Федеральное государственное бюджетное образовательное учреждение
высшего образования «Московский государственный университет
имени М. В. Ломоносова»

На правах рукописи

Апишев Мурат Азаматович

Эффективная реализация алгоритмов тематического моделирования с аддитивной регуляризацией

05.13.17 – Теоретические основы информатики

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата технических наук

Научный руководитель
д. ф.-м. н., проф. РАН
Воронцов Константин Вячеславович

Москва – 2020

Оглавление

Введение

Актуальность и степень разработанности темы исследования. В современном анализе текстовых данных возникают различные задачи, немалая часть которых может успешно решаться методами *тематического моделирования* [1–5]. Имея в своей основе гибкий математический аппарат, эффективная реализация алгоритма обучения тематических моделей способна успешно решать задачи кластеризации и классификации текстов [6], использоваться для поиска информации [7], суммаризации текстов [8], анализа трендов и новостных потоков [9, 10], обработки мультиязычных данных [11–13].

За более чем два десятилетия тематические модели стали одним из стандартных инструментов текстовой аналитики. Обучаемые с помощью различных EM-подобных алгоритмов, они решают одну и ту же задачу разложения разреженной стохастической матрицы «слова-документы», построенной по текстовой коллекции, в произведение матриц «слова-темы» и «темы-документы». Под темой понимается вероятностное распределение на множестве уникальных слов, с неформальной же точки зрения тема — это набор слов, относящихся к одной предметной области.

Исторически первой была предложена модель вероятностного латентного семантического анализа (PLSA) [1, 2], однако на текущий момент основным инструментом является модель латентного размещения Дирихле (LDA) [3, 4], которая вводит априорные распределения Дирихле для распределений слов в темах и тем в документах. Со времени появления LDA опубликовано множество работ, предлагающих различные модификации этой модели для решения разнообразных задач [6, 8–13]. Однако общей проблемой такого рода модификаций является сложность их комбинирования и учёта новых дополнительных требований к модели [5].

Подход аддитивной регуляризации тематических моделей (ARTM) [5] предоставляет теоретический аппарат для построения тематических моделей, кото-

рые оптимизируют заданный перед началом обучения набор формализованных критериев. Единственным программным инструментом, позволяющим обучать модели ARTM с высокой скоростью в потоковом (онлайновом) режиме, является библиотека BigARTM [14, 15]. Данная научная работа прежде всего посвящена модернизации алгоритмов, лежащих в основе этой библиотеки, повышению её производительности и расширению возможностей по обучению моделей с разнообразными свойствами. Другим направлением исследований является решение задач анализа данных с помощью аддитивно регуляризованных тематических моделей специальных видов, обучаемых в BigARTM.

Цели и задачи диссертационной работы. Целью диссертационной работы является разработка библиотеки тематического моделирования для построения моделей больших текстовых коллекций с возможностью гибкой настройки процесса обучения. Для достижения этой цели в диссертации решается следующий набор задач.

1. Разработка более эффективной версии онлайнового EM-алгоритма для обучения моделей ARTM.

2. Оптимизация программных реализаций алгоритмов для работы с разреженными тематическими моделями.

3. Повышение скорости тематического моделирования и итогового качества моделей путём оптимизации алгоритма за счёт модификации решаемой оптимизационной задачи.

4. Разработка моделей для извлечения из текстовых коллекций информации по заданной тематике.

5. Реализация методов повышения качества работы с данными транзакционной природы в тематических моделях.

Научная новизна. Научная новизна данной работы заключается в разработке и оптимизации единственной на текущий момент высокопроизводитель-

ной библиотеки, реализующей базовый аппарат аддитивно регуляризованных тематических моделей и его ключевые обобщения. Новыми результатами являются также: предлагаемые стратегии использования оптимизированных шагов алгоритма; метод получения тематической модели для извлечения информации заданной тематики из текстов и связанные с ним исследования свойств регуляризаторов; реализация общего гиперграфового (транзакционного) варианта модели ARTM и связанные с ней экспериментальные результаты.

Теоретическая и практическая значимость. Предлагается библиотека для эффективного построения разнообразных тематических моделей, которая позволяет строить сложные модели на больших данных при решении задач разведочного поиска, кластеризации, категоризации, классификации и суммаризации текстов.

Предлагаемая модель для извлечения информации из текстов разрабатывалась для решения практической задачи анализа этно-социального дискурса в социальных сетях. Помимо социологических исследований, такого рода модели могут быть полезны в маркетинговых исследованиях, бренд-аналитике, анализе новостных трендов.

Разработанная реализация алгоритма обучения гиперграфовых моделей позволяет использовать тематические моделей в задачах, в которых данные имеют транзакционную структуру, строить модели частей слов, предложений, решать задачи классификации, формирования рекомендаций, анализа банковских транзакций и т.д.

Методология и методы исследования. В работе используются методы разработки высокопроизводительных вычислительных систем, аппараты теории вероятностей, методов оптимизации и машинного обучения. Разработка программного кода производится на языках C++ и Python в рамках библиотеки BigARTM. Эксперименты проводятся с использованием библиотек с открытым кодом BigARTM, Gensim и Vowpal Wabbit и удовлетворяют принципам воспроизводимости результатов.

Положения, выносимые на защиту:

1. Алгоритм параллельного асинхронного онлайнового обучения регуляризованных мультимодальных тематических моделей.
2. Алгоритм обучения регуляризованных мультимодальных тематических моделей с разреженным хранением параметров.
3. Модификация EM-алгоритма с ускоренным E-шагом без нормировки и стратегии её применения для оффлайнового и онлайновых алгоритмов.
4. Стратегия комбинирования регуляризаторов для выделения специфических тем по заданному словарю с приложением к анализу этно-релевантных тем в текстах социальной сети.
5. Реализация алгоритма обучения гиперграфовых тематических моделей транзакционных данных.

Степень достоверности и апробация результатов.

Достоверность полученных результатов обеспечивается большим объемом исследуемого материала, широкой теоретической базой, а также применением стандартных программных инструментов и общепринятых методик проведения эксперимента. Основные результаты этой работы докладывались автором на следующих конференциях:

1. 26-я международная конференция ассоциации FRUCT, Ярославль, 2020.
2. 5-я международная конференция по анализу изображений, социальных сетей и текстов (AIST), Екатеринбург, 2016.
3. 23-я международная научная конференция студентов, аспирантов и молодых учёных «Ломоносов-2016», Москва, 2016.

Публикации. Материалы диссертации опубликованы в 11 печатных работах, из них 7 статей индексируются в базе Scopus [14, 16–21], одна опубликована

в журнале, входящем в перечень ВАК [22]. Работы [23–25] являются тезисами докладов.

Личный вклад автора. Разработка и реализация первой версии алгоритма DetAsync производились совместно с Фреем А.И. [17], вклад автора был решающим. Устранение ряда её технических недоработок и первое сравнение по производительности с конкурентами произведены автором лично [20]. Вклад автора во все прочие основные положения, выносимые на защиту, также является решающим.

Структура и объем диссертации. Диссертация состоит из введения, двух обзорных глав, трех глав с результатами проведенного исследования, заключения и библиографии. Общий объем диссертации оставляет 124 страницы, из них 116 страниц текста, включая 8 рисунков и 20 таблиц. Библиография включает 77 наименования на 8 страницах.

Краткое содержание по главам. В главе 1 приводятся базовая теоретическая информация о задаче тематического моделирования. Рассматриваются различные существующие математические модели и методы их обучения. Описываются недостатки классических подходов. Вводится обобщающий аппарат аддитивной регуляризации тематических моделей, представляющий собой математическую основу описываемых далее исследований. Описывается набор базовых регуляризаторов.

Глава 2 содержит обзор представленных в научной литературе работ по повышению эффективности обучения байесовских тематических моделей. Систематизируются, обобщаются и излагаются основные используемые методы, анализируются их достоинства и недостатки. Приводится описание самих реализаций и их особенностей. Ряд выводов, полученных на основании этой главы, используется в главе 3.

В главе 3 рассматривается высокопроизводительная библиотека для обучения тематических моделей BigARTM, описывается эволюция лежащих в её основе алгоритмов обучения. Демонстрируются преимущества использования

асинхронных вычислений и алгоритмов, учитывающих разреженность параметров модели.

В главе 4 вводится теория обучения тематических моделей с произвольной функцией потерь. Одним из её следствий является возможность обучения моделей с использованием быстрого E-шага — модификации обычного E-шага EM-алгоритма, не требующей выполнения операции нормировки. Предлагаются различные стратегии комбинирования обычных и быстрых E-шагов. Демонстрируются эксперименты, в ходе которых для различных типов алгоритмов выявляются стратегии, лучшие с точки зрения скорости и качества моделирования.

Глава 5 посвящена алгоритмическим расширениям и приложениям аддитивно регуляризованных тематических моделей. Вводится теория мультимодальных тематических моделей. На её основе предлагается методика построения модели, решающей задачу поиска в коллекции текстов тематик специфической направленности, заданных набором релевантных ключевых слов. Приводятся описания и результаты многочисленных экспериментов, в которых показывается превосходство данной модели над более простыми конкурентами, а также исследуются методы её настройки и дальнейшего использования. Во второй части главы описывается теория гиперграфовых (транзакционных) тематических моделей. Приводятся детали реализации поддержки обучения таких моделей в библиотеке BigARTM, а также описание условий и результатов набора экспериментов на синтетических данных, демонстрирующих преимущества использования гиперграфовых моделей при обработки транзакционных текстовых данных.

Глава 1

Вероятностное тематическое моделирование

В данной главе рассматривается задача тематического моделирования, а также три вида популярных тематических моделей и способы их обучения. Одной из первых была опубликована модель вероятностного латентного семантического анализа (Probabilistic Latent Semantic Analysis, PLSA) [1, 2]. Модель латентного размещения Дирихле (Latent Dirichlet Allocation, LDA) [3] является усовершенствованием PLSA, основанным на байесовском обучении. Подход аддитивной регуляризации тематических моделей (Additive Regularization of Topic Models, ARTM) [5] появился позднее и является обобщением как описанных классических моделей, так и большого числа модификаций, созданных на основе LDA. Описываются формулы EM-алгоритма и EM-подобных алгоритмов для обучения моделей PLSA и ARTM методом максимального правдоподобия, а также модели LDA методами максимизации апостериорной вероятности, сэмплирования Гиббса и вариационного вывода. Теоретический материал, излагаемый в этой главе, используется в обзоре в главе 2 и экспериментах в главах 3 и 4. В главе 5 демонстрируются обобщения подхода ARTM для работы с многомодальными и транзакционными текстовыми данными.

1.1. Постановка задачи тематического моделирования

Тематическое моделирование (topic modeling) — одно из современных направлений машинного обучения и обработки естественного языка, активно развивающееся с конца 90-х годов [1, 3, 5]. *Вероятностная тематическая модель* (probabilistic topic model, BTM) коллекции текстовых документов описывает каждый документ дискретным распределением вероятностей тем, каждую тему — дискретным распределением вероятностей слов. Тематическое моделирование преследует несколько целей: выявить тематическую кластерную структу-

ру коллекции; построить тематические векторные представления документов; описать каждую тему словами или фразами естественного языка. В отличие от обычной «жёсткой» кластеризации тематическая модель распределяет каждый документ по многим кластерам-темам «мягко», с различными вероятностями.

ВТМ применяются при решении различных задач, связанных с обработкой текста: поиск актуальных трендов в потоках новостей и научных публикаций [9, 10], классификация документов [6], семантический информационный поиск [26] (включая многоязычный [27]). Существуют иерархические обобщения тематических моделей [28], модели, учитывающие связи между документами через авторов и ссылки, связи слов в предложении [29], работающие с внутренней структурой документов.

В обзоре [30] представлено большое количество разновидностей ВТМ, основанных на латентном размещении Дирихле. Проблемой этого подхода является существенная сложность теоретического вывода при наложении более чем 2-3 дополнительных требований на модель. Подход аддитивной регуляризации тематических моделей обходит эту проблему, допуская комбинирование в любых сочетаниях разнообразных свойств, таких как иерархичность, динамичность, мультиязычность, N-граммность, разреженность, робастность и т.д. В работе [20] представлен развёрнутый обзор такого рода свойств, описанных в терминах модели ARTM.

Опишем задачу формально. Пусть заданы три конечных множества: коллекция D текстовых документов, словарь W всех употребляемых в них термов, и множество тем T. В роли термов могут выступать исходные слова, леммати-зированные слова, словосочетания, термины — в зависимости от того, какие методы были использованы на стадии предварительной обработки текста. Последовательность термов всех документов описывается вектором наблюдаемых переменных $X = (d_i, w_i)_{i=1}^n$, где n — суммарная длина всех документов коллекции. С каждым i-м термом связана неизвестная тема t_i. Последовательность $Z = (t_i)_{i=1}^n$ называется вектором скрытых переменных. Таким образом, D рас-

сматривается как случайная и независимая выборка троек d_i, w_i, t_i.

Предположим выполнение двух гипотез:

1. *Гипотеза «мешка слов»*: порядок термов в документе является несущественным, что позволяет перейти к компактному представлению, где каждому документу d ставится в соответствие словарь его уникальных термов, в котором для каждого терма w подсчитывается значение n_{dw} частоты встречаемости этого терма в документе d.

2. *Гипотеза условной независимости*: появление в документе d термов, связанных с темой t, зависит только от t никак не зависит от d, т.е. $p(w \mid t, d) = p(w \mid t)$.

Вероятностная тематическая модель описывает распределение термов в документе $p(w \mid d)$ вероятностной смесью распределений $p(w \mid t) = \phi_{wt}$ термов в темах, причём каждая тема имеет свою условную вероятность $p(t \mid d) = \theta_{td}$ в документе d:

$$p(w \mid d) = \sum_{t \in T} p(w \mid t)\, p(t \mid d) = \sum_{t \in T} \phi_{wt}\theta_{td}. \tag{1.1}$$

Задача тематического моделирования состоит в том, чтобы по наблюдаемым данным X найти матрицы параметров модели $\Phi = (\phi_{wt})_{W \times T}$ и $\Theta = (\theta_{td})_{T \times D}$. В классических моделях число тем является гиперпараметром и фиксируется заранее.

1.2. Вероятностный латентный семантический анализ

Для оценивания параметров Φ и Θ тематической модели 1.1 по коллекции D максимизируем правдоподобие:

$$p(X \mid \Phi, \Theta) = \prod_{i=1}^{n} p(d_i, w_i) = \prod_{i=1}^{n} p(w_i \mid d_i) p(d_i) = \prod_{d \in D} \prod_{w \in d} p(w \mid d)^{n_{dw}} p(d)^{n_{dw}} \;\to\; \max_{\Phi, \Theta}.$$

Прологарифмируем предыдущее выражение, отбросим константные члены, вспомним о (1.1) и получим следующую задачу максимизации при ограничениях неотрицательности и нормированности столбцов матриц Φ и Θ:

$$\ln p(X \mid \Phi, \Theta) = \sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} \;\to\; \max_{\Phi, \Theta}; \tag{1.2}$$

$$\sum_{w \in W} \phi_{wt} = 1, \phi_{wt} \geq 0; \tag{1.3}$$

$$\sum_{t \in T} \theta_{td} = 1, \theta_{td} \geq 0. \tag{1.4}$$

На задаче максимизации правдоподобия модели данных (1.2)–(1.4) основывается *вероятностный латентный семантический анализ* (PLSA) [1].

В [31] показано, что решение данной оптимизационной задачи удовлетворяет следующей системе уравнений относительно искомых параметров модели ϕ_{wt}, θ_{td} и неизвестных вероятностей скрытых переменных $p_{tdw} = p(t \mid d, w)$:

$$p_{tdw} = \underset{t \in T}{\operatorname{norm}}\big(\phi_{wt} \theta_{td}\big); \tag{1.5}$$

$$\phi_{wt} = \underset{w \in W}{\operatorname{norm}}\big(n_{wt}\big); \qquad n_{wt} = \sum_{d \in D} n_{dw} p_{tdw}; \tag{1.6}$$

$$\theta_{td} = \underset{t \in T}{\operatorname{norm}}\big(n_{td}\big); \qquad n_{td} = \sum_{w \in d} n_{dw} p_{tdw}; \tag{1.7}$$

где переменная n_{td} оценивает число термов темы t в документе d; переменная n_{wt} оценивает, сколько раз терм w относился к теме t во всей коллекции; оператор norm преобразует произвольный заданный вектор $(x_i)_{i \in I}$ в вектор вероятностей $(p_i)_{i \in I}$ дискретного распределения путём обнуления отрицательных элементов и нормировки:

$$p_i = \underset{i \in I}{\operatorname{norm}}(x_i) = \frac{\max\{0, x_i\}}{\sum\limits_{j \in I} \max\{0, x_j\}}, \text{ для всех } i \in I.$$

Для решения системы уравнений применяется метод простых итераций: сначала выбираются начальные приближения параметров ϕ_{wt} и θ_{td}, по ним

вычисляются вспомогательные переменные p_{tdw} и следующее приближение параметров ϕ_{wt} и θ_{td}. Вычисления продолжаются в цикле до сходимости. Этот итерационный процесс называется *EM-алгоритмом* [32]. Вычисление условных распределений скрытых переменных (1.5) называется E-шагом (expectation), оценивание параметров модели (1.6) и (1.7) — M-шагом (maximization).

1.3. Аддитивная регуляризация тематических моделей

В PLSA решается задача низкорангового неотрицательного матричного разложения. Стоит отметить, что она является некорректно поставленной и имеет бесконечно много решений, поскольку для любых Φ и Θ, являющихся решением задачи, матрицы $\Phi' = \Phi S$ и $\Theta' = S^{-1}\Theta$ также будут решением для всех невырожденных матриц S, при которых матрицы Φ' и Θ' удовлетворяют условиям (1.3) и (1.4).

Чтобы доопределить постановку задачи и сделать решение устойчивым, можно ввести дополнительный критерий регуляризации $R(\Phi, \Theta) \to \max$ [33].

Аддитивная регуляризация тематических моделей (ARTM) [5] обобщает эту идею введением взвешенной суммы нескольких регуляризаторов:

$$\ln p(X \mid \Phi, \Theta) + R(\Phi, \Theta) \;\to\; \max_{\Phi,\Theta}, \qquad R(\Phi, \Theta) = \sum_k \tau_k R_k(\Phi, \Theta). \tag{1.8}$$

Как и в случае PLSA, задача (1.8), (1.3), (1.4) может быть решена с помощью EM-алгоритма [31]:

$$p_{tdw} = \underset{t \in T}{\operatorname{norm}}\big(\phi_{wt}\theta_{td}\big); \tag{1.9}$$

$$\phi_{wt} = \underset{w \in W}{\operatorname{norm}}\left(n_{wt} + \phi_{wt}\frac{\partial R}{\partial \phi_{wt}}\right); \qquad n_{wt} = \sum_{d \in D} n_{dw} p_{tdw}; \tag{1.10}$$

$$\theta_{td} = \underset{t \in T}{\operatorname{norm}}\left(n_{td} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}\right); \qquad n_{td} = \sum_{w \in d} n_{dw} p_{tdw}. \tag{1.11}$$

Несложно заметить, что модель PLSA соответствует тривиальному частному случаю $R(\Phi, \Theta) = 0$.

Взвешенная сумма регуляризаторов $R(\Phi, \Theta)$ позволяет наложить на модель ряд ограничений, которые могут привести к желаемым свойствам итоговых матриц параметров.

В работе [20] предлагается описание многочисленных регуляризаторов, рассмотрим несколько базовых примеров, которые важны для дальнейшего изложения.

Введём понятие *KL-дивергенции*, которое для пары дискретных распределений p и q без нулевых элементов на одинаковом носителе размера k вычисляется по формуле

$$KL(p||q) = \sum_{i=1}^{k} p_i \ln \frac{p_i}{q_i}.$$

и является несимметричной мерой близости.

