaScore(name='sparsity_theta_score'))
model.scores.add(artm.TopTokensScore(name='top-tokens', num_tokens=10))

```

Характеристики для оценки качества используются всё те же.

На место параметров модели (param1, param3) в функции создания и обучения, которую можно увидеть в приложениях, будут подставляться значения из некоторого набора, затем модель будет обучаться param2 раз. После обучения будет выведена соответствующая таблица с результатами по моделям.

В результате получаем следующую таблицу:

2.6 Модель с регуляризатором декоррелирования

В данном случае модели соответствует ЕМ-алгоритм с регуляризатором декоррелирования. Создать модель можно следующим образом:

```
model = artm.ARTM( num_topics=param1,  
                  num_document_passes=param3,  
                  dictionary=bv.dictionary,  
                  class_ids={ '@default_class': 1.0} )  
  
model.scores.add(artm.PerplexityScore(name= 'perplexity', dictionary=bv.dictionary))  
model.scores.add(artm.SparsityPhiScore(name= 'sparsity_phi_score'))  
model.scores.add(artm.SparsityThetaScore(name= 'sparsity_theta_score'))  
model.scores.add(artm.TopTokensScore(name= 'top-tokens', num_tokens=10))
```

Характеристики для оценки качества используются всё те же.

На место параметров модели (param1, param3) в функции создания и обучения, которую можно увидеть в приложениях, будут подставляться значения из некоторого набора, затем модель будет обучаться param2 раз. После обучения будет выведена соответствующая таблица с результатами по моделям.

В результате получаем следующую таблицу:

2.7 Выбор лучшей модели

В работе

ЗАКЛЮЧЕНИЕ

В работе

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Вероятностное тематическое моделирование: теория регуляризации ARTM и библиотека с открытым исходным кодом BigARTM [Электронный ресурс]. — URL: <http://www.machinelearning.ru/wiki/images/d/d5/Voron17survey-artm.pdf> (Дата обращения 26.10.2023). Загл. с экр. Яз. рус.
- 2 Вероятностные тематические модели Лекция 1. Постановка задачи, оптимизация и регуляризация [Электронный ресурс]. — URL: <http://www.machinelearning.ru/wiki/images/6/6a/Voron24ptm-intro.pdf> (Дата обращения 4.11.2023). Загл. с экр. Яз. рус.
- 3 Тематическое моделирование. Лекция 1. [Электронный ресурс]. — URL: <https://youtu.be/sBdFG8Rl-i8?si=pEmqLSU8yJU2M3DH> (Дата обращения 4.11.2023). Загл. с экр. Яз. рус.
- 4 *Николаевич, Ш.* Вероятность-1 / Ш. Николаевич. — Москва: МЦНМО, 2021.
- 5 Вероятностные тематические модели Лекция 2. Онлайновый EM-алгоритм и аддитивная регуляризация [Электронный ресурс]. — URL: <http://www.machinelearning.ru/wiki/images/b/be/Voron24ptm-regular.pdf> (Дата обращения 18.11.2024). Загл. с экр. Яз. рус.
- 6 Тематическое моделирование. Лекция 2. [Электронный ресурс]. — URL: <https://youtu.be/bA6wS6j6akU?si=HhiEqyPQOo-PbyfX> (Дата обращения 18.11.2024). Загл. с экр. Яз. рус.
- 7 Вероятностные тематические модели Лекция 4. Оценивание качества тематических моделей [Электронный ресурс]. — URL: <http://www.machinelearning.ru/wiki/images/0/0b/Voron24ptm-quality.pdf> (Дата обращения 01.01.2024). Загл. с экр. Яз. рус.
- 8 Тематическое моделирование. Лекция 3. [Электронный ресурс]. — URL: <https://youtu.be/FfxuULD-TmU?si=n6hMcjbQTvFjUQsf> (Дата обращения 01.01.2024). Загл. с экр. Яз. рус.