Базовым примером регуляризатора является сглаживание тем. Потребуем, чтобы столбцы ϕ_t и θ_d были близки к заданным распределениям $\beta = (\beta_w)_{w \in W}$ и $\alpha = (\alpha_t)_{t \in T}$ в смысле KL-дивергенции:

$$\sum_{t \in T} \mathrm{KL}(\beta||\phi_t) \;\to\; \min_{\Phi}; \qquad \sum_{d \in D} \mathrm{KL}(\alpha||\theta_d) \;\to\; \min_{\Theta}.$$

Сложив два критерия с коэффициентами τ_1 и τ_2 и удалив из суммы константы получим итоговый регуляризатор:

$$R(\Phi, \Theta) = \tau_1 \sum_{t \in T} \sum_{w \in W} \beta_w \ln \phi_{wt} + \tau_2 \sum_{d \in D} \sum_{t \in T} \alpha_t \ln \theta_{td} \;\to\; \max_{\Phi, \Theta}. \tag{1.12}$$

Применение общих формул (1.10)–(1.11) приводит к следующим выражениям для M-шага:

$$\phi_{wt} = \underset{w \in W}{\mathrm{norm}}(n_{wt} + \tau_1 \beta_w), \qquad \theta_{td} = \underset{t \in T}{\mathrm{norm}}(n_{td} + \tau_2 \alpha_t).$$

Недостаток сглаживания заключается в его противоречии гипотезе разреженности параметров тематической модели, естественным образом проистекающей из реальных данных. В этой ситуации полезным может оказаться регуляризатор, приводящий к обратному, разреживающему эффекту. Рост разреженности распределения приводит к падению его энтропии, наибольшей же

энтропией обладает равномерное распределение. Таким образом, регуляризатор разреживания параметров может быть построен на максимизации KL-дивергенции между столбцами матриц Φ и Θ и равномерными распределениями β и α. В этом случае итоговый регуляризатор и формулы М-шага получаются такими же, как и при сглаживании, с точностью до изменения знака:

$$R(\Phi,\Theta) = -\tau_1 \sum_{t\in T}\sum_{w\in W} \beta_w \ln \phi_{wt} - \tau_2 \sum_{d\in D}\sum_{t\in T} \alpha_t \ln \theta_{td} \;\to\; \max_{\Phi,\Theta}; \tag{1.13}$$

$$\phi_{wt} = \underset{w\in W}{\operatorname{norm}}(n_{wt} - \tau_1\beta_w), \qquad \theta_{td} = \underset{t\in T}{\operatorname{norm}}(n_{td} - \tau_2\alpha_t).$$

На практике тематическая модель тем более полезна, чем более различными темы она выделяет. Одним из способов формализации понятия различности для пары тем является их ковариация:

$$\operatorname{cov}(\phi_t, \phi_s) = \sum_{w\in W} \phi_{wt}\phi_{ws}.$$

Тогда можно определить регуляризатор *декорреляции* тем с коэффициентом τ_3, задачей которого является получение модели, содержащей как можно более различные темы:

$$R(\Phi,\Theta) = -\frac{\tau_3}{2}\sum_{t\in T}\sum_{s\in T\setminus t} \operatorname{cov}(\phi_t, \phi_s) \;\to\; \max_{\Phi,\Theta}; \tag{1.14}$$

$$\phi_{wt} = \underset{w\in W}{\operatorname{norm}}\left(n_{wt} - \tau_3\phi_{wt}\sum_{s\in T\setminus t}\phi_{ws}\right).$$

Очевидно, что этот регуляризатор является разреживающим и никак не зависит от матрицы Θ.

Исходно идея декорреляции предложена в байесовской модели Topic Weak Correlated LDA (TWC-LDA) [34], а затем адаптирована в виде не-байесовского регуляризатора в [31].

1.4. Латентное размещение Дирихле

В байесовском подходе для задания ограничений на параметры модели вводится априорное распределение $p(\Phi, \Theta \,|\, \gamma)$ с вектором гиперпараметров γ. Принцип *максимума апостериорной вероятности* (maximum a posteriori probability, MAP) эквивалентен введению регуляризатора, равного логарифму априорного распределения:

$$\ln p(X \,|\, \Phi, \Theta) + \ln p(\Phi, \Theta \,|\, \gamma) \;\to\; \max_{\Phi,\Theta}. \tag{1.15}$$

Таким образом, вероятностные предположения о параметрах модели, задаваемые через априорное распределение, можно переводить в вероятностный регуляризатор, и применять для решения задачи всё тот же описанный ранее EM-алгоритм (1.9)–(1.11).

Модель *латентного размещения Дирихле* (LDA) [3] является наиболее известной в тематическом моделировании. Она основана на априорном предположении, что столбцы матрицы Φ порождаются $|W|$-мерным распределением Дирихле $\mathrm{Dir}(\phi_t|\beta)$ с вектором гиперпараметров $\beta = (\beta_w)_{w\in W}$, а столбцы матрицы Θ — $|T|$-мерным распределением Дирихле $\mathrm{Dir}(\theta_d|\alpha)$ с вектором гиперпараметров $\alpha = (\alpha_t)_{t\in T}$. Таким образом, в модели LDA $\gamma = (\beta, \alpha)$.

Общая система уравнений ARTM (1.9)–(1.11) после подстановки в неё вероятностного регуляризатора модели LDA принимает вид

$$p_{tdw} = \underset{t\in T}{\mathrm{norm}}\big(\phi_{wt}\theta_{td}\big); \tag{1.16}$$

$$\phi_{wt} = \underset{w\in W}{\mathrm{norm}}\big(n_{wt} + \beta_w - 1\big); \qquad n_{wt} = \sum_{d\in D} n_{dw} p_{tdw}; \tag{1.17}$$

$$\theta_{td} = \underset{t\in T}{\mathrm{norm}}\big(n_{td} + \alpha_t - 1\big); \qquad n_{td} = \sum_{w\in d} n_{dw} p_{tdw}. \tag{1.18}$$

В тематическом моделировании наибольшее распространение получили методы байесовского обучения. Чтобы оценить параметры Φ, Θ, выводят их апостериорные распределения $p(\Phi, \Theta \,|\, X, \gamma)$, затем берут по ним оценки математического ожидания. Это более трудный путь, по сравнению с ARTM, но он

возник на десять лет раньше.

Среди методов байесовского обучения в тематическом моделировании наиболее популярны вариационный байесовский вывод и сэмплирование Гиббса.

Вариационный байесовский вывод (Variational Bayes, VB) основан на вычислении совместного апостериорного распределения параметров модели и скрытых переменных $p(\Phi, \Theta, Z \mid X, \gamma)$. Точное выражение для него получить не удаётся, поэтому ищется его приближённое представление в виде произведения независимых распределений по переменным t_i, ϕ_t, θ_d. Для модели LDA вариационный байесовский вывод приводит к системе уравнений, похожей на EM-алгоритм [3, 35]:

$$p_{tdw} = \underset{t \in T}{\operatorname{norm}} \left(\frac{E\big(n_{wt} + \beta_w\big)}{E\big(\sum_w (n_{wt} + \beta_w)\big)} \cdot \frac{E\big(n_{td} + \alpha_t\big)}{E\big(\sum_t (n_{td} + \alpha_t)\big)} \right); \tag{1.19}$$

$$\phi_{wt} = \underset{w \in W}{\operatorname{norm}}\big(n_{wt} + \beta_w\big); \qquad n_{wt} = \sum_{d \in D} n_{dw} p_{tdw}; \tag{1.20}$$

$$\theta_{td} = \underset{t \in T}{\operatorname{norm}}\big(n_{td} + \alpha_t\big); \qquad n_{td} = \sum_{w \in d} n_{dw} p_{tdw}; \tag{1.21}$$

где $E(x) = \exp(\psi(x)) \approx x - \frac{1}{2}$ — экспонента от дигамма-функции $\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$.

Сэмплирование Гиббса (Gibbs Sampling, GS) — это байесовский вывод апостериорного распределения скрытых переменных

$$p(Z \mid X, \gamma) = \int_\Phi \int_\Theta p(\Phi, \Theta, Z \mid X, \gamma)\, d\Phi\, d\Theta,$$

из которого сэмплируются значения $Z \sim p(Z \mid X, \gamma)$. Для этих Z вычисляется апостериорное распределение параметров модели $p(\Phi, \Theta \mid X, Z, \gamma)$, и по нему находятся оценки математического ожидания параметров Φ, Θ. Для модели LDA сэмплирование Гиббса приводит к системе уравнений, снова похожей на

EM-алгоритм [4]:

$$t_i \sim p_{td_iw_i} = \underset{t\in T}{\operatorname{norm}} \left(\frac{n_{w_it} + \beta_{w_i} - 1}{\sum_w (n_{w_t} + \beta_w) - 1} \cdot \frac{n_{td_i} + \alpha_t - 1}{\sum_t (n_{td_i} + \alpha_t) - 1} \right); \tag{1.22}$$

$$\phi_{wt} = \underset{w\in W}{\operatorname{norm}} \big(n_{wt} + \beta_w\big); \qquad n_{wt} = \sum_{i=1}^{n} [w_i = w][t_i = t]; \tag{1.23}$$

$$\theta_{td} = \underset{t\in T}{\operatorname{norm}} \big(n_{td} + \alpha_t\big); \qquad n_{td} = \sum_{i=1}^{n} [d_i = d][t_i = t]. \tag{1.24}$$

Главное отличие этого алгоритма от предыдущих в том, что для каждого терма (d_i, w_i) на каждой итерации сэмплируется единственная тема t_i, которая и участвует в аккумулировании счётчиков n_{wt} и n_{td}. Фактически, на M-шаге суммируются не сами распределения $p(t \mid d_i, w_i)$, а их вырожденные эмпирические оценки $[t = t_i]$, сделанные каждый раз по единственной сэмплированной теме t_i. Сумма таких оценок сходится к сумме исходных распределений, согласно закону больших чисел.

В работе [36] сэмплирование рассматривалось как отдельная эвристика, которую можно использовать в любом EM-подобном алгоритме тематического моделирования, начиная с PLSA. Эксперименты показали, что сэмплирование практически не влияет на сходимость и качество модели. В ARTM оно может свободно сочетаться с любыми регуляризаторами.

Глава 2

Эффективное обучение тематических моделей

В данной главе, основанной на работе [22], описываются различные техники повышения производительности алгоритмов обучения тематических моделей, производится их сравнительный анализ. Рассматриваются такие аспекты, как параллелизм модели и данных, асинхронность, онлайновость обучения, потребление оперативной памяти, использование свойств разреженности модели и отказоустойчивость. Помимо набора методов, глава содержит обзор ряда публикаций о реализациях алгоритмов, повышающих эффективность обучения тематических моделей. Материал этой главы носит обзорный характер и используется в главах 3 и 4.

2.1. Техники эффективного обучения тематических моделей

Описанные в главе 1 EM-подобные алгоритмы (1.9)–(1.11), (1.16)–(1.18), (1.19)–(1.21), (1.22)–(1.24) отличаются небольшими поправками к частотным оценкам условных вероятностей. При $n_{wt} \gg 1$, $n_{td} \gg 1$ эти поправки пренебрежимо малы. Они влияют лишь на близкие к нулю условные вероятности ϕ_{wt} и θ_{td}, которые не являются значимыми для тематической модели. Сходство EM-подобных алгоритмов PLSA, MAP, VB, GS для модели LDA и ещё нескольких их вариантов было замечено в [37].

Во всех рассмотренных в 1 алгоритмах вычисление переменных-счётчиков n_{wt} и n_{td} на M-шаге требует однократного прохода коллекции в цикле по всем термам $w \in d$ всех документов $d \in D$. Внутри этого цикла значение p_{tdw} вычисляется только один раз при обработке терма w в документе d, после чего его можно забыть. Это позволяет выделить итерационный процесс обработки

одного документа при фиксированной матрице Φ. В этом процессе E-шаг чередуется с M-шагом для одного столбца матрицы Θ, соответствующего данному документу. Счётчики n_{wt} и n_{td} собираются в процессе выполнения E-шага. По завершении обработки всех документов матрица Φ обновляется (если это требуется алгоритму). Такая организация вычислений не требует дополнительных вычислительных затрат и обходится без хранения трёхмерной матрицы p_{tdw}.

Практическая эффективность EM-подобных алгоритмов тематического моделирования определяется не столько математическими различиями, сколько особенностями программной реализации. Не столь важно, используется ли байесовский вывод или аддитивная регуляризация, используется ли сэмплирование или нет. Важно, в каком порядке организованы вычисления по формулам E- и M-шагов, как хранятся исходные данные и параметры модели, и как используются механизмы параллельных вычислений. Модификации такого рода, о которых пойдёт речь ниже, как правило подходят для любых описанных алгоритмов, более того, они полезны не только для тематического моделирования, но и для других задач матричного разложения больших разреженных матриц.

Распределённые хранение и обработка коллекции. В современных приложениях анализа текстов объём коллекции может быть настолько большим, что либо её не удастся разместить во внешней памяти одного вычислительного узла, либо время её обработки на одном узле окажется неприемлемым.

Возможным решением данной проблемы является использование вычислительного кластера [38–48].

Документы вместе с соответствующими им счётчиками n_{td} распределяются равномерно по ядрам узлов кластера и обрабатываются параллельно. Распределённо храниться могут также параметры, зависящие от документов, например θ_{td} или сэмплированные скрытые темы Z. Увеличение числа машин и ядер может обеспечивать рост производительности до определённого момента. Однако обработка сверхбольших или динамически пополняемых коллекций

может потребовать дополнительной оптимизации.

Параллельное выполнение сэмплирования или E-шага для ускорения обработки документов может использоваться и в рамках одного вычислительного узла, если он имеет достаточный объём оперативной памяти. Такой подход используется, например, в [42, 43, 46, 47].

Синхронная параллельная обработка. Параллельная обработка документов, вне зависимости от того, выполняется она на кластере или на одной машине, может быть организована различными способами. Отличаются они, главным образом, методами накопления счётчиков n_{wt} и обновления параметров ϕ_{wt}. Проблема в том, что для любого параллельного алгоритма эти величины являются разделяемыми ресурсами, к которым все рабочие процессы должны иметь доступ на этапе обработки документов.

Один из способов организации параллельных вычислений, представленный в работах [38, 39, 43–46, 48], можно условно назвать синхронным.

В нём текущая версия матрицы n_{wt} или нужная её часть копируется в начале итерации обработки коллекции в память каждого рабочего процесса и доступна ему на чтение, а получаемые в результате обработки приращения счётчиков n_{wt} аккумулируются локально. После того, как все параллельные процессы завершают работу над своими порциями документов, они складывают приращения в глобальную матрицу n_{wt}, которая копируется и используется для обработки текстов во время следующей итерации.

Синхронный подход к аккумулированию счётчиков прост в реализации, но плох тем, что нагрузка на вычислительные и сетевые ресурсы распределена неравномерно: попеременно то сеть, то процессоры или простаивают, или перегружены. Кроме того, скорость параллельной обработки во время одной итерации определяется самым медленным из обработчиков, что может приводить к существенной деградации производительности.

Асинхронная параллельная обработка. В отличие от синхронной параллельной обработки документов, асинхронный вариант [40–42, 47] не имеет выделенного шага синхронизации и обновляет счётчики n_{wt} (а при необходимости и матрицу Φ) одновременно с обработкой документов. Архитектуры на его основе сложнее в реализации и настройке. Кроме того, асинхронность часто увеличивает потребление оперативной памяти. Однако производительность и масштабируемость при использовании асинхронного подхода обычно выше, чем при синхронном.

Способы реализации асинхронных архитектур разнообразны. В [40] случайным образом выбранные пары вычислительных узлов обмениваются своими приращениями n_{wt} после завершения обработки документов (асинхронность заключается в том, что все прочие узлы могут продолжать работать независимо от других). В реализации [42] обновления счётчиков передаются в глобальное хранилище модели в фоновом режиме, без остановки процесса обработки документов. Иной способ реализации асинхронной архитектуры предлагается в главе 3 настоящей работы.

Внешнее хранение параметров документов. Очевидным узким местом с точки зрения используемой памяти является хранение счётчиков документов n_{td}, значений Z или параметров Θ в памяти отдельного компьютера или кластера. В реализации Light LDA [47] предлагается хранить все связанные с документами счётчики и параметры на диске, подгружая их в память по мере необходимости. В главе 3 описывается ещё один способ обхода этой проблемы, устраняющий необходимость в хранении значительной части связанных с документами величин. Схожим образом производится обработка данных в онлайновых алгоритмах, которые рассматриваются ниже.

Онлайновая (потоковая) обработка. Оффлайновые алгоритмы делают на каждой итерации полный проход коллекции, накапливая счётчики n_{wt}, после

чего обновляют параметры ϕ_{wt}. Такие алгоритмы удобны, когда модель обучается на небольшой статической коллекция. В случае большой или динамически пополняемой коллекции алгоритму может потребоваться слишком много проходов для сходимости матрицы Φ.

Для обхода этих ограничения используются *онлайновые* алгоритмы, которые обновляют матрицу Φ после обработки каждого документа [49] или (в пакетном варианте) после каждого пакета (batch) документов [43, 50, 51]. Это ускоряет сходимость итерационного процесса. На большой коллекции матрица Φ может сойтись и перестать меняться задолго до окончания первой итерации. В таких случаях одного прохода по коллекции может оказаться достаточно для построения модели. Поэтому онлайновые алгоритмы предпочтительны для обработки больших или потоковых данных. При отказе от хранения параметров θ_{td} и счётчиков n_{td} онлайновый алгоритм позволяет тематизировать потенциально бесконечное число документов при фиксированном потребляемом объёме оперативной памяти.

Распределённое или оптимизированное хранение модели. При обработке больших текстовых коллекций в моделях с большим числом тем размер матрицы счётчиков n_{wt} и матрицы параметров Φ может превысить объём оперативной памяти, доступный одному компьютеру. Эту проблему можно решать двумя способами: либо хранением значительной части модели во внешней памяти с подгрузкой нужных её фрагментов в оперативную память [49], либо распределённым хранением модели в памяти машин в кластере [41, 45–48].

Первый вариант позволяет строить большие модели на одной машине, но представляет меньший интерес, поскольку операции с внешней памятью существенно медленнее, чем с оперативной.

Во втором случае предполагается, что суммарный объём оперативной памяти на машинах кластера достаточно велик для хранения всей модели, но в память одной машины вся модель не поместится. В этой ситуации и кол-

лекция, и модель разбиваются некоторым образом на части, которые хранятся и обрабатываются на различных вычислительных узлах (конкретный способ разбиения и взаимодействия частей зависит от реализации). Это обеспечивает и параллельную обработку коллекции, и размещение большой модели в памяти кластера без увеличения объёма оперативной памяти на отдельных узлах.

Разреженное хранение модели. Ещё одним способом работы с большой моделью является её представление в разреженном виде. В ходе итераций матрицы n_{wt} и Φ могут становиться всё более разреженными [31, 36], что открывает возможность для их более экономного хранения. Для этого могут использоваться различные форматы хранения разреженных данных, например CSR (Compressed Sparse Row) или хэш-таблицы.

Обратной стороной разреженного хранения может стать снижение производительности, вызываемое заменой прямого доступа к элементам матриц на последовательный. По этой причине в [47] и [48] используется гибридный подход: параметры и счётчики, связанные с наиболее часто встречающимися термами, хранятся в плотном виде для ускорения вычислений, а оставшиеся термы хранятся разреженно в виде хэш-таблицы для экономии памяти.

Разреженная инициализация модели. Как было отмечено выше, разреженные матрицы n_{wt} и Φ позволяют ощутимо снизить потребление оперативной памяти. Чтобы этот подход давал наибольшую экономию, матрицы должны быть разреженными на протяжении всего итерационного процесса, а не только начиная с некоторого момента. Стандартная процедура инициализации параметров случайными числами и сэмплирование тем на стартовой итерации дают в результате плотную матрицу, что нивелирует все усилия по снижению пикового потребления памяти (по крайней мере, для оффлайновых алгоритмов).

Одним из возможных решений этой проблемы является разреженная инициализация. В зависимости от алгоритма обучения она может заключаться в об-

нулении части значений в матрице Φ при случайной генерации значений параметров или в сужении допустимого множества присваиваемых термам тем при сэмплировании.

Несмотря на кажущуюся грубость такого решения, эксперименты в [47, 48] показывают, что использование разреженной инициализации не сильно ухудшает качество тематической модели, существенно снижая при этом потребление оперативной памяти.

Динамическое изменение размера модели. Другой способ экономии памяти при построении разреженных моделей заключается в постепенном добавлении новых тем по мере увеличения числа документов в коллекции. В этом случае сначала строится предварительная модель с относительно небольшим числом тем по имеющейся части коллекции. Затем в модель добавляются новые темы по мере поступления новых документов. Возможность изменения числа тем в матрицах Φ и Θ доступна в библиотеке BigARTM [15], речь о которой пойдёт в следующей главе.

Разреженная обработка термов. Термы могут иметь существенно различную частоту в коллекции. Термы, которые часто встречаются в документах, обрабатываются чаще, поэтому связанные с ними параметры ϕ_{wt} сходятся быстрее. с определённого момента приращения счётчиков n_{wt} высокочастотных термов перестают добавлять в модель новую информацию, продолжая потреблять значительные ресурсы на своё вычисление.

Для решения данной проблемы в [48] и [49] предлагается хранить для каждого терма предшествующие результаты сэмплирования или E-шага и оценивать, насколько сильно они изменились. Термы, для которых изменения систематически оказываются незначительными, исключаются из дальнейшей обработки либо безусловно, либо с некоторой вероятностью.

Обеспечение отказоустойчивости. Обеспечение отказоустойчивости важно для любого длительного процесса, выполняемого в сложной вычислительной среде, где возможны сбои в сети, аппаратном или программном обеспечении. Обучение тематических моделей в этом случае не является исключением.

Сложность реализации отказоустойчивого алгоритма на кластере во многом зависит от базовой программной платформы, поверх которой строится реализация. Так, промышленные системы Hadoop [52] и Spark [53] обеспечивают высокую устойчивость кластера к сбоям в процессе выполнения вычислительной задачи, в то время как MPI [54] такой возможности не предоставляет, и обеспечение отказоустойчивости целиком возлагается на разработчиков.

Основным методом повышения отказоустойчивости EM-подобных алгоритмов является периодическое сохранение на диск текущего состояния модели и счётчиков. Такой подход позволяет восстановить состояние системы и продолжить обучение с места остановки при сбоях, не затрагивающих диск. Он может использоваться в реализациях алгоритма как на кластере, так и на отдельной машине.

2.2. Обзор реализаций алгоритмов обучения

В этом разделе рассматриваются (в основном в хронологическом порядке) реализации алгоритмов тематического моделирования, и описываются детали использования технических приёмов, описанных выше.

AD-LDA. AD-LDA [38] — одна из первых параллельных реализаций обучения модели LDA. За основу взят алгоритм сэмплирования Гиббса, параллелизм реализован на кластере на уровне ядер с помощью технологии MPI. Каждое ядро обрабатывает свою порцию документов и получает локальную копию счётчиков n_{wt} перед началом очередной итерации. Используется синхронная архитектура, то есть после того, как обработка документов завершается на всех ядрах, запус-

кается процедура добавления всех полученных приращений счётчиков в общую глобальную матрицу счётчиков. Отказоустойчивость и дополнительные оптимизации в AD-LDA отсутствуют.

PLDA. PLDA [39] является ещё одной реализацией идей AD-LDA с помощью технологии MPI. Никаких существенных алгоритмических или архитектурных улучшений алгоритма не предлагается. В отличие от своего предшественника, PLDA поддерживает отказоустойчивость между блоками итераций сэмплирования путём сохранения промежуточных данных на диск.

AS-LDA. AS-LDA [40] — одна из первых асинхронных многопроцессорных архитектур для параллельного выполнения алгоритма сэмплирования Гиббса. Как и в AD-LDA, все процессоры получают при старте свою порцию документов и вычисляют приращения счётчиков. Синхронизация организована следующим образом: каждый процессор по завершении очередного этапа обработки обменивается результатом (коммутирует) со случайным процессором, также завершившим свой этап.

Содержательной частью реализации является процедура агрегации данных при таком обмене. В ситуации первой коммутации процессоры просто прибавляют полученные счётчики к собственным. При повторном обмене аппроксимированные значения счётчиков n_{wt}, полученных в предыдущей коммутации, вычитаются из текущих значений счётчиков процессоров, после чего результат складывается с новыми значениями от текущей коммутации. Тривиальное решение в виде хранения предыдущих счётчиков авторы посчитали неприемлемым из-за существенных затрат оперативной памяти.

Алгоритм реализован на кластере с помощью MPI без дополнительных усовершенствований.

PLDA+. PLDA+ [41] — асинхронная кластерная реализация на основе сэмплирования Гиббса. Ключевой её особенностью является разделение процессоров на две группы: рабочие, которые обрабатывают документы и выполняют сэмплирование тем, и транспортные, отвечающие за обновление и доставку глобальных параметров. Такой подход позволяет производить обмен данными в фоновом режиме, без прерывания обработки документов.

Жизненный цикл группы транспортных процессоров состоит из этапа размещения матрицы n_{wt} и этапа обработки запросов. Каждый процессор получает свою часть счётчиков и далее занимается их обновлением и доставкой рабочим процессорам.

Перед началом работы документы коллекции распределяются между рабочими процессорами случайным образом. Необычным решением является одновременная обработка вхождений одного и того же терма во всех документах данного процессора. Для удобства реализации этой идеи термы из словаря процессора группируются в блоки для выполнения итераций сэмплирования Гиббса и отправки запросов. На данном этапе редкие термы сливаются с частыми для сокращения времени сэмплирования. Для минимизации вероятности одновременной обработки одного терма двумя рабочими процессорами создаётся циклическая очередь обрабатываемых термов и расписание на ней. Это позволяет избежать коллизий при отправке запросов к транспортным процессорам.

Y!LDA. Y!LDA [42] — это кластерная версия алгоритма сэмплирования Гиббса, в которой параллелизм реализован на уровне вычислительных узлов. На каждом узле запускается многопоточный процесс, занимающийся сэмплированием тем для термов документов, размещённых на узле перед стартом обучения. Общая для всех потоков память позволяет хранить в пределах одного узла всего две локальные матрицы n_{wt} вместо $O(N)$, как в AD-LDA и PLDA, где N — число ядер в процессорах узла. При этом на каждом узле имеется выделенный транспортный поток, который асинхронно получает от рабочих потоков

приращения счётчиков и обновляет локальную матрицу n_{wt}.

Глобальная матрица счётчиков размещается в распределённом хранилище memcached [55]. Каждый вычислительный узел работает с ней (отправляет свои приращения и обновляет локальные значения), блокируя по одному терму за раз вне зависимости от действий других узлов, что обеспечивает асинхронность и на кластерном уровне.

Между итерациями сэмплирования потоки в узлах сохраняют текущие значения тем Z для термов своих документов на диск, что обеспечивает возможность рестарта обработки с последней итерации в случае сбоя.

Vowpal Wabbit LDA. Vowpal Wabbit [51] — это библиотека онлайновых алгоритмов машинного обучения, которая включает в себя реализацию онлайнового вариационного EM-алгоритма для модели LDA, предложенного в [50]. Данная реализация алгоритма является пакетной, однопоточной и рассчитана на использование в рамках одного компьютера.

Gensim. Gensim [43] — это набор библиотек для построения тематических моделей и решения других задач текстовой аналитики. В нём реализованы две версии онлайнового вариационного EM-алгоритма [50]: однопоточная и многопоточная. Однопоточный вариант представляет собой обычный пакетный онлайновый алгоритм. В многопоточном используется модель параллелизма, схожая с Y!LDA на одном вычислительном узле: коллекция разбивается на пакеты, обрабатываемые в несколько потоков, выделенный поток асинхронно собирает счётчики, полученные рабочими потоками, и обновляет матрицу n_{wt} этого узла.

В библиотеке также заявлена поддержка обучения моделей на кластере из нескольких вычислительных узлов на базе описанных выше алгоритмов.

FOEM-LDA. FOEM-LDA [49] предоставляет собой однопоточную реализацию онлайнового EM-алгоритма для критерия MAP в модели LDA (по фор-

мулам (1.16)–(1.18)) для работы на одном вычислительном узле. Алгоритм не является пакетным (обновление параметров производится после каждого обработанного документа) и не хранит матрицу Θ целиком.

В данной реализации решается проблема хранения модели, не помещающейся в оперативной памяти. Для этого матрицы Φ и n_{wt} разделяются на две части, соответствующие более и менее важным термам. Первые хранятся в оперативной памяти постоянно, вторые подгружаются по мере необходимости. Важность термов на первой итерации оценивается по их частоте, далее — по скорости сходимости, рассчитываемой на основе текущих и предшествующих значений вспомогательных переменных p_{tdw}.

Для минимизации количества обновлений n_{wt} на текущей итерации (и, соответственно, числа загрузок данных с диска) расчёт скорости сходимости производится сразу после обработки пакета документов. Приращения счётчиков для термов, у которых значения p_{tdw} изменились слабо, игнорируются.

Mr.LDA. Mr.LDA [44] представляет собой реализацию байесовского вариационного EM-алгоритма в рамках парадигмы MapReduce [56] на базе платформы Hadoop.

Каждой итерации алгоритма соответствует запуск процедуры MapReduce. На этапе Map для каждого документа коллекции независимо выполняется E-шаг (1.19). Полученные результаты агрегируются по ключам-номерам тем и отправляются на этап Reduce, где для каждой темы выполняется M-шаг (1.20)–(1.21). После этого производится обновление параметров модели, которые хранятся в распределённом кэше Hadoop-кластера — специальной, доступной только для чтения и общей для всех компонентов системы памяти.

Spark-LDA. Spark-LDA [45] — одна из первых библиотек для обучения модели LDA с помощью сэмплирования Гиббса в рамках платформы Spark. И данные, и параметры модели в Spark-LDA распределяются между вычислительны-

ми узлами таким образом, чтобы множества документов для разных машин не пересекались, а множества термов не пересекались для частей данных, обрабатываемых узлами в текущий момент времени. Для этого матрицы исходных данных n_{dw} и параметров n_{wt} нарезаются по термам на P частей. После на столько же частей производится нарезка данных и счётчиков n_{td} по документам, в результате получается $P \times P$ блоков данных. Из этих блоков набирается P наборов по P блоков, после чего для каждого набора выполняется параллельное сэмплирование и обновление локальных счётчиков n_{wt} и n_{td}. Документные счётчики локальны для данного вычислительного узла, поэтому их не нужно передавать по сети, счётчики же n_{wt} перемещаются между машинами при обработке очередного блока данных.

Peacock. Peacock [46] предназначена для обучения модели LDA на кластере с помощью сэмплирования Гиббса с возможностью распределённого хранения как данных, так и модели.

В Peacock все ядра разделяются на три группы: серверы сэмплирования, серверы данных и серверы синхронизации. Матрица входных данных n_{dw} и счётчики n_{wt} разбиваются на M блоков: первая матрица разбивается по документам, каждый блок связывается с некоторым сервером данных; вторая — по термам, каждый блок связывается с некоторым сервером сэмплирования.

Каждый сервер данных посылает каждому серверу сэмплирования те части своих документов, в которых содержатся термы, связанные с этим сервером сэмплирования. После обработки очередного блока сервер сэмплирования обновляет свои счётчики n_{wt} и отправляет приращения счётчиков на соответствующий сервер данных.

В обычной для текстовых коллекций ситуации $\|D\| \gg \|W\|$ разделение словаря и коллекции на множество одинаковых частей может привести к перегрузке серверов данных. Для решения этой проблемы коллекция предварительно делится на C частей, для каждой из которых производится своё разби-

ение на блоки и, соответственно, серверы. Для синхронизации счётчиков между различными разбиениями используются серверы синхронизации. Системе требуется M таких серверов (по числу блоков в разбиении), каждый работает с соответствующим блоком во всех разбиениях. В конце каждого набора итераций сэмплирования Гиббса все серверы сэмплирования отправляют обновления счётчиков n_{wt} соответствующим серверам агрегирования. После сбора всех обновлений каждый сервер агрегирования пересылает соответствующим серверам сэмплирования новую версию части матрицы n_{wt}, за которую они ответственны.

Light LDA. Light LDA [47] производит распределённое обучение модели LDA. Вместо сэмплирования Гиббса используется более общий алгоритм Метрополиса-Гастингса. Как данные, так и модель распределяются между узлами кластера. При этом данные хранятся статично, а фрагменты модели пересылаются между обработчиками по мере необходимости, как в PLDA+ и Spark-LDA.

Текстовая коллекция нарезается на блоки данных, каждый из которых является единицей обработки вычислительного узла. При загрузке блока данных в оперативную память узла происходит отбор небольшого количества термов из словаря этого блока. Выбранное множество достаточно мало, чтобы соответствующее его элементам множество строк матрицы n_{wt}, называемое срезом, могло поместиться в оперативной памяти узла. Система загружает указанный срез из распределённого хранилища, содержащего все параметры n_{wt}, после чего на узле обрабатываются только те термы блока, которые покрываются текущим срезом модели. Остальные игнорируются. Как только сэмплирование для термов текущего среза завершается, система загружает следующий срез и возобновляет сэмплирование.

Сетевые коммуникация производятся в фоновом режиме. Обновления счётчиков n_{wt} вычисляются локально и отправляются в хранилище асинхронно, как в Y!LDA. Для хранения счётчиков используется гибридное решение — для 10% самых частых термов строки матриц хранятся в плотном виде, для остальных

— в разреженном.

Light LDA реализован поверх распределённой системы Petuum [57], представляющей собой платформу для реализации параллельных алгоритмов машинного обучения. Каждый обработчик на вычислительном узле работает в многопоточном режиме, для чего обрабатываемый блок данных разделяется на непересекающиеся подблоки, которые обрабатываются отдельными потоками. Срез модели в этой схеме является общим для всех потоков и доступен только для чтения.

ZenLDA. ZenLDA [48] — ещё одна реализация параллельного алгоритма сэмплирования Гиббса для модели LDA на Spark. Её ключевой особенностью является представление данных и модели в виде двудольного ненаправленного графа. Граф имеет два типа вершин — термы и документы. Ребро между вершиной-документом и вершиной-термом означает, что данный терм встречается в данном документе. Счётчики n_{wt} хранятся раздельно в виде разреженных векторов, каждый из которых прикреплён к своей вершине-терму. Для счётчиков документов n_{td} всё аналогично. Значения Z привязаны к рёбрам между соответствующими вершинами-термами и вершинами-документами. с каждым ребром связан вектор таких значений, поскольку каждый терм может встретиться в документе более одного раза.

Параллельность вычислений достигается разделением графа на части, обрабатываемые вычислительными узлами одновременно и независимо, что обеспечивает распределённость как данных, так и модели. В реализации используется модификация алгоритма «degree-based hashing», производящего разделение графа по вершинам [48]. ZenLDA является синхронным итеративным алгоритмом.

Для ускорения алгоритма и уменьшения потребления ресурсов используется ряд оптимизаций. Во-первых, это разреженная инициализация вектора Z путём случайного сужения множества возможных тем для сэмплирования, ко-

Реализация	A	B	C	D	E	F	G	H	I	J	K	L
AD-LDA	MPI	C++	+	+								
PLDA	MPI	C++	+	+								+
AS-LDA	MPI	C	+		+							
PLDA+	RPC [59]	C++	+		+			+				
Y!LDA	Memcached + многопоточность	C++	+		+							+
VW LDA	Однопоточность	C++				+	+					
Gensim	Многопоточность	Python			+	+	+					
FOEM-LDA	Однопоточность	C					+	+			+	
Mr.LDA	Hadoop + MapReduce	Java	+	+								+
Spark-LDA	Spark	Scala	+	+				+				+
Peacock	RPC + многопоточность	Go	+	+				+				
Light LDA	Petuum + многопоточность	C++	+		+			+	+	+		+
ZenLDA	Spark	Scala	+	+				+	+	+	+	+

Таблица 2.1. Сравнение особенностей представленных в обзоре реализаций. Столбцы: A — платформа или способ организации вычислений; B — использованный язык программирования (основной, без учёта языков интерфейсов); C — распределённые хранение или обработка коллекции; D — синхронная параллельная обработка; E — асинхронная параллельная обработка; F — хранение параметров документов во внешней памяти; G — онлайновая обработка; H — распределённое или оптимизированное хранение модели; I — разреженное хранение модели; J — разреженная инициализация модели; K — разреженная обработка термов в документах; L — обеспечение отказоустойчивости.

торая приводит к получению менее плотной матрицы n_{wt} на первых итерациях. Во-вторых — исключение из обработки с некоторой вероятностью термов в документах, для которых значение присвоенной темы в векторе Z не изменилось с прошлой итерации. В-третьих, это использование гибридного метода хранения параметров для частых и редких термов (такого же, как в Light LDA). Кроме того, ZenLDA переопределяет ряд стандартных функций библиотеки GraphX [58] для использования всех ядер машины в параллельной обработке одной части графа.

2.3. Выводы из обзорного материала

В данном разделе описаны различные подходы к повышению эффективности процесса обучения тематических моделей. В таблице 2.1 сведены воедино все техники повышения эффективности и реализации алгоритмов. Заметим, что техники одинаково применимы к EM-подобным алгоритмам тематического моделирования PLSA, MAP, VB, GS, ARTM.

Обзор показывает, что не существует идеального решения, объединяющего достоинства всех подходов, поскольку оптимизация одних характеристик часто приводит приводить к ухудшению других. Тем не менее, к числу модификаций, присутствие которых в большинстве случаев оказывается полезным, можно отнести наличие параллельности вычислений (как на уровне потоков, так и на уровне процессов), параллелизм модели, отсутствие необходимости хранения параметров, связанных с документами, наличие отказоустойчивости.

Описанный материал лежит в основе алгоритмических и архитектурных решений, предлагаемых в следующей главе. Стоит отметить, что они связаны в первую очередь с вычислениями на CPU, поэтому реализации алгоритмов обучения моделей на GPU (например, [60]) не вошли в обзорный материал.

Глава 3

Реализации EM-алгоритма в библиотеке BigARTM

Все реализации алгоритмов обучения тематических моделей, рассмотренные в предыдущей главе, предназначены для обучения модели LDA. В главе 1 показано, что подход ARTM позволяет производить обучение с гораздо большей гибкостью и получать в итоге модели более высокого качества, однако отсутствие эффективной программной реализации не позволяло в полной мере пользоваться этими преимуществами.

В данной главе описывается устройство высокопроизводительной библиотеки BigARTM, предназначенной для построения аддитивно регуляризованных моделей на одном вычислительном узле. Рассматривается эволюция реализаций лежащего в её основе параллельного EM-алгоритма. Демонстрируется последовательный переход от первоначального синхронного оффлайнового алгоритма к асинхронному онлайновому варианту Async.

Производится анализ достоинств и недостатков алгоритма Async, после чего демонстрируется его детерминированная и более производительная версия DetAsync.

В дополнение к обычному режиму работы синхронных реализаций и алгоритма DetAsync предлагается режим разреженного хранения параметров и обработки данных. В условиях разреженных моделей, которые часто встречаются на практике, он позволяет существенно ускорить процесс обучения и в ряде случаев сократить пиковый объём потребляемой оперативной памяти.

Алгоритм 3.1.1. ProcessDocument(d, Φ)

Входные данные: документ $d \in D$, матрица $\Phi = (\phi_{wt})$;

Выходные данные: матрица $(\hat{n}_{wt})$, вектор (θ_{td}) для документа d;

1 инициализировать $\theta_{td} := \frac{1}{|T|}$ для всех $t \in T$;

2 **повторять**

3 $\quad p_{tdw} := \underset{t \in T}{\text{norm}}\big(\phi_{wt}\theta_{td}\big)$ для всех $w \in d$ и $t \in T$;

4 $\quad \theta_{td} := \underset{t \in T}{\text{norm}}\big(\sum_{w \in d} n_{dw} p_{tdw} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}\big)$ для всех $t \in T$;

5 **до тех пор, пока** θ_d не сойдётся;

6 $\hat{n}_{wt} := n_{dw} p_{tdw}$ для всех $w \in d$ и $t \in T$;

3.1. Синхронные алгоритмы

BigARTM [14, 15] — это библиотека с открытым программным кодом для построения регуляризованных тематических моделей больших текстовых коллекций. Помимо нескольких параллельных алгоритмов обучения, она содержит набор регуляризаторов и метрик качества, которые используются в экспериментах в этой и последующих главах. BigARTM написана на C++11 с использованием технологии protobuf [61]. Реализованные в ней алгоритмы обучения моделей являются пакетными и работают в многопоточном режиме: данные для библиотеки сохраняются на диске или в памяти в виде специальных пакетов с заданным числом документов и обрабатываются параллельно. Один поток в каждый момент времени может обрабатывать один пакет. Рассмотрим прежде всего две базовые версии алгоритма: оффлайновый и синхронный онлайновый.

Оффлайновый алгоритм. Базовым вариантом EM-алгоритма для модели ARTM является оффлайновый (Алгоритм 3.1.2). Он основывается на функции `ProcessDocument` (Алгоритм 3.1.1), которая соответствует уравнениям (1.9) и (1.11) решения задачи ARTM. Эта функция получает на вход фиксированную матрицу Φ и вектор n_{dw} частот термов для заданного документа $d \in D$.

Алгоритм 3.1.2. Offline ARTM

Входные данные: коллекция D;

Выходные данные: матрица $\Phi = (\phi_{wt})$;

1 инициализировать (ϕ_{wt});

2 создать пакеты $D := D_1 \sqcup D_2 \sqcup \cdots \sqcup D_B$;

3 **повторять**

4 $\quad (n_{wt}) := \sum_{b=1,\ldots,B} \sum_{d \in D_b} \mathsf{ProcessDocument}(d, \Phi)$;

5 $\quad (\phi_{wt}) := \underset{w \in W}{\mathrm{norm}}(n_{wt} + \phi_{wt} \frac{\partial R}{\partial \phi_{wt}})$;

6 **до тех пор, пока** (ϕ_{wt}) не сойдётся;

Выходными данными являются распределение на темах данного документа θ_{td} и матрица накопленных счётчиков $\hat{n}_{wt}$ размера $|d| \times |T|$, где $|d|$ обозначает число уникальных термов в документе d. Стоит отметить, что на практике цикл 2–5 повторяется не до сходимости вектора θ_d текущего документа, а фиксированное заранее число раз, называемое количеством проходов по документу.

`ProcessDocument` может также быть полезной как отдельная операция, позволяющая получать векторы θ_{td} для новых документов, но в оффлайновом алгоритме она используется в качестве базового блока обработки в EM-алгоритме, и предназначена для вычисления обновлений матрицы n_{wt}.

Алгоритм `Offline ARTM` проходит по всей коллекции текстов, вызывая функцию `ProcessDocument` для каждого документа $d \in D$, а затем агрегирует результирующие матрицы $\hat{n}_{wt}$ в итоговую матрицу n_{wt} размера $|W| \times |T|$.

После каждого прохода по коллекции матрицы Φ обновляется в соответствии с уравнением (1.10).

На шаге 2 производится разделение коллекции D на пакеты D_b, что не является обязательным для самого оффлайнового алгоритма, а является особенностью реализации. В целях повышения производительности внешний цикл по пакетам $b = 1, \ldots, B$ распараллеливается по нескольким потокам, и внутри

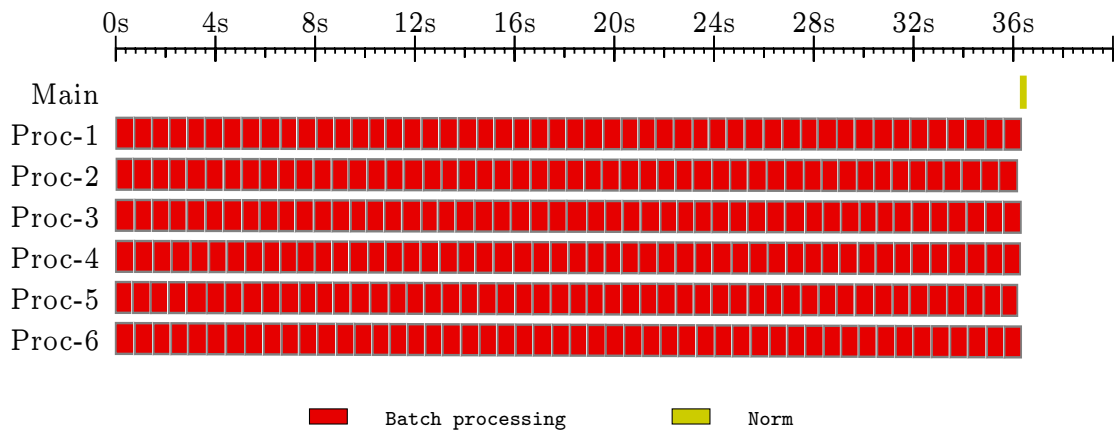

Рис. 3.1. Диаграмма Гантта `Offline ARTM` (Алгоритм 3.1.2) с 6 рабочими потоками. Элементы: `Batch processing` — время, затраченное на обработку пакета, `Norm` — время, затраченное на нормировку n_{wt}.

каждого пакета внутренний цикл по документам $d \in D_b$ выполняется в рамках одного потока.

Стоит обратить внимание на то, что значения θ_{td} появляются лишь внутри функции `ProcessDocument`. Это приводит к эффективному использованию памяти, поскольку реализация никогда не хранит целую матрицу Θ. Вместо этого значения θ_{td} пересчитываются с нуля на каждом проходе по коллекции.

На рис. 3.1 можно увидеть диаграмму Гантта для `Offline ARTM`. В этой и последующей диаграммах показана одна итерация EM-алгоритма на данных NYTimes [62] ($|D| = 300\text{K}$, $|W| = 102\text{K}$) в модели с $|T| = 16$ темами. Прямоугольники `ProcessBatch` соответствуют времени, потраченному на обработку одного пакета. Финальный прямоугольник `Norm`, выполняющийся в главном потоке, соответствует времени, затраченному на шаг 4 в Алгоритме 3.1.2, где счётчики n_{wt} нормируются для создания новой матрицы Φ.

Синхронный онлайновый алгоритм. `Online ARTM` (Алгоритм 3.1.3) является обобщением онлайнового вариационного EM-алгоритма, предложенного в [50] для модели LDA. Онлайновый алгоритм улучшает сходимости оффлайнового за счёт пересчёта матрицы Φ, производимого не в конце обработки всей

Алгоритм 3.1.3. Online ARTM

Входные данные: коллекция D, гиперпараметры η, τ_0, κ;

Выходные данные: матрица $\Phi = (\phi_{wt})$;

1 создать пакеты $D := D_1 \sqcup D_2 \sqcup \cdots \sqcup D_B$;

2 инициализировать (ϕ^0_{wt});

3 **цикл** $i = 1, \ldots, \lfloor B/\eta \rfloor$ **выполнять**

4 $\quad (\hat{n}^i_{wt}) := \mathsf{ProcessBatches}(\{D_{\eta(i-1)+1}, \ldots, D_{\eta i}\}, \Phi^{i-1})$;

5 $\quad \rho_i := (\tau_0 + i)^{-\kappa}$;

6 $\quad (n^i_{wt}) := (1 - \rho_i) \cdot (n^{i-1}_{wt}) + \rho_i \cdot (\hat{n}^i_{wt})$;

7 $\quad (\phi^i_{wt}) := \underset{w \in W}{\mathrm{norm}}(n^i_{wt} + \phi^{i-1}_{wt} \frac{\partial R}{\partial \phi_{wt}})$;

коллекции, а в конце обработки некоторой порции пакетов. Для упрощения обозначений введём следующую тривиальную функцию:

$$\texttt{ProcessBatches}(\{D_b\}, \Phi) = \sum_{D_b} \sum_{d \in D_b} \texttt{ProcessDocument}(d, \Phi).$$

Она агрегирует результаты `ProcessDocument` для заданного множества пакетов при фиксированной матрице Φ.

В онлайновом алгоритме разбиение коллекции $D := D_1 \sqcup D_2 \sqcup \cdots \sqcup D_B$ на пакеты играет гораздо более важную роль, чем в алгоритме оффлайновом, поскольку различные разбиения приводят к различным результатам.

На шаге 6 новые значения n^{i+1}_{wt} вычисляются как выпуклая комбинация старых значений n^i_{wt} и значения $\hat{n}^i_{wt}$, полученного по только что обработанным пакетам. Старые счётчики n^i_{wt} умножаются на множитель $(1 - \rho_i)$, зависящий от номера итерации. Общепринятая стратегия состоит в использовании $\rho_i = (\tau_0 + i)^{-\kappa}$, где стандартные значения τ_0 лежат в диапазоне от 64 до 1024, а κ — от 0.5 до 0.7 [50]. В библиотеке BigARTM значениями по умолчанию являются $\tau_0 = 1024$ и $\kappa = 0.7$, для упрощения настройки именно они используются во

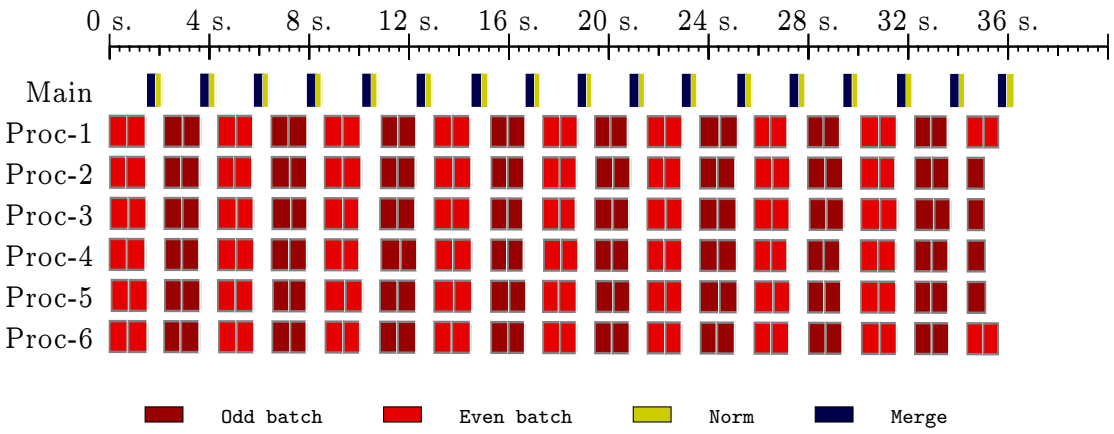

Рис. 3.2. Диаграмма Гантта `Online ARTM` (Алгоритм 3.1.3) с 6 рабочими потоками. Элементы: `Even batch`/`Odd batch` — время, затраченное на обработку пакета (чередование означает смену версии матрицы Φ), `Merge` — время, затраченное на слияние n_{wt} и $\hat{n}_{wt}$, `Norm` — время, затраченное на нормировку n_{wt}.

всех экспериментах с онлайновыми алгоритмами, представленными в данной работе.

Так же, как и в оффлайновом алгоритме, внешний цикл по пакетам документов $D_{\eta(i-1)+1}, \ldots, D_{\eta i}$ выполняется параллельно в нескольких потоках. Проблема такого подхода в том, что во время шагов 5–7 алгоритма `Online ARTM` рабочие потоки простаивают.

Потоки не могут начать обработку следующей порции пакетов, поскольку новая версия матрицы Φ ещё не готова. Результатом этого является неэффективное использование процессорного времени, обычная диаграмма Гантта для алгоритма `Online ARTM` показана на рис. 3.2.

Прямоугольники `Even batch` и `Odd batch` оба соответствуют шагу 4 и обозначают версию матрицы Φ^i (чётное i или нечётное i). Прямоугольник `Merge` соответствует времени, затраченному на слияние n_{wt} и $\hat{n}_{wt}$. `Norm`, как и выше, обозначает время, затраченное на нормализацию счётчиков n_{wt} для получения новой матрицы Φ, которая будет использована во время следующей итерации.

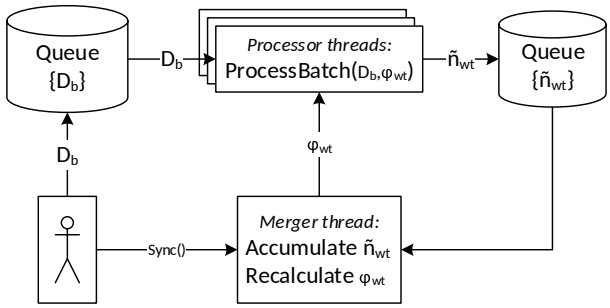

Рис. 3.3. Схема компонентов BigARTM (Async)

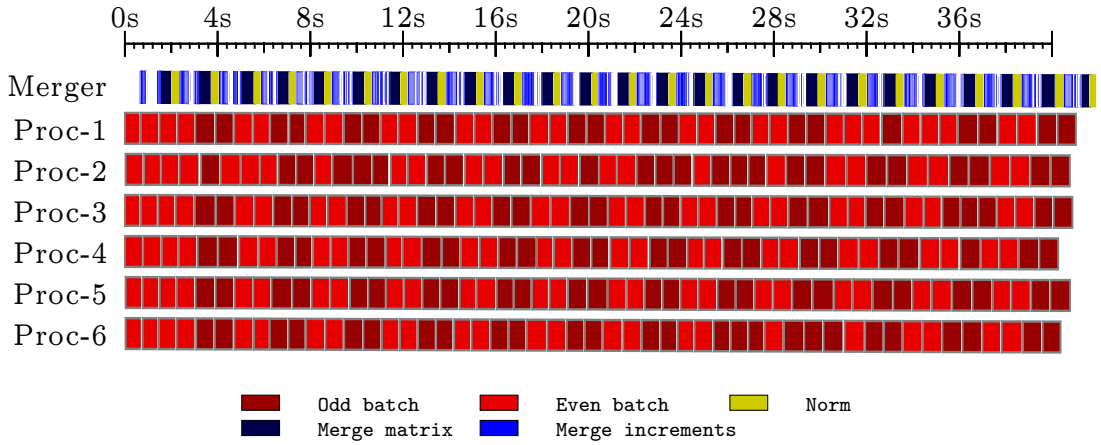

Рис. 3.4. Диаграмма Гантта `Async ARTM` с 6 рабочими потоками — нормальная ситуация. Элементы: `Even batch`/`Odd batch` — время, затраченное на обработку пакета (чередование означает смену версии матрицы Φ), `Merge matrix` — время, затраченное на слияние n_{wt} и $\hat{n}_{wt}$, `Merge increments` — время, затраченное на слияние матриц обновлений n_{wt} из очереди, `Norm` — время, затраченное на нормировку n_{wt}.

3.2. Алгоритм Async

Алгоритм `Async ARTM` [14] позволяет решить проблемы обычного онлайнового синхронного алгоритма, связанные с простоем потоков при обновлении матрицы Φ. Общая схема работы алгоритма показана на рис. 3.3. Выделенный поток загрузки данных (называемый `DataLoader`) производит подгрузку пакетов с диска в специализированную очередь задач. Рабочие потоки выполняют процедуру `ProcessBatches`, причём делают это асинхронно, то есть параллельно с процессом обновления матрицы Φ. Каждый поток, обработав свой пакет и получив обновления счётчиков $\hat{n}_{wt}$, отправляет их в другую очередь. Другой

выделенный поток, занимающийся слиянием обновлений (`Merger`), контролирует заполненность этой очереди. В ситуации, когда количество обновлений достигает заданного порога η, он извлекает их из очереди, сливает в одну матрицу счётчиков и производит шаги 5–7 алгоритма `Online ARTM` (Алгоритм 3.1.3). Как уже отмечено, вся эта процедура выполняется в фоновом режиме, без остановки обработки новых пакетов.

Указанный алгоритм показывает высокую производительность в сравнению с аналогичными инструментами [14], но обладает рядом недостатков.

Первая проблема состоит в том, что алгоритм не детерминирует порядок слияния счётчиков $\hat{n}_{wt}$. Этот порядок зависит от порядка обработки пакетов и меняется от запуска к запуску. Это приводит к тому, что и результирующая матрица Φ от запуска к запуску может быть различной.

Другая проблема, связанная с `Async ARTM`, состоит в том, что хранение счётчиков $\hat{n}_{wt}$ в очереди может существенно увеличить потребление памяти, а необходимость их слияния может привести к тому, что поток `Merger` станет узким местом алгоритма с точки зрения производительности.

При правильном подборе параметров эффективность подобного алгоритма высока, что можно видеть на рис. 3.4. Однако несложно подобрать и такой набор параметров (например, слишком малый размер пакета или маленькое число внутренних итераций в `ProcessDocument`), который приведёт к перегрузке потока слияния. В такой ситуации диаграмма Гантта примет вид, показанный на рис. 3.5: большинство рабочих потоков простаивают, поскольку в очереди нет места для новых счётчиков n_{wt}, а поток `Merger` не успевает справляться со своими задачами.

3.3. Алгоритм DetAsync

`DetAsync ARTM` [17] (Алгоритм 3.3.1) основывается на двух новых функциях, `AsyncProcessBatches` и `Await`.

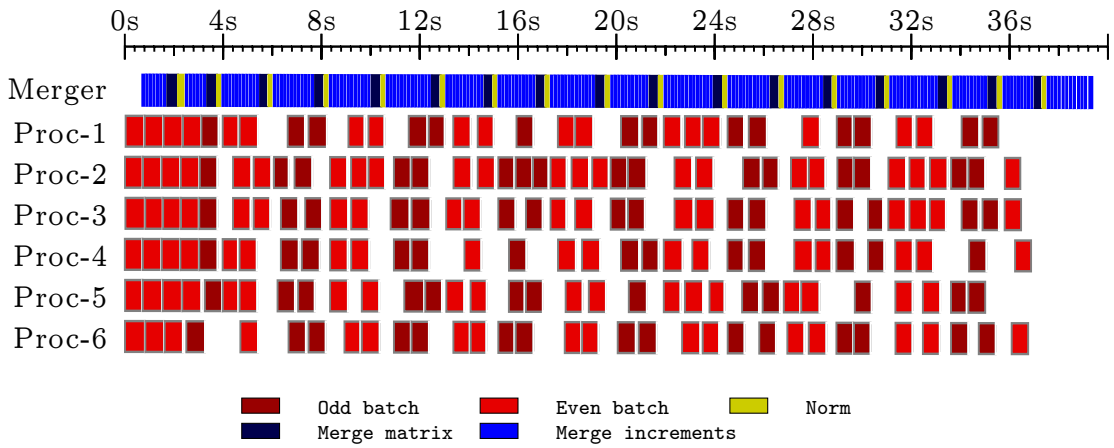

Рис. 3.5. Диаграмма Гантта `Async ARTM` с 6 рабочими потоками — ситуация проблем с производительности. Элементы: `Even batch`/`Odd batch` — время, затраченное на обработку пакета (чередование означает смену версии матрицы Φ), `Merge matrix` — время, затраченное на слияние n_{wt} и $\hat{n}_{wt}$, `Merge increments` — время, затраченное на слияние матриц обновлений n_{wt} из очереди, `Norm` — время, затраченное на нормировку n_{wt}.

Первая во всём эквивалентна описанной ранее `ProcessBatches`, за исключением того, что она берёт задачу на асинхронную обработку и немедленно возвращает управление в вызвавший поток. Её результатом является future-объект (например, `std::future` из стандарта `C++11`), который может быть затем передан в вызов `Await` для получения вычисленного результата, в рассматриваемом случае это счётчики $\hat{n}_{wt}$.

Между вызовами `AsyncProcessBatches` и `Await` алгоритм может выполнять различную вспомогательную работу, пока рабочие потоки в фоновом режиме производят вычисление матрицы $\hat{n}_{wt}$.

Для вычисления $\hat{n}_{wt}^{i+1}$ `DetAsync ARTM` использует матрицу Φ^{i-1} с предыдущего обновления. Это добавляет некоторое запаздывание между моментом вычисления очередной версии матрицы Φ и моментом её использования, что даёт алгоритму в результате дополнительную гибкость в распределении нагрузки на рабочие потоки. Шаги 3 и 5 — это технический трюк, направленный на реализацию описанной идеи с запаздыванием.

Добавление запаздывания может негативно сказаться на сходимости алго-

Алгоритм 3.3.1. DetAsync ARTM

Входные данные: коллекция D, параметры η, τ_0, κ;

Выходные данные: матрица $\Phi = (\phi_{wt})$;

создать пакеты $D := D_1 \sqcup D_2 \sqcup \cdots \sqcup D_B$;
инициализировать (ϕ^0_{wt});
$F^1 := \mathsf{AsyncProcessBatches}(\{D_1, \ldots, D_\eta\}, \Phi^0)$;
цикл $i = 1, \ldots, \lfloor B/\eta \rfloor$ **выполнять**
    если $i \neq \lfloor B/\eta \rfloor$ **тогда**
        $F^{i+1} := \mathsf{AsyncProcessBatches}(\{D_{\eta i+1}, \ldots, D_{\eta i+\eta}\}, \Phi^{i-1})$;
    $(\hat{n}^i_{wt}) := \mathsf{Await}(F^i)$;
    $\rho_i := (\tau_0 + i)^{-\kappa}$;
    $(n^i_{wt}) := (1 - \rho_i) \cdot (n^{i-1}_{wt}) + \rho_i \cdot (\hat{n}^i_{wt})$;
    $(\phi^i_{wt}) := \underset{w \in W}{\mathrm{norm}}(n^i_{wt} + \phi^{i-1}_{wt} \frac{\partial R}{\partial \phi_{wt}})$;

ритма в сравнении с `Online Async`. Например, в `AsyncProcessBatches` начальная матрица Φ^0 используется дважды, в то время как последние две матрицы $\Phi^{\lfloor B/\eta \rfloor - 1}$ и $\Phi^{\lfloor B/\eta \rfloor}$ вообще не будут использованы.

С другой стороны, асинхронный алгоритм позволяет добиться более высокой степени загрузки ядер, что наглядно продемонстрировано на рис. 3.6.

В этом состоит некоторый компромисс между сходимостью и загрузкой CPU, и он изучается подробнее в экспериментальной части.

С точки зрения реализации `DetAsync ARTM` претерпел ряд изменений по сравнению с `Async ARTM`. Прежде всего изменения затронули способ агрегации счётчиков $\hat{n}_{wt}$, получаемых при обработке каждого пакета. В DetAsync отсутствует выделенный поток `Merger` и связанная с ним очередь, вместо этого счётчики $\hat{n}_{wt}$ вносятся напрямую в результирующую матрицу n_{wt} асинхронно всеми рабочими потоками, что существенно редуцирует пиковое потребление памяти алгоритма по сравнению с Async. Для синхронизации доступа на запись необхо-

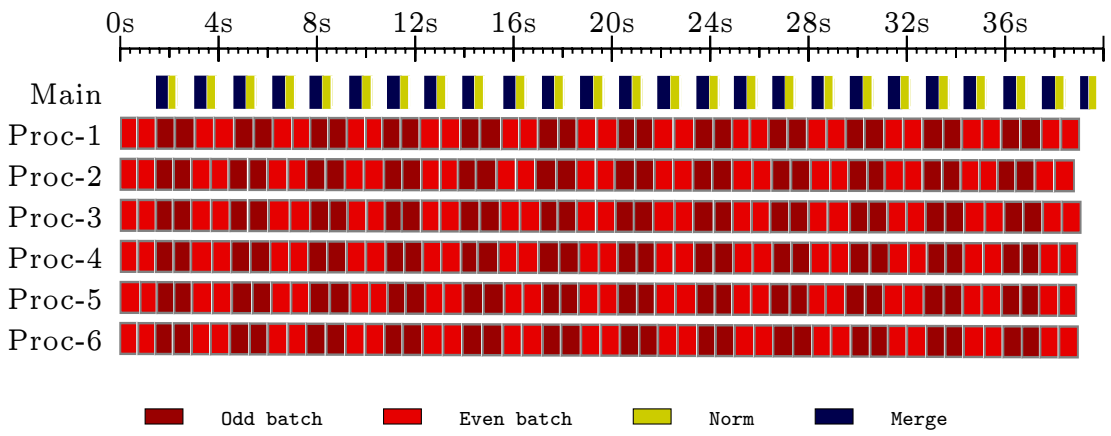

Рис. 3.6. Диаграмма Гантта для `DetAsync ARTM` (алгоритм 3.3.1) с 6 рабочими потоками. Элементы: `Even batch`/`Odd batch` — время, затраченное на обработку пакета (чередование означает смену версии матрицы Φ), `Merge` — время, затраченное на слияние n_{wt} и $\hat{n}_{wt}$, `Norm` — время, затраченное на нормировку n_{wt}.

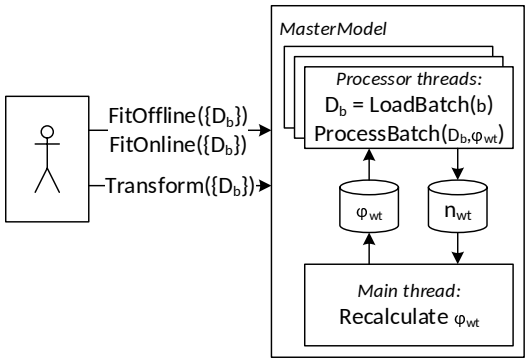

Рис. 3.7. Схема компонентов BigARTM (DetAsync)

димо обеспечить невозможность возникновения ситуации, в которой два потока одновременно производят запись в одну строку матрицы n_{wt}. Это достигается с помощью спин-локов l_w, по одному на каждый терм из словаря W. В конце вызова `ProcessDocument` производится итерирование по всем $w \in d$, для каждого терма производится блокировка соответствующего спин-лока, добавление $\hat{n}_{wt}$ к n_{wt} и разблокировка спин-лока. Этот подход схож с предлагаемым в [42], где подобная система организована в распределённой среде.

Также стоит отметить, что в DetAsync ликвидирован выделенный поток загрузки данных, процесс загрузки производится непосредственно самими рабочими потоками, что упростило архитектуру без потери производительности.

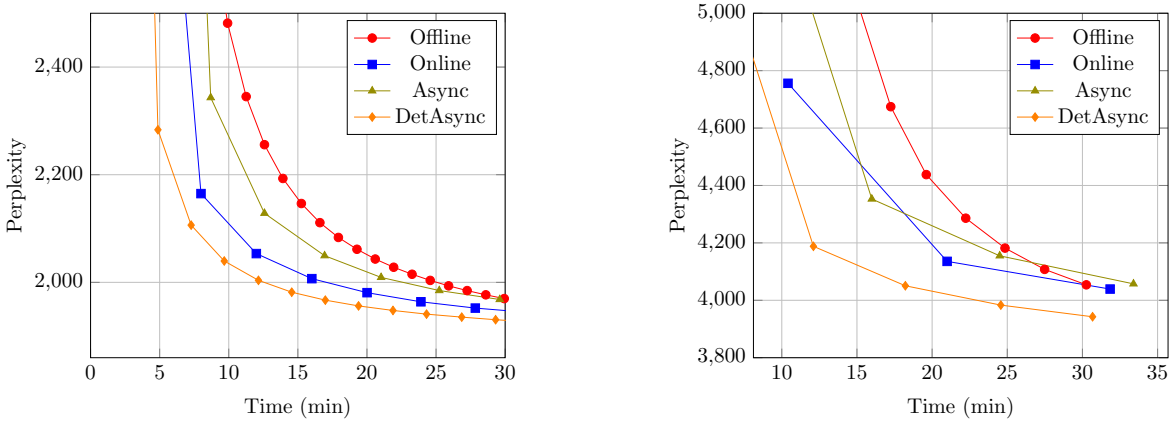

Рис. 3.8. График перплексии от времени работы для Pubmed (слева) и Wikipedia (справа), $|T| = 100$ topics

Общая схема компонентов BigARTM после внедрения DetAsync показана на рис. 3.7. За счёт того, что набор пакетов, обрабатываемый между обновлениями Φ, для фиксированного набора данных не меняется, алгоритм с одинаковыми гиперпараметрами от запуска к запуску будет давать один и тот же результат.

3.4. Экспериментальные результаты

Сравнение алгоритмов в BigARTM. Прежде всего следует сравнить между собой описанные выше алгоритмы `Offline ARTM`, `Online ARTM`, `Async ARTM` и `DetAsync ARTM` [17]. Стандартной метрикой качества построения тематической модели является перплексия, которая отражает степень сходимости модели и определяется как

$$\mathcal{P}(D;\Phi,\Theta) = \exp\left(-\frac{1}{n}\sum_{d\in D}\sum_{w\in d} n_{dw}\ln p(w\,|\,d)\right), \qquad n = \sum_{d} n_d. \tag{3.1}$$

Более низкое значение перплексии соответствует более качественному результату.

В описываемом эксперименте для оценки эффективности сравниваются итоговые значения перплексии, полученные каждым из алгоритмов за фиксированный отрезок времени. В качестве экспериментального набора данных используются коллекция статей англоязычной Википедии (Wiki) [63] ($|D| = 3.7$M

статей, $|W| = 100$K слов в словаре) и коллекция аннотаций Pubmed [64] ($|D| = 8.2$M аннотаций, $|W| = 141$K слов в словаре). Все модели обучались на вычислительном сервере с процессором Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.00GHz с 16 физическими ядрами в совокупности (32 рабочих потока).

Рис. 3.8 показывает перплексию как функцию от времени, потраченного описанными выше алгоритмами на обучение. Каждая точка на графике соответствует моменту завершения алгоритмом очередного прохода по коллекции. Каждому алгоритму было выделено на работу 30 минут. Можно видеть, что на обеих коллекциях алгоритм DetAsync показал наилучший результат.

Сравнение BigARTM с другими библиотеками. Следующим шагом является сравнение производительности онлайновых алгоритмов в BigARTM с ближайшими конкурентами — библиотеками Gensim и Vowpal Wabbit [20]. Стоит отметить, что перед проведением этого сравнения была проведена техническая оптимизация реализации алгоритма DetAsync: был устранён ряд ошибок и недоработок, приводивших к излишнему потреблению вычислительных ресурсов и памяти.

Сравнение производилось по времени обучения и итоговой перплексии на основании запуска одного онлайнового прохода по уже описанной коллекции текстов англоязычной Википедии. Перплексия вычислялась на текстовом наборе из 100 тысяч документов.

В Gensim реализованы две версии LDA: `LdaModel`, `LdaMulticore`. Первая из них является однопоточной, вторая производит вычисления параллельно, она использовалась во всех экспериментов, где число рабочих потоков было больше одного. Реализация Vowpal Wabbit LDA является однопоточной, поэтому с ней сравнение производилось только в однопоточном режиме.

Для чистоты эксперимента BigARTM запускался в режиме LDA, то есть со сглаживающими регуляризаторами для Φ и Θ с константными векторами гиперпараметров, состоящими из значений $\beta = 1/|T|$ и $\alpha = 1/|T|$ соответственно.

	Gensim LDA	VW LDA	BigARTM Online	BigARTM DetAsync
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)

Таблица 3.1. Сравнение производительности BigARTM, Vowpal Wabbit LDA и Gensim. В ячейках указано затраченное время (в минутах) и итоговое значение тестовой перплексии. Столбцы: P — количество рабочих потоков, T — количество тем.

Эти же гиперпараметры использовались при запуске Vowpal Wabbit и Gensim.

Эксперименты производились на сервере Dell Precision T5600 с процессором Intel(R) Xeon(R) CPU E5-2650 @ 2.00GHz, 8 физических ядер, 16 потоков.

В таблице 3.1 показаны результаты сравнения производительности библиотек для случаев различного числа тем и рабочих потоков. Видно, что в однопоточном случае BigARTM работает в 3–5 раз быстрее Gensim и в 2–3 раза быстрее Vowpal Wabbit. Значение перплексии при этом получается на уровне Gensim, для Vowpal Wabbit оно ощутимо выше. Сравнение в случае использования нескольких рабочих потоков демонстрирует ещё большее ускорение (в 5–10 раз для Vowpal Wabbit и в 10–20 для Gensim), качество модели с точки зрения перплексии у BigARTM становится выше, чем у Gensim, в случае синхронного алгоритма и остаётся на том же уровне в случае DetAsync.

Дополнительно для BigARTM демонстрируются замеры скорости работы и итоговой перплексии для случаев большого числа тем (2000 и 5000), видно, что и в этом случае алгоритмы работают за разумное время (таблица 3.2).

Легко заметить, что во всех случаях асинхронный алгоритм работает суще-

Алгоритм/Темы	2000	5000
BigARTM Online	166m (2377)	399m (1942)
BigARTM DetAsync	119m (2645)	281m (2216)

Таблица 3.2. Запуск BigARTM с большим числом тем. В ячейках указано затраченное время (в минутах) и итоговое значение тестовой перплексии.

ственно быстрее синхронного, но при этом строит менее качественную модель, однако выше уже было продемонстрировано, что за фиксированный промежуток времени `Async ARTM` уменьшает перплексию сильнее, чем `Online ARTM`.

3.5. Разреженные алгоритмы

Одним из важных свойств тематической модели является её *интерпретируемость*. Будучи плохо формализованным понятием, оно содержательно означает то, что если в теме такой модели выделить набор наиболее вероятных термов, то для эксперта этот набор будет описывать некоторую тематику. В [31] предлагается формализация понятия интерпретируемости, предполагающая наличие у тем *лексических ядер* — множеств термов, которые характерны для определённой предметной области, часто употребляются совместно, с большой вероятностью встречаются в данной теме и почти не употребляются в других темах. Таким образом определяемая интерпретируемость тесно связана с разреженностью модели. Среди множества решений задачи (1.8) наибольший интерес представляют те матрицы Φ и Θ, которые обладают подходящей структурой разреженности: в каждой теме имеется небольшой набор ненулевых элементов, и наборы эти для разных имеют как можно меньшее пересечение. Получить модель такого вида можно с помощью правильно подобранного набора регуляризаторов, например, путём использования регуляризатора декорреляции тем (1.14).

Разреживающая регуляризация не является единственным источником роста числа нулевых параметров в тематических моделях. Даже в том случае, если

стартовая матрица Φ является плотной, в отсутствие принудительного сглаживания с итерациями количество нулевых или близких к нулю элементов в ней будет расти само по себе по мере сходимости алгоритма и уточнения модели. Эта особенность не раз отмечалась в научной литературе, авторы ряда реализаций алгоритмов обучения моделей в своих работах стараются использовать её для сокращения времени счёта и потребления оперативной памяти [47, 48].

Таким образом, вне зависимости от природы своего появления, высокая степень разреженности является частым свойством тематических моделей. Описанные же выше алгоритмы, реализованные в библиотеке BigARTM, работают со всеми моделями одинаково, никак не учитывая того, что значительная часть параметров может быть безболезненно исключена из процесса обучения.

Плотное хранение параметров в BigARTM. Прежде всего рассмотрим то, как параметры модели хранились в BigARTM до предлагаемых далее модификаций. Библиотека использует одну и ту же структуру для хранения матриц Φ и n_{wt}, называемую Φ-подобной матрицей. Важно заметить, что помимо указанных матриц, BigARTM хранит также матрицу поправок регуляризаторов r_{wt}. Она нужна для аккумуляции второго слагаемого в правой части формулы для вычисления ϕ_{wt} в (1.10) и создаётся только в том случае, если в процессе обучения используется хотя бы один регуляризатор. Эта матрица также хранится как Φ-подобная. Очевидно, что в ситуации фиксированных набора тем и словаря коллекции все три указанных матрицы имеют одинаковые размеры.

Φ-подобная матрица представляла собой плотный вещественнозначный массив, в котором значения хранятся построчно. Это значит, что для каждого терма w соответствующие параметры расположены в оперативной памяти в виде одного последовательного блока. Очевидным недостатком такого подхода является необходимость хранить большое количество нулей в случае разреженной матрицы. Ключевым преимуществом — возможность локального последовательного доступа при обработке циклов по множеству тем. Такой цикл выпол-

няется в формуле (1.9), на которую в процессе обучения приходится большая часть вычислений.

Локальный последовательный доступ даёт существенный рост скорости работы алгоритма по сравнению с доступом случайным (то есть ситуации, в которой обработка логически последовательного блока данных требует обращения в различные части оперативной памяти). Однако в случае разреженной модели значительная часть вычислений оказывается бесполезной, поскольку большинство участвующих значений равны нулю. Очевидно, что примитивное решение на основе условного оператора только усугубит проблему, а не решит её: ветвление кода является ещё более сложной и затратной операцией, чем умножение и сложение. По схожим причинам не подойдёт популярный метод хранения разреженных данных в хэш-таблицах.

В результате появляется необходимость в решении, которое позволит хранить и обрабатывать только ненулевые элементы Ф-подобной матрицы и сохранит при этом локальность доступа к памяти.

Разреженные хранение и обработка. Одним из возможных решений является хранение Ф-подобной матрицы в CSR-формате (Compressed Sparse Row, [65]). Такой подход устраняет необходимость в хранении нулевых элементов и даёт возможность эффективно организовать циклы по ненулевым элементам своих строк. Однако для описанной задачи он является непригодным, поскольку в этом случае возникает необходимость в фиксации структуры разреженности, т.е. позиций ненулевых элементов в матрице. В случае матрицы n_{wt} это не представляется возможным, поскольку значения обновлений счётчиков агрегируются в ней в течение всего E-шага. Следует также учитывать, что применение комбинаций регуляризаторов может привести к неравномерной структуре разреженности модели, когда часть строк может быть содержать много нулей, а часть — мало. В такой ситуации полезно иметь возможность хранить первые в разреженном виде, а вторые — в плотном.

По этим причинам в [21] предлагается гибридный формат хранения (что идейно схоже с подходом, использованным в [47] и [48]), в котором единицей хранения является не вся матрица, а одна её строка. В рамках этого подхода массив S длины m может храниться в одной из двух форм, разреженной или плотной. Текущая форма определяется числом k ненулевых элементов в нём. В плотной форме массив хранится в виде одного последовательного блока, как и раньше. Для хранения в разреженном формате предлагается использовать три массива: V, I и M. Вещественнозначный массив V содержит все ненулевые элементы массива S в их исходном порядке. Целочисленный массив I хранит для каждого элемента V его индекс в исходном массиве. M представляет собой битовый массив длины m; его элементы равны единице или нулю в зависимости от того, превосходят ли соответствующие им в S значения ноль или нет.

Массив M требуется для организации эффективного ($O(1)$) случайного доступа к нулевым элементам S. Массив индексов I за счёт двоичного поиска даёт возможность обратиться к ненулевым элементам S со сложностью $O(\log_2 k)$. Также он позволяет производить циклический обход ненулевых элементов. Эта особенность является наиболее важной при подсчёте скалярного произведения между ϕ_{wt} и θ_{td} на E-шаге (формула (1.9)). Векторы θ_d присутствуют явно только для документов текущего обрабатываемого пакета, поэтому их можно хранить в плотном виде. а значит, наличие индексов ненулевых элементов в строке ϕ_w позволяет вычислять скалярное произведение только по этим элементам.

Описанный подход сохраняет локальность операции, устраняет лишние вычисления без использования тяжеловесных условных операторов и уменьшает потребление памяти матрицей при росте степени её разреженности.

Последним шагом является определения порога для доли ненулевых элементов. Строки, у которых число ненулевых значений превышает порог, хранятся в плотной форме, прочие — в разреженной. Указанный порог может быть приближённо оценён на основании следующего требования: объём оперативной памяти, занимаем строкой в разреженной форме не должен превышать объё-

ма, занимаемого ею в плотной форме. Переформулируем это требование в виде простого неравенства и решим его:

$$2k + (m/8) < m \quad \Rightarrow \quad k < 0.4375\, m.$$

Исходя из этого, в качестве порога было выбрано значение 0.4.

3.6. Экспериментальные результаты

В целях демонстрации преимуществ предлагаемой оптимизации производится сравнение следующих моделей:

- обучение без использования оптимизации и принудительного разреживания (N/N);
- обучение без использования оптимизации, но с использованием принудительного разреживания (N/Y);
- обучение с использованием как оптимизации, так и принудительного разреживания (Y/Y).

Во всех случаях матрица Θ не хранится. Сравниваются время обучения и пиковый объём использованной оперативной памяти. Для разреживания моделей используется регуляризатор `SmoothSparsePhi` из библиотеки BigARTM, работающий в соответствии с формулой (1.13) (только часть, соответствующая Φ). Модели разреживаются равномерным распределением до достижения 95%-99% (в зависимости от числа тем) нулей в итоговой матрице Φ.

Обучение моделей производилось на сервере с процессором Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz и 128Гб ОЗУ. Все вычисления выполнялись в 8 потоков. Для тестирования оффлайнового алгоритма используется 200 тысяч статей англоязычной Википедии, разбитых на 100 пакетов, для онлайновых — 1 миллион статей из этой же коллекции (500 пакетов). В обоих случаях размер словаря

Темы	Модель	Память	Вр. (1)	Вр. (5)	Вр. (10)	Вр. (15)
100	N/N	**630**	**110**	**165**	230	295
	N/Y	710	120	170	240	305
	Y/Y	720	130	170	**220**	**275**
500	N/N	**1010**	**350**	605	940	1275
	N/Y	1220	370	635	965	1305
	Y/Y	1210	390	**570**	**775**	**990**
1000	N/N	**1470**	**690**	1265	1925	2595
	N/Y	1840	745	1305	1955	2640
	Y/Y	1800	755	**1070**	**1410**	**1750**
2000	N/N	**2620**	**1735**	2885	4140	5430
	N/Y	3190	1815	2945	4310	5530
	Y/Y	3160	1860	**2585**	**3315**	**3960**

Таблица 3.3. Оптимизация для разреженных моделей, оффлайновый алгоритм. Столбцы: Темы — число тем в модели, Модель — тип модели, Память — пиковое потребление памяти (в мегабайтах), Вр. (X) — время обучения (в секундах) при использовании X проходов по документу. Лучшие значения выделены жирным.

составляет 100 тысяч термов. Число тем в моделях изменяется по следующей сетке значений: 100, 500, 1000, 2000.

Оффлайновый алгоритм. При обучении оффлайнового алгоритма используется 10 проходов по коллекции, количество проходов по документу перебирается по сетке значений 1, 5, 10, 15. Коэффициент разреживающей регуляризации τ_1 в формуле (1.13) подбирался так, чтобы обеспечить заданный уровень разреженности матриц Φ и n_{wt} к концу обучения без существенного ухудшения перплексии. Для моделей с 100, 500, 1000 и 2000 тем он равен 1.0, 0.5, 0.5 и 0.2 соответственно.

Результаты проделанных экспериментов представлены в таблице 3.3. Стоит отметить, что здесь и далее изменение числа проходов по документу приводит к незначительным (в пределах 5%) флуктуациям объёма пиковой потреблённой памяти, поэтому в таблицах приводятся усреднённые значения.

Можно видеть, что предлагаемая оптимизация не даёт никаких улучше-

Темы	Модель	Память	Вр. (1)	Вр. (5)	Вр. (10)	Вр. (15)
100	N/N	**900**	**85**	115	185	230
	N/Y	990	95	125	165	200
	Y/Y	950	90	**110**	**150**	**160**
500	N/N	**1880**	410	515	770	1010
	N/Y	2100	395	495	805	1015
	Y/Y	1950	**390**	**450**	**650**	**700**
1000	N/N	**3150**	720	900	1200	1785
	N/Y	3600	740	940	1270	1780
	Y/Y	3270	**715**	**820**	**950**	**1245**
2000	N/N	5800	1330	1820	2580	3650
	N/Y	6580	1380	1890	2630	3315
	Y/Y	**5590**	**1200**	**1510**	**1840**	**2210**

Таблица 3.4. Оптимизация для разреженных моделей, синхронный онлайновый алгоритм. Столбцы: Темы — число тем в модели, Модель — тип модели, Память — пиковое потребление памяти (в мегабайтах), Вр. (X) — время обучения (в секундах) при использовании X проходов по документу. Лучшие значения выделены жирным.

ний для моделей с 100 темами или обучающихся всего с одним проходом по документу на один проход по коллекции. Модель без разреживающей регуляризации потребляет меньше всего памяти, что объясняется отсутствием в ней матрицы r_{wt}. В то же время, модель, комбинирующая разреживание и оптимизацию для него, обучается быстрее прочих, в различных случаях выигрыш составляет до 30%. Очевидно, что он становится тем более явным, чем больше тем и проходов по документу используется в модели, поскольку их рост усложняет подсчёт E-шага (1.9).

Синхронный онлайновый алгоритм. В экспериментах с `Online ARTM` используется всего один проход по коллекции, количество проходов по коллекции также изменяется по сетке значений 1, 5, 10, 15. Слияние матриц n_{wt} с последующими регуляризацией и нормализацией производится каждые $\eta = 32$ пакетов. Значения коэффициента разреживающей регуляризации τ_1 выбраны из тех же соображений, что и для оффлайнового алгоритма, и для моделей со 100, 500,

1000 и 2000 тем равны 0.1, 0.03, 0.015 и 0.01 соответственно.

Таблица 3.4 демонстрирует результаты сравнения. Как и в случае оффлайнового алгоритма, оптимизация не даёт никаких преимуществ в случае моделей с небольшим числом тем или алгоритмов обучения, запущенных с одним проходом по документу. В прочих ситуациях, он, как и в случае оффлайнового алгоритма, обеспечивает модели Y/Y существенное (до 30%) уменьшение времени обучения по сравнению с конкурентами. Дополнительным преимуществом оптимизации в случае онлайнового алгоритма является экономия оперативной памяти, которой удаётся достичь за счёт сжатия разреженных промежуточных версий матрицы счётчиков n_{wt}. Благодаря этому свойству, разреженная модель с оптимизацией покрывает издержки, возникающие из-за наличия матрицы r_{wt}, и потребляет примерно такой же пиковый объём оперативной памяти, что и базовая модель N/N.

Алгоритм DetAsync. Все параметры запуская детерминированного асинхронного алгоритма DetAsync полностью совпадают с параметрами синхронного. Результаты сравнительных экспериментов показаны в таблице 3.5. Во всех случаях разреженная модель с включенной оптимизацией обеспечивает выигрыш во времени до 25% по сравнению базовой модели N/N. Асинхронный алгоритм является более требовательным по памяти, чем синхронный, в силу наличия дополнительной версии матрицы n_{wt} для обновлений с запаздыванием. Это позволяет предлагаемой оптимизации ощутимо сократить потребление памяти при обучении разреженной модели (до 23% экономии в различных случаях по сравнению с базовой плотной).

3.7. Основные выводы

Описаны и реализованы различные алгоритмы обучения тематических моделей ARTM, обладающие различными свойствами: работающие в оффлайно-

Темы	Модель	Память	Вр. (1)	Вр. (5)	Вр. (10)	Вр. (15)
100	N/N	1170	65	80	120	150
	N/Y	1250	65	85	125	150
	Y/Y	**1110**	**60**	**75**	**110**	**125**
500	N/N	2700	200	340	515	660
	N/Y	2980	190	340	535	670
	Y/Y	**2270**	**185**	**290**	**450**	**500**
1000	N/N	4640	520	730	1095	1435
	N/Y	5230	555	760	1115	1475
	Y/Y	**3910**	**500**	**600**	**900**	**1065**
2000	N/N	9110	860	1400	2140	2730
	N/Y	9760	865	1415	2120	2720
	Y/Y	**7230**	**780**	**1120**	**1670**	**2010**

Таблица 3.5. Оптимизация для разреженных моделей, алгоритм DetAsync. Столбцы: Темы — число тем в модели, Модель — тип модели, Память — пиковое потребление памяти (в мегабайтах), Вр. (X) — время обучения (в секундах) при использовании X проходов по документу. Лучшие значения выделены жирным.

вом или онлайновом режиме, синхронные или асинхронные, детерминированные или нет, использующие свойство разреженности модели или нет. Предложены и реализованы асинхронный детерминированный алгоритм DetAsync и оптимизация хранения и обработки разреженных моделей, подходящая для всех режимов обучения в библиотеке BigARTM. В серии экспериментов получен описанный ниже набор выводов.

1. Библиотека BigARTM содержит набор различных алгоритмов, позволяющих в разных режимах эффективно обучать регуляризованные тематические модели на одном вычислительном узле.

2. Детерминированный асинхронный алгоритм DetAsync позволяет достичь более низкого уровня перплексии за выделенный отрезок времени по сравнению с оффлайновым алгоритмом, синхронным онлайновым алгоритмом и алгоритмом Async.

3. И синхронный онлайновый алгоритм, и DetAsync существенно превосхо-

дят ближайших конкурентов (библиотеки Vowpal Wabbit LDA и Gensim) с точки зрения производительности и масштабируемости, оставаясь на одном уровне по качеству с точки зрения итоговой перплексии.

4. Оптимизированные хранение и обработка разреженных матриц в случае разреженных моделей позволяют ускорить обучение всех алгоритмов и уменьшить потребление оперативной памяти онлайновыми алгоритмами.

Результаты данной главы опубликованы в работах [17, 20, 21].

Глава 4

Произвольная функция потерь, быстрый E-шаг

4.1. Произвольная функция потерь, быстрый E-шаг

Произвольная функция потерь. Функционал (1.2), используемый в задаче ARTM (1.8), (1.3), (1.4), использует логарифмическую функцию потерь, которая приводит к уже известным формулам EM-алгоритма (1.9)–(1.11). Такой выбор, являющийся следствием логарифмирования правдоподобия, является стандартным, но не единственным. В теоретической части работы [21] предлагается обобщение оптимизационной задачи ARTM, заключающееся в замене логарифмической функции потерь $\ln p(w \mid d)$ на произвольную гладкую возрастающую функцию ℓ:

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \ell\Big(\sum_{t \in T} \phi_{wt}\theta_{td}\Big) + R(\Phi, \Theta) \;\to\; \max_{\Phi,\Theta}. \tag{4.1}$$

Формулируется и доказывается следующая теорема:

Теорема 1. *Локальный максимум* (Φ, Θ) *оптимизационной задачи* (4.1), (1.3) *и* (1.4) *с дифференцируемой возрастающей функцией потерь* ℓ *и дифференцируемым регуляризатором* R *удовлетворяет системе из уравнения M-шага* (1.10)–(1.11) *и уравнения E-шага*

$$p_{tdw} = \phi_{wt}\theta_{td}\ell'\Big(\sum_{t \in T} \phi_{wt}\theta_{td}\Big).$$

Таким образом, классическая система (1.9)–(1.11) отличается от полученной только формулой E-шага и так же является результатом применения теоремы 1 в случае логарифмической функции потерь $\ell(p) = \ln p$.

Быстрый E-шаг. Заметим, что изменение функции потерь с логарифмической на линейную $\ell(p) = p$ приводит к наиболее простой форме E-шага. Вместо

максимизации лог-правдоподобия происходит максимизация взвешенной суммы скалярных произведений между модельными распределениями термов $p(w|d)$ и соответствующими им эмпирическими оценками $\hat{p}(w|d) = \frac{n_{dw}}{n_d}$:

$$\sum_{d \in D} n_d \big\langle \hat{p}(w|d), p(w|d) \big\rangle + R(\Phi, \Theta) \;\to\; \max_{\Phi,\Theta}.$$

Подобный принцип оптимизации кажется столь же разумным, как и классическая максимизация правдоподобия. Более того, в этом случае формула вычисления вспомогательных переменных на E-шаге примет вид $p_{tdw} = \phi_{wt}\theta_{td}$. Это означает возможность избавиться от вычисления нормировочной константы в (1.9), на которое, как было неоднократно замечено в предыдущей главе, тратится значительная часть времени при обработка каждого уникального терма каждого документа.

Описанная модификация потенциально способна обеспечить существенное ускорение EM-алгоритма, поэтому она получила название *быстрого E-шага*.

Стратегии использования быстрого E-шага. Заметим, что оптимизация функционала (4.1) с $\ell(p) = p$ может ухудшить сходимость модели, в силу того, что перплексия (3.1), отражающая скорость и качество сходимости, продолжает быть функцией лог-правдоподобия. Рост значения перплексии не является неотвратимым: наличие регуляризаторов в функционале (1.8) так же уводит алгоритм от прямой оптимизации правдоподобия, однако правильно подобранный набор регуляризаторов позволяет минимизировать потери, существенно улучшив при этом другие показатели модели [31].

Таким образом, следует рассмотреть различные комбинации использования обычных и быстрых E-шагов с целью добиться сохранения или увеличения качества модели при росте скорости работы обучающего алгоритма.

Попеременное использование двух видов E-шагов возможно как на уровне проходов по коллекции, так и на уровне проходов по документу. В первом случае часть проходов по коллекции производится полностью с использованием

обычных E-шагов, а часть — с использованием быстрых шагов. Во втором разделение происходит на уровне обработки каждого документа. Назовём первый вид стратегий *внешними*, а второй — *внутренними*. С точностью до смысла слова «проход» для обоих видов предлагается следующий общий набор из пяти стратегий:

- FULL (F): крайний случай, все проходы производятся с использованием обычного E-шага — это базовый вариант, с которым соревнуются прочие стратегии;
- NONE (N): другой крайний случай, в котором на всех проходах используется только быстрый E-шаг, и который потенциально способен ощутимо ухудшить сходимость модели;
- MIXED (M): на каждом проходе быстрые и обычные E-шаги чередуются один за другим, последний проход всегда производится с обычным E-шагом;
- HALF (H): первая половина проходов быстрая, вторая — обычная;
- LAST (L): первые 80% быстрые, оставшиеся — обычные.

Далее проводится сравнение предлагаемых стратегий.

4.2. Экспериментальные результаты

Помимо итогового значения перплексии и времени обучения в экспериментах этого раздела используется ещё одна автоматически измеряемая метрика качества модели — когерентность. В данной работе когерентность темы t определяется как среднее по парам термов значение положительной по-точечной взаимной информации (positive pointwise mutual information, PPMI):

$$\mathcal{C}_t(\Phi) = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \mathrm{PPMI}(w_i, w_j), \tag{4.2}$$

где w_i — это i-й терм в списке из k наиболее вероятных термов в распределении темы t, $\mathrm{PPMI}(u,v) = \max\{0, \ln \frac{|D|N_{uv}}{N_u N_v}\}$, N_{uv} — это число документов, которые содержат оба терма u и v в окне заданной длины (или во всём документе в случае «мешка слов»), а N_u означает число документов, содержащих терм u.

Существует множество подходов к определению когерентности, но все они, как и формула выше, используют информацию о попарной встречаемости термов. Когерентность известна своей высокой корреляцией с человеческим понятием интерпретируемости [66, 67], поэтому среднее её значение по темам часто используется для получения оценки интерпретируемости без необходимости обращения к экспертам.

Условия экспериментов полностью совпадают с описанными в разделе 3.6: обучение всех моделей производилось в 8 рабочих потоков на сервере с процессором Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz и 128Гб ОЗУ, в качестве обучающих данных для оффлайнового алгоритма используются 200 тысяч статей англоязычной Википедии, для онлайновых — 1 миллион статей из этой же коллекции с общим словарём из 100 тысяч термов. Модели строятся со 100, 500, 1000 и 2000 темами.

Для подсчёта когерентности используются счётчики со-встречаемости термов, собранные по указанной выше коллекции для онлайновых алгоритмов из 1 миллиона статей. Счётчик для пары термов увеличивался на единицу за каждый документ, в котором оба этих терма встретились хотя бы один раз. Счётчики со значением меньше 100 были обнулены и исключены из словаря со-встречаемостей. Число k наиболее вероятных слов во всех экспериментах принято равным 10.

Помимо сравнения друг с другом, предлагаемые в этой главе стратегии комбинирования E-шагов также сравниваются с обычной (т.е. FULL) моделью, в которой используется регуляризатор равномерного разреживания (SPARSE, S), и обычной моделью, в которой используется регуляризатор декорреляции тем (DECOR, D). Для удобства в дальнейшем обозначим модель без регуляризации

Темы	Модель	Перплексия [ЗМВ]	Когерентность [ЗМВ]	Время
100	N/F	3729 [3769]	0.086 [0.083]	720
	N/N	5333 [5333]	0.104 [0.104]	510
	N/M	4038 [4357]	0.088 [0.098]	630
	N/H	3747 [3774]	0.084 [0.083]	620
	N/L	3827 [3874]	0.087 [0.083]	555
	D/F	**3741 [3795]**	**0.092 [0.090]**	**760**
500	N/F	2076 [2123]	0.092 [0.093]	2850
	N/N	3759 [3759]	0.098 [0.098]	1740
	N/M	2442 [2811]	0.103 [0.103]	2375
	N/H	**2077 [2121]**	**0.093 [0.097]**	**2300**
	N/L	2169 [2264]	0.095 [0.096]	1945
	D/F	2101 [2169]	0.094 [0.091]	3570
1000	N/F	1483 [1546]	0.078 [0.078]	5430
	N/N	3122 [3122]	0.088 [0.088]	3295
	N/M	1845 [2198]	0.092 [0.094]	4465
	N/H	**1474 [1498]**	**0.084 [0.084]**	**4224**
	N/L	1561 [1612]	0.084 [0.085]	3672
	D/F	1571 [1619]	0.077 [0.077]	6649
2000	N/F	998 [1025]	0.057 [0.059]	11405
	N/N	2574 [2574]	0.069 [0.069]	7775
	N/M	1298 [1618]	0.071 [0.074]	9975
	N/H	**977 [1006]**	**0.063 [0.063]**	**9380**
	N/L	1049 [1129]	0.064 [0.065]	8570
	D/F	1255 [1524]	0.056 [0.057]	28500

Таблица 4.1. Сравнение стратегий комбинирования E-шагов и разреживания, оффлайновый алгоритм. Столбцы: Темы — число тем, Модель — тип модели/стратегия комбинирования E-шагов, Перплексия — итоговые значения перплексии, Когерентность — итоговые значения средней когерентности, Время — время завершения всех проходов по коллекции (в секундах). В ячейках перплексии и когерентности указаны финальные значения метрик и значения на момент завершения работы самым быстрым из алгоритмов (ЗМВ — за минимальное время). Лучшие значения выделены жирным.

как NONE (N).

Оффлайновый алгоритм. Для оффлайнового алгоритма исследуются только внешние стратегии комбинирования шагов, включать внутренние не имеет

смысла в силу наличия возможности многократно обрабатывать один и тот же документ различными способами на разных проходах по коллекции.

Для того, чтобы провести честное сравнение с моделью, обучаемой с регуляризатором декорреляции тем, проведён предварительный отбор значения коэффициента τ_3 из формулы (1.14) этого регуляризатора. Наилучший результат был получен при значениях 1×10^6, 2×10^5, 1×10^5 и 5×10^4 для 100, 500, 1000 и 2000 тем соответственно. Эти значения используются в дальнейшем.

Измеряются два вида значений перплексии и средней когерентности: одна пара значений вычисляется на момент завершения 40 проходов по коллекции, вторая — на момент завершения фиксированного отрезка времени. Для каждого значения числа тем величина этого отрезка определяется временем работы самого быстрого из алгоритмов.

Таблица 4.1 демонстрирует следующие результаты:

- декоррелирование тем без использования быстрых E-шагов является лучшим вариантом в ситуации модели со 100 темами;
- в прочих случаях этот регуляризатор оказывается в проигрышном положении, поскольку его вычислительная сложность растёт квадратично от числа тем;
- стратегия NONE ожидаемо является наиболее быстрой и, в дополнение к этому, улучшает значение когерентности, однако приводит к существенному ухудшению перплексии;
- стратегия HALF выглядит оптимальным вариантом как в случае фиксированного числа проходов по коллекции (до 20% экономии времени обучения по сравнению с FULL), так и в случае обработки в пределах выделенного отрезка времени; во обеих ситуациях такой подход позволяет сохранить или даже улучшить значение перплексии и обеспечивает до 10% роста когерентности.

Темы	Модель	η	Перплексия	Когерентность	Время
100	N/F	32	8983	0.091	180
	N/N	32	10323	0.074	100
	N/M	32	9138	0.084	135
	N/H	32	9033	0.081	135
	N/L	32	9721	0.077	115
	S/F	32	8891	0.066	155
	D/F	**32**	**8862**	**0.128**	**190**
	N/H	24	8104	0.077	155
	N/H	16	7031	0.084	190
	N/H	8	6030	0.095	285
500	N/F	32	6630	0.080	710
	N/N	32	9427	0.047	365
	N/M	32	7342	0.071	540
	N/H	32	7030	0.076	545
	N/L	32	8314	0.056	440
	S/F	32	6563	0.056	525
	D/F	32	6401	0.077	820
	N/H	24	6189	0.075	620
	N/H	**16**	**5265**	**0.080**	**750**
	N/H	8	4442	0.071	1120

Таблица 4.2. Сравнение стратегий комбинирования E-шагов и разреживания, синхронный онлайновый алгоритм, 100 и 500 тем. Столбцы: Темы — число тем, Модель — тип модели/стратегия комбинирования E-шагов, η — количество обрабатываемых между обновлениями Φ пакетов, Перплексия — итоговое значение перплексии, Когерентность — итоговое среднее значение когерентности, Время — время завершения прохода по коллекции (в секундах). Лучшие значения выделены жирным.

Стоит заметить, что эксперименты с равномерным разреживанием в оффлайновом алгоритме не отражены в этой работе, поскольку ни в одном случае не показали хорошего результата.

Синхронный онлайновый алгоритм. Онлайновый алгоритм запускается с одним проходом по коллекции, поэтому для улучшения сходимости перплексии количество проходов по документу увеличивается по сравнению с оффлайновым, и устанавливается равным 10 для всех экспериментов. Стандартная ча-

Темы	Модель	η	Перплексия	Когерентность	Время
1000	N/F	32	5871	0.063	1435
	N/N	32	9113	0.044	835
	N/M	32	6712	0.055	1145
	N/H	32	6315	0.057	1120
	N/L	32	7789	0.050	955
	S/F	32	5663	0.051	1065
	D/F	32	5558	0.053	2110
	N/H	24	5519	0.057	1215
	N/H	**16**	**4652**	**0.060**	**1475**
	N/H	8	3890	0.058	2175
2000	N/F	32	5144	0.045	2995
	N/N	32	8879	0.038	1630
	N/M	32	6145	0.046	2315
	N/H	32	5694	0.046	2320
	N/L	32	7324	0.042	1911
	S/F	32	4920	0.040	2145
	D/F	32	7211	0.031	12780
	N/H	24	4962	0.045	2595
	N/H	**16**	**4157**	**0.044**	**3155**
	N/H	8	3428	0.039	4589

Таблица 4.3. Сравнение стратегий комбинирования E-шагов и разреживания, синхронный онлайновый алгоритм, 1000 и 2000 тем. Столбцы: Темы — число тем, Модель — тип модели/стратегия комбинирования E-шагов, η — количество обрабатываемых между обновлениями Φ пакетов, Перплексия — итоговое значение перплексии, Когерентность — итоговое среднее значение когерентности, Время — время завершения прохода по коллекции (в секундах). Лучшие значения выделены жирным.

стота обновления матрицы Φ — раз в $\eta = 32$ пакетов.

Как и в случае с оффлайновым алгоритмом, хочется не только получить ускорение за счёт использования стратегии смешивания видов E-шагов, но также иметь возможность попробовать использовать этот выигрыш в скорости для улучшения качества модели. Возможность досрочно завершить обработку, как в случае с оффлайновым подходом, отсутствует (требуется хотя бы один раз обработать всю текстовую коллекцию). Поэтому алгоритм, работающий быстрее, можно запустить с повышенной частотой обновлений (каждые 24, 16 и 8

пакетов), что положительно сказывается на сходимости модели.

Для онлайнового алгоритма, как и для оффлайнового, значения коэффициентов регуляризации подбирались в предварительном эксперименте перебором по сетке значений. Набор значений 1×10^5, 5×10^3, 2×10^3 и 1×10^3 для 100, 500, 1000 и 2000 тем соответственно оказался наилучшим. Та же процедура, проделанная с регуляризатором равномерного разреживания, позволила выбрать для указанного набора количеств тем значения 0.1, 0.03, 0.015 и 0.01 соответственно.

Как для синхронного онлайнового алгоритма, так и для DetAsync в этой работе представлены результаты экспериментов, относящиеся к группе внутренних стратегий смешивания E-шагов. Внешние стратегии в случае онлайнового подхода заключаются в том, что часть документов обрабатывается только с помощью быстрых шагов, а часть — только с помощью обычных. Экспериментально выявлено, что все попытки такого обучения моделей приводят к существенному ухудшению итогового качества.

Результаты экспериментов с онлайновым синхронным алгоритмом и группой внутренних стратегий демонстрируются в таблицах 4.2 и 4.3. В соответствии с ними можно сделать следующие выводы:

- декорреляция тем вновь показала себя лучшим решением для модели со 100 темами и плохим вариантом для больших моделей;
- использование стратегий смешивания при сохранении частоты обновления модели приводит к сокращению времени обучения на 25-50%, но также сильно ухудшает и перплексию, и когерентность;
- запуск обучения с наиболее перспективной стратегией HALF и более частыми обновлениями Φ позволяет получить значительное (до 23%) улучшение значения перплексии с небольшим ухудшением когерентности или даже без него.

Попытки использовать выигрыш во времени, получаемый разреженными моделями за счёт описанной в главе 3.5 оптимизации, не увенчался успехом — при повышение частоты обновлений в синхронном алгоритме разреживание приводит к неприемлемо сильному ухудшению когерентности.

Алгоритм DetAsync. Параметры экспериментов для асинхронного онлайнового алгоритма DetAsync идентичны используемым для синхронного. Для лучшего понимания полученных результатов необходимо вспомнить о том, что DetAsync производит одновременно обработку новой порции документов и уточнение модели с помощью счётчиков n_{wt}, полученных при обработке предыдущей порции. Это факт значительно усиливает эффект любого ускорения E-шага. Кроме того, это позволяет алгоритму игнорировать затраты на регуляризацию в том случае, если время, которое тратится на неё, не превышает времени обработки рабочими потоками порции из η пакетов.

Таблицы 4.4 и 4.5 позволяют сделать следующие выводы об использовании быстрых E-шагов в асинхронном алгоритме:

- Декорреляция тем остаётся наиболее выгодным решением при обучении со 100 и 500 темами, но при дальнейшем росте числа тем и скорость обучения, и качество итоговых моделей ощутимо падают;
- быстрый E-шаг, как и ожидалось, даёт ещё большее ускорение, чем в случае синхронного алгоритма (до двух раз в случае стратегии NONE), но качество моделей при сохранении частоты обновления Φ заметно ухудшается по сравнению с базовой моделью;
- стратегия HALF с более частыми обновлениями Φ является наилучшим решением для моделей с 1000 и 2000 тем: при максимальных потерях когерентности до 10% она позволяет получить более чем двукратное уменьшение перплексии, во всех случаях оставаясь быстрее базового варианта FULL на 10–30%;

- в отличие от синхронного алгоритма, в DetAsync равномерное разрежи-вание в ряде случаев позволяет получить хорошие показатели скорости и перплексии, сопоставимые с показателями стратегии HALF и даже превосходящие их, однако более частые обновления матрицы Φ, необходимые для улучшения сходимости модели, приводят к сильному (до 80%) ухудшению когерентности.

4.3. Основные выводы

Описана теория произвольной функции потерь в EM-алгоритме при обучении тематических моделей ARTM. Предложено использование частного случая линейной функции потерь для устранения необходимости нормировки на E-шаге алгоритма (быстрый E-шаг). Выдвинут набор стратегий комбинирования обычных и быстрых E-шагов. Экспериментально определены стратегии для оффлайнового и онлайновых EM-алгоритмов, позволяющие в большинстве случае добиться наилучшего результата по совокупности критериев скорости работы, правдоподобия и интерпретируемости. В ряде экспериментов получен следующий набор основных выводов, связанных с обучением тематических моделей на крупных текстовых коллекциях.

1. Использование быстрых E-шагов лишено смысла при обучении моделей с небольшим (менее 500) числом тем, лучшего результата можно добиться с помощью хорошо настроенной декорреляции тем.

2. В процессе обучения более крупных моделей регуляризатор декорреляции тем становится слишком медленным и неподходящим для использования.

3. При обучении моделей с 500 и более темами с помощью оффлайнового алгоритма наилучшим из опробованных решений оказалась внешняя стратегия HALF с уменьшенным количеством проходов по коллекции.

4. В случае онлайновых алгоритмов лучшим выбором является внутренняя стратегия HALF с использованием более частых обновлений матрицы Φ (в 1.5–2 раза).

5. В синхронных алгоритмах равномерное разреживание работает быстро (за счёт оптимизации для разреженных моделей в BigARTM), но даёт недостаточно качественные результаты по совокупности критериев.

6. В асинхронном алгоритме равномерное разреживание даёт ещё более сильное ускорение, за счёт которого появляется возможность максимально быстро получить модель с наилучшей перплексией, однако проблема ухудшения когерентности сохраняется и в этом случае.

7. Для оффлайнового и онлайновых алгоритмов использование комбинаций обычных и быстрых E-шагов даёт ощутимо разные результаты с точки зрения качества модели: в первом случае происходит некоторое улучшение когерентности при сохранении перплексии, во втором — значительное улучшение перплексии при сохранении или небольшом падении когерентности. Предположительно это связано не только со свойствами самих алгоритмов, но и типом стратегий (внешние или внутренние), а также со способом использования ускорения для повышения качества (сокращение числа проходов по коллекции или повышение частоты обновлений Φ).

Результаты данной главы опубликованы в статье [21].

Темы	Модель	η	Перплексия	Когерентность	Время
100	N/F	32	13391	0.070	115
	N/N	32	13856	0.054	60
	N/M	32	13262	0.066	90
	N/H	32	13308	0.064	90
	N/L	32	13545	0.061	70
	S/F	32	12258	0.065	100
	D/F	32	13619	0.096	115
	N/H	24	11374	0.067	90
	N/H	16	9373	0.077	95
	N/H	8	7197	0.083	125
	S/F	24	10638	0.063	100
	S/F	16	9112	0.057	100
	S/F	8	7554	0.039	130
	D/F	24	11630	0.100	115
	D/F	**16**	**9614**	**0.134**	**120**
	D/F	8	7586	0.105	170
500	N/F	32	11034	0.070	490
	N/N	32	12716	0.041	185
	N/M	32	11385	0.059	340
	N/H	32	11273	0.060	340
	N/L	32	12024	0.047	245
	S/F	32	9762	0.057	365
	D/F	**32**	**11007**	**0.093**	**495**
	N/H	24	9265	0.071	340
	N/H	16	7343	0.075	355
	N/H	**8**	**5400**	**0.076**	**455**
	S/F	24	8166	0.054	345
	S/F	16	6683	0.050	345
	S/F	8	5377	0.036	415
	D/F	**24**	**8892**	**0.079**	**490**
	D/F	**16**	**6908**	**0.079**	**540**
	D/F	8	5137	0.063	810

Таблица 4.4. Сравнение стратегий комбинирования E-шагов и разреживания, алгоритм DetAsync, 100 и 500 тем. Столбцы: Темы — число тем, Модель — тип модели/стратегия комбинирования E-шагов, η — количество обрабатываемых между обновлениями Φ пакетов, Перплексия — итоговое значение перплексии, Когерентность — итоговое среднее значение когерентности, Время — время завершения прохода по коллекции (в секундах). Лучшие значения выделены жирным.

Темы	Модель	η	Перплексия	Когерентность	Время
1000	N/F	32	10154	0.062	1010
	N/N	32	12344	0.039	380
	N/M	32	10652	0.054	710
	N/H	32	10456	0.054	710
	N/L	32	11452	0.042	520
	S/F	32	8796	0.054	750
	D/F	32	9895	0.061	1040
	N/H	24	8472	0.057	700
	N/H	**16**	**6599**	**0.056**	**710**
	N/H	**8**	**4788**	**0.058**	**900**
	S/F	24	7161	0.054	710
	S/F	16	5716	0.047	680
	S/F	8	4421	0.032	785
	D/F	24	7856	0.057	1175
	D/F	16	5935	0.052	1675
	D/F	8	4346	0.044	3280
2000	N/F	32	9267	0.049	2150
	N/N	32	12024	0.033	1050
	N/M	32	9969	0.045	1590
	N/H	32	9707	0.045	1540
	N/L	32	10902	0.037	1225
	S/F	**32**	**7821**	**0.047**	**1560**
	D/F	32	8654	0.040	5435
	N/H	**24**	**7748**	**0.046**	**1590**
	N/H	**16**	**5976**	**0.044**	**1615**
	N/H	8	4274	0.043	1890
	S/F	**24**	**6236**	**0.045**	**1470**
	S/F	16	4855	0.038	1390
	S/F	8	3921	0.025	1550

Таблица 4.5. Сравнение стратегий комбинирования E-шагов и разреживания, алгоритм DetAsync, 1000 и 2000 тем. Столбцы: Темы — число тем, Модель — тип модели/стратегия комбинирования E-шагов, η — количество обрабатываемых между обновлениями Ф пакетов, Перплексия — итоговое значение перплексии, Когерентность — итоговое среднее значение когерентности, Время — время завершения прохода по коллекции (в секундах). Лучшие значения выделены жирным.

Глава 5

Мультимодальные и транзакционные модели

В предшествующих главах рассматриваются различные подходы к повышению эффективности EM-алгоритма для обучения аддитивно регуляризованных моделей. Заключительная глава этой работы посвящена реализации и использованию расширений базовой модели ARTM в рамках библиотеки BigARTM.

Сперва рассматривается обобщение ARTM, предназначенное для обработки мультимодальных текстовых данных (M-ARTM), производится доработка BigARTM для обучения такого рода моделей. Предлагается комбинация регуляризаторов, повышающих качество решения задачи поиска информации в текстовой коллекции с помощью тематического моделирования. Производится экспериментальное обоснование результативности предлагаемой модели M-ARTM на реальных данных.

Затем вводится теоретическое описание транзакционной аддитивно регуляризованной тематической модели T-ARTM, которая представляет собой расширение модели M-ARTM для обработки данных, состоящих из наборов связанных друг с другом термов различных модальностей. Производится разработка системы для обучения транзакционных моделей в библиотеке BigARTM, в экспериментах на синтетических данных демонстрируется превосходство T-ARTM над M-ARTM в задаче восстановления параметров модели по данным транзакционной природы.

5.1. Мультимодальные модели M-ARTM

Предположим, что текстовый документ может содержать не только обычные термы-слова, но и термы других *модальностей*. Модальность представляет собой тип термов и характеризуется словарём термов, относящихся к ней. Модальностей может быть любое конечное число M, требуется, чтобы множества

термов различных модальностей $W^m,\ m = 1, \ldots, M$ не пересекались друг с другом.

Примерами модальностей, отличными от обычных слов документа, могут быть имена авторов, метки категорий и классов, метки времени, ссылки на документ или из него, пользователи, скачавшие документ и т.д.

Как и в главе 1, текстовая коллекция рассматривается как множество троек d_i, w_i, t_i, сэмплированных из дискретного вероятностного пространства $W \times D \times T$, но теперь $W = W^1 \sqcap \cdots \sqcap W^M$ представляет собой объединение непересекающихся термов всех представленных в коллекции модальностей.

В соответствии с идеями моделей Correspondence LDA [68] и Dependency LDA [6], для каждой модальности $W^m, m = 1, \ldots, M$ вводится своя тематическая модель $p(w|d)$:

$$p(w|d) = \sum_{t \in T} p(w|t)p(t|d) = \sum_{t \in T} \phi_{wt}\theta_{td}, w \in W^m.$$

Матрицы вероятностных распределений $\Phi^m = (\phi_{wt})_{W^m \times T}$, соединённые вертикально, образуют матрицу параметров всей тематической модели Φ.

Обучение параметров Φ^m, Θ производится путём максимизации лог-правдоподобия для каждой из модальностей:

$$\ln p(X \mid \Phi^m, \Theta) = \sum_{d \in D} \sum_{w \in W^m} n_{dw} \ln \sum_{t \in T} \phi_{wt}\theta_{td} \ \to \ \max_{\Phi^m, \Theta}.$$

Стоит отметить, что распределения тем в документах Θ являются общими для всех модальностей.

Сформулируем задачу M-ARTM [14] максимизации с ограничениями взвешенной суммы функционалов лог-правдоподобия по всем модальностям и набора регуляризаторов:

$$\sum_{m=1}^{M} \tau_m \ln p(X \mid \Phi^m, \Theta) + R(\Phi, \Theta) \ \to \ \max_{\Phi, \Theta}; \tag{5.1}$$

$$\sum_{w \in W^m} \phi_{wt} = 1,\ \phi_{wt} \geq 0,\ m = 1, \ldots, M; \tag{5.2}$$

$$\sum_{t\in T}\theta_{td} = 1,\ \theta_{td} \geq 0. \tag{5.3}$$

Коэффициенты регуляризации τ_m (функционалы дополнительных модальностей $p(X \mid \Phi^m, \Theta)$ по своей сути тоже являются регуляризаторами) отвечают за степень важности каждой из модальностей при обучении всей модели. В [14] доказывается, что решение задачи (5.1)–(5.3) удовлетворяет следующей системе уравнений:

$$p_{tdw} = \underset{t\in T}{\operatorname{norm}}\big(\phi_{wt}\theta_{td}\big); \tag{5.4}$$

$$\phi_{wt} = \underset{w\in W^m}{\operatorname{norm}}\left(n_{wt} + \phi_{wt}\frac{\partial R}{\partial \phi_{wt}}\right); \qquad n_{wt} = \sum_{d\in D}\tau_{m(w)}n_{dw}p_{tdw}; \tag{5.5}$$

$$\theta_{td} = \underset{t\in T}{\operatorname{norm}}\left(n_{td} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}\right); \qquad n_{td} = \sum_{w\in d}\tau_{m(w)}n_{dw}p_{tdw}; \tag{5.6}$$

где $\tau_{m(w)}$ является коэффициентом модальности терма w. Легко заметить, что в случае одной модальности получается EM-алгоритм для обычной модели ARTM (1.8)–(5.6).

Многие байесовские тематические модели на размеченных документах могут рассматриваться в качестве частного случая M-ARTM с различными видами модальностей. Тематическая модель содержимого документа и гиперссылок [69] вводит модальность ссылок между документами. В модели CI-LDA [70] есть модальность именованных сущностей, упомянутых в документе. В Tag-LDA [71] теги документов используются в качестве отдельного типа термов. Модели LDA-JS и LDA-post [72] содержат модальность процитированных в документе публикаций. В [6] предлагается тематическая модель Dependency LDA, которая использует модальность меток документов для решения задачи классификации. Модели MultiLingual LDA [11] и PolyLingual Topic Model [12] вводят L модальностей для L различных языков, параллельные документы всегда имеют одно и то же распределение тем. BiLingual LDA [13] также представляет собой

многоязычную тематическую модель, но только с двумя модальностями.

5.2. Тематические модели для поиска специфической информации с частичным обучением

Общий подход. Рассматривается задача поиска информации о специфической предметной области в неструктурированной текстовой коллекции. Пусть дан набор ключевых термов $Q \subset W$, который может быть слишком большим для традиционных поисковых систем. Для извлечения тем, связанных с нужной предметной областью используется тематическая модель частичного обучения с заданной априорной информацией.

Похожая техника предлагалась ранее в задаче кластеризации новостей [73], поиска тем, связанных со здоровьем, в социальных медиа [74] и задаче поиска обсуждений этнических вопросов в постах блогов [75, 76].

Во всех этих исследованиях для каждой теме задавался предопределённый набор ключевых слов, часто очень маленький, т.е. категория новости или название этничности. Это означает, что информация о точном числе тем и их примерном содержимом известна заранее.

Модель Interval Semi-supervised LDA (ISLDA) позволяет присвоить каждому ключевому слову из набора Q больше, чем одну тему, однако, довольно сложно определить, сколько реально тем соответствует каждому из слов. И если исследователь не задаст ключевые слова для каждой из тем, обучение модели невозможно.

Например, в [75, 76], где цель исследования была схожа с описываемой в данной работе, ISLDA использовался для поиска этнического материала, но, поскольку этнонимы (составляющие Q в этом случае) были соотнесены с различными темами, появление мульти-этничных тем было невозможно.

Предлагаемое решение описанной проблемы состоит в задании общей для всех целевых тем лексической априорной информации на основе Q. Модели

предоставляется самостоятельно определять их распределения внутри тем и их комбинаций по темам.

Частичное обучение в модели ARTM. Прежде всего необходимо произвести разбиение всех тем T на два подмножества: *предметные* темы S и *фоновые* темы B. Темы из первого подмножества являются целевыми и должны в результате моделирования получиться связанными с интересующей предметной областью. Подмножество B нужно для моделирования всех тематик текстовой коллекции, не связанных с целевой предметной областью. В случае моделирования без фоновых тем получается ситуация, при которой целевая задача и решаемая алгоритмом задача оптимизации вступают в противоречие: алгоритм пытается строить модель, хорошо описывающую все данные в коллекции D, в то время как её пытаются заставить строить темы только по данным из одной предметной области, которая может быть очень слабо представлена в коллекции. Фоновые темы дают модели большую степень свободы, позволяя разместить в себе ту общую информацию о коллекции, появление которой в предметных темах является нежелательным.

Подобный подход уже использовался в [77]. Отличие от указанной работы заключается в том, что используется не одна, а множество фоновых тем, для того, чтобы как можно лучше очистить предметные темы от нерелевантных термов и повысить общее качество модели.

Наиболее простым и естественным способом включения в модель априорной информации в виде набора Q термов искомой предметной области является использование регуляризаторов сглаживания (1.12) и разреживания (1.13) с распределением β, содержащим для каждого трема из словаря W значения следующего вида:

$$\beta_w = \frac{1}{|Q|}[w \in Q].$$

Регуляризатор сглаживания, применяемый только к набору тем S, поощ-

ряет появление релевантных термов $w \in Q$ в предметных темах. Как следствие, значительная часть вероятностной массы этих тем будет сосредоточена в релевантных термах и термах, которые отсутствуют в словаре Q, но часто встречаются рядом с релевантными, совместно с ними образовывая описание того, что обсуждается в документах и при этом связано с целевой предметной областью.

Регуляризатор разреживания, действующий на подмножестве фоновых тем, решает обратную задачу: вычитая поправки из счётчиков n_{wt}, соответствующих термам $w \in Q$, он поощряет формирование в B тем, связанных со всеми прочими предметными областями, которые встречаются в D, кроме целевой.

Таким образом, первые предлагаемые регуляризаторы имеют следующий вид:

$$R(\Phi) = \tau_1' \sum_{t \in S} \sum_{w \in Q} \ln \phi_{wt} - \tau_2' \sum_{t \in B} \sum_{w \in Q} \ln \phi_{wt}.$$

Стоит отметить, что распределение β_w нарочно исключено из формул: его роль сведена к булевому массиву, определяющему то, какие термы в теме нужно регуляризовывать, а какие — нет. Степень воздействия регуляризаторов полностью определяется абсолютными значениями их коэффициентов τ_1' и τ_2'. Это позволяет избежать проблем с настройкой гиперпараметров модели в случае изменения мощности множества Q.

Как уже отмечено выше, задача решается в предположении незначительности доли релевантного содержимого в коллекции. Требуется описать тематическую структуру релевантных документов большим числом некрупных качественных тем. Документы, не относящиеся к релевантным, должны описываться темами из фонового подмножества B.

Эти требования формализуются в терминах сглаживающего регуляризатора матрицы Θ, работающего только с фоновыми темами, и регуляризатора, который равномерно разреживает предметные темы в этой же матрице:

$$R(\Theta) = \tau_3' \sum_{d \in D} \sum_{t \in B} \ln \theta_{td} - \tau_4' \sum_{d \in D} \sum_{t \in S} \ln \theta_{td}.$$

Идея этого регуляризатора Θ заключается в сглаживании фоновых тем для того, чтобы они оттянули на себя как можно больше нерелевантных термов, и разреживании предметных с целью повышения степени несхожести их друг с другом.

Вновь вспомним о том, что повышение различности распределений термов в темах приводит к росту интерпретируемости тем [34]. Для того, чтобы сделать темы настолько различными, насколько это возможно, используется регуляризатор декорреляции тем в матрице Φ (1.14) для всех предметных тем $t \in S$.

Отметим, что декоррелятор, помимо стимулирования разреженности, имеет тенденцию к группировке термов общей лексики (т.е. термов, которые часто встречаются во многих документах коллекции) в отдельные темы [34]. Для того, чтобы эти темы образовались среди фонового подмножества, в не среди предметных, применяется ещё один аддитивный регуляризатор, равномерно сглаживающий все фоновые темы B.

Таким образом, в общую конструкцию добавляется ещё два регуляризатора:

$$R(\Phi) = -\frac{\tau_5'}{2} \sum_{t \in S} \sum_{s \in S \setminus t} \sum_{w \in W} \phi_{wt}\phi_{ws} + \tau_6' \sum_{t \in B} \sum_{w \in W} \ln \phi_{wt}.$$

Модальность ключевых термов. В качестве альтернативного метода регуляризации на основе априорной лексической информации предлагается использование ключевых термов в качестве отдельной модальности, в дополнение к модальности обычных термов. Модальность ключевых термов определяется словарём Q и матрицей $\tilde{\Phi}$ размера $|Q| \times |T|$.

В 5.1 отмечалось, что логарифм правдоподобия дополнительной модальности представляет собой регуляризатор, влияние которого в данном случае

определяется коэффициентом τ_7':

$$R(\tilde{\Phi}, \Theta) = \tau_7' \sum_{d \in D} \sum_{w \in Q} n_{dw} \ln \sum_{t \in T} \tilde{\phi}_{wt} \theta_{td}.$$

Для того, чтобы сделать целевые темы более различными с точки зрения отражаемых в них подмножеств релевантных термов, используем дополнительный регуляризатор декорреляции тем, действующий на множестве предметных тем в матрице $\tilde{\Phi}$:

$$R(\tilde{\Phi}) = -\frac{\tau_8}{2} \sum_{t \in S} \sum_{s \in S \setminus t} \sum_{w \in Q} \tilde{\phi}_{wt} \tilde{\phi}_{ws}.$$

Ещё раз заметим, что декорреляция предметных тем S вводится отдельно для термов матрицы Φ и термов дополнительной модальности в матрице $\tilde{\Phi}$.

Описанная выше модель M-ARTM, состоящая из восьми регуляризаторов, предложена в работах [19] и [18]. Далее рассматриваются её свойства и результаты в экспериментах по выявлению тем, связанных с этническими обсуждениями.

5.3. Экспериментальные результаты

Применим предложенную выше модель выявления тем, связанных с определённой предметной областью, в задаче выявления этно-социального дискурса, т.е. извлечения из коллекции текстов тематик, связанных с обсуждением этнических вопросов и связанных с ними социальных явлений. Эксперименты производятся на данных популярной российской блог-платформы LiveJournal, полученных с помощью агрегатора. Ставится задача получения набора предметных тем, которые будут как можно более релевантными, разнообразными и интерпретируемыми.

Данные и основные гиперпараметры. Используемая коллекция содержит 1.58 миллиона лемматизированных постов, написанных популярными бло-

герами LiveJournal за годичный период с середины 2013 до середины 2014. Полный словарь коллекции составил порядка 860 тысяч слов, объём итогового словаря после процедуры фильтрации составил 90 тысяч слов. Предобработка заключалась в удалении всех слов, не удовлетворяющих следующим требованиям: содержание только символов кириллицы и не более чем одного дефиса, длина не менее трёх символов, наличие не менее 20 употреблений во всей коллекции.

Все исследуемые модели обучаются с помощью библиотеки BigARTM. Число тем $|T|$ выбиралось из набора значений 100, 300 и 400. В ходе предварительных экспериментов экспертами по оцениванию качества моделирования было выявлено, что модели с 300 и 400 темами дают схожие результаты. Поскольку в схожих более ранних исследованиях [75, 76] было показано, что набор из 400 является оптимальным для рассматриваемых объёмов данных, выбор был остановлен на этом значении. Во всех последующих экспериментах этого раздела строятся модели с $|T| = 400$ тем. Все модели обучаются с помощью онлайнового алгоритма с одним проходом по коллекции и 25 проходами по каждому документу; обновления матрицы Φ производятся каждый 10000 документов.

В качестве словаря данных для частичного обучения Q предварительно подготовлен список *этнонимов* — существительных, отражающих названия различных народностей, из них в коллекции LiveJournal встретились 249 слов. Этнонимы выглядят лучшими кандидатами на роль средства улучшения качества извлечения тем, связанных с этносами и межэтническими отношениями. Участниками таких отношений являются конкретные люди или группы людей. Нужно отличать их от отношений международных, в которых основную роль играют государства, их правительства или официальные представители, а затрагиваемые вопросы далеко не всегда касаются этничностей. Предполагается, что в этнических темах будут превалировать этнонимы (турки), в то время как прилагательные (турецкий) и названия стран (Турция) окажутся более связаны с международными отношениями. В русском языке эти три категории, как правило, представляют собой различные слова, что позволяет проще классифи-

цировать темы по рассматриваемым отношениям на международные и межэтнические.

В то же время, могут встречаться названия народностей, для которых в коллекции может не оказаться соответствующего этнонима-существительного, но будет подходящее прилагательное или название страны. Вопрос пользы от включения этой информации в модель является одним из предметов изучения в описываемых далее экспериментах.

Основные модели для сравнения. Ниже приводится базовый список моделей, качество которых сравнивалось в рассматриваемой задаче. Во всех моделях с регуляризацией соответствующие коэффициенты τ_i' подбирались вручную в ходе многократных запусков обучения. Процедура подбора позиционируется в качестве отдельного дополнительного исследования. Полученная в его ходе информация легла в основу набора рекомендаций по настройке, который рассматривается в конце этого раздела. Во всех моделях с регуляризацией темы разделяются на $|S| = 250$ предметных и $|B| = 150$ фоновых.

Эксперименты производятся со следующими моделями:

1. **Model-Plsa**: классическая модель вероятностного латентного семантического анализа (PLSA) без регуляризаторов;

2. **Model-Lda**: классическая модель латентного размещения Дирихле (LDA), реализованная в BigARTM как PLSA с регуляризаторами сглаживания Φ и Θ равномерными распределениями α и β с гиперпараметрами $\tau_1 = \tau_2 = 10^{-4}$;

3. **Model-Smooth**: модель ARTM со сглаживанием и разреживанием по этнонимам, с коэффициентами регуляризации $\tau_1' = 10^{-5}$ и $\tau_2' = 100$; кроме того, в этом и всех последующих экспериментах использовался описанный ранее регуляризатор матрицы Θ с коэффициентами $\tau_3' = 0.05$ и $\tau_4' = 1$;

4. **Model-Decorrelated**: модель ARTM, усложняющая предыдущую путём добавления декорреляции с параметрами $\tau_5' = 5 \times 10^4$ и $\tau_6' = 10^{-8}$; коэффициент сглаживания этнических тем принимается равным $\tau_1' = 10^{-6}$;

5. **Model-Extended-Dictionary**: модель ARTM, которая усложняет предыдущую путём добавления декоррелируемой модальности этнонимов с коэффициентами $\tau_7' = 100$ и $\tau_8' = 2 \times 10^4$; прочие коэффициенты приняли следующие значения: $\tau_5' = 1.5 \times 10^6$, $\tau_6' = 10^{-7}$ и $\tau_1' = 1.1 \times 10^{-4}$; данная модель использует расширенный словарь: помимо этнонимов, в него добавляются прилагательные и названия стран для тех этничностей, для которых соответствующий этноним в коллекции отсутствует;

6. **Model-Restricted-Dictionary**: модель ARTM, идентичная предыдущей, но в этой модели используется только базовый словарь этнонимов;

7. **Model-Filtered-by-Theta**: модель PLSA, обученная на подмножестве документов, полученных из тем модели Model-Restricted-Dictionary, которые эксперты признали этническими: при этом используются все документы, имеющие в указанных темах в матрице Θ вероятность выше порога 10^{-6};

8. **Model-Keyword-Documents**: модель PLSA, обученная на подмножестве документов всей коллекции, содержащих хотя бы один этноним из Q.

Модели Model-Filtered-by-Theta и Model-Keyword-Documents обучаются для сравнения двух методов обогащения исходной коллекции. Вторая используется в качестве базовой при проверке предположения о том, что последовательное использование тематических моделей может дать лучший результат, чем простое извлечение текстов по ключевым словам.

Метрики качества. В силу практической ориентированности решаемой задачи, ключевой метрикой в ней являются оценки экспертов, определяющие степень интерпретируемости и релевантности получаемых моделей (подробнее они

вводятся непосредственно при описании результатов). Тем не менее, дополнительно качество обучаемых моделей оценивается с помощью среднего значения когерентности. В работах [75, 76] при решение схожей задачи предложен новый вид когерентности, называемый *tf-idf когерентностью*. Показана лучшая корреляция этой метрики с человеческими оценками интерпретируемости по сравнению с классической когерентностью из [67], по этой причине в экспериментах этого раздела используется именно она. Формула для tf-idf когерентности выглядит следующим образом:

$$\mathcal{C}_t^{\text{tf-idf}}(\Phi) = \sum_{i=1}^{k} \sum_{j=1, j\neq i}^{k} \log_{10} \frac{\sum_{d\in D: w_i, w_j \in d} \text{tf-idf}(w_i, d)\ \text{tf-idf}(w_j, d) + \epsilon}{\sum_{d\in D: w_i \in d} \text{tf-idf}(w_i, d)}.$$

Здесь ϵ — небольшая сглаживающая константа (обычно равняется 0.01), а величину tf-idf для терма u в документе d можно рассчитать по формуле:

$$\text{tf-idf} = \left(\frac{1}{2} + \frac{N_u^d}{\max_{v\in d} N_v^d}\right) \log_{10} \frac{|D|}{|\{g \in D : u \in g\}|},$$

где N_u^d обозначает число вхождений слова u в документ d. Все прочие обозначения в определении tf-idf когерентности совпадают с введёнными ранее в (4.2).

Результаты измерения средней по всем темам когерентности. Результаты измерения tf-idf когерентности для каждой модели показаны в таблице 5.1, представлены две версии метрики, посчитанные на 10 и 20 наиболее вероятных слов в каждой теме. Видно, что лучше всех с точки зрения интерпретируемости себя показывает модель Model-Restricted-Dictionary. Тем не менее, все прочие модели, за исключением Model-Smooth и Model-Decorrelated, имеют сопоставимые результаты. Эти факты оказались подтверждены предварительными экспертными оценками, поэтому две наиболее слабые модели исключаются из дальнейшего рассмотрения. Делается это в том числе и для более эффективного использования ограниченного времени асессоров.

Модель	Когерентность (10)	Когерентность (20)
Model-Plsa	-212.0	-1011.6
Model-Lda	-230.9	-1121.2
Model-Smooth	-261.2	-1210.2
Model-Decorrelated	-274.0	-1296.1
Model-Extended-Dictionary	-209.6	-995.3
Model-Restricted-Dictionary	**-196.4**	**-908.4**
Model-Filtered-by-Theta	-212.1	-982.5
Model-Keyword-Documents	-214.4	-1014.5

Таблица 5.1. Средние значения tf-idf когерентности для основных моделей. Столбцы: Модель — вид модели, Когерентность (X) — итоговое среднее значение tf-idf когерентности, посчитанной по X наиболее вероятным словам в темах. Лучшие значения выделены жирным.

Результаты экспертной оценки интерпретируемости и релевантности. Следующим шагом является измерение качества моделей с помощью экспертных оценок. Они собирались на основании наборов наиболее вероятных слов в темах, таблица 5.2 демонстрирует несколько примеров таких наборов, соответствующих релевантным темам. В [18] подробно описан процесс организации асессорской разметки тем в каждой из рассматриваемых моделей, обоснования его корректности и согласованности оценок экспертов. Результаты этой разметки отражены в нижеследующих таблицах и выводах.

Поскольку тренируемые модели пытаются извлечь заданное число тем высокого качества, отодвигая «мусорные» темы в фон, не имеет особого смысла производить сравнение моделей по всем темам. Важнее смотреть на когерентности тех тем, которые были признаны качественными человеческими экспертами.

Таблица 5.3 суммирует связь экспертной интерпретируемости и tf-idf когерентности. В ней отражены темы, которые были признаны асессорами хорошо или же частично интерпретируемыми. Видно, что Model-Restricted-Dictionary в целом превосходит прочие по значению tf-idf когерентности в интерпретируемых темах, однако стоит отметить, что наибольшее с точки зрения экспертов количество интерпретируемых тем выделяет модель Model-Extended-Dictionary.

#	Темы
1	мусульманин, религиозный, ислам, экстрасенс, секта, христианин, радикал, лабиринт, узбекистан, христианский, мусульманский, исламистский, таджикистан, религия, экстремист
2	республика, кавказ, иногда, чеченец, кавказский, дагестан, национальность, чечня, регион, власть, ингушетия, казань, житель, русский, северный
3	армения, азербайджан, армянский, армянин, караван, ереван, таджик, азербайджанский, узбек, сср, татарский, опрос, республика, турок, ильдар, ататюрк, таджикский, туркменский
5	узбек, русский, россия, мигрант, узбекистан, рабочий, москва, страна, таджик, дворник, место, работа, гражданин, дом, азия, жить, полиция
6	русский, узбек, таджик, мигрант, россия, работа, дворник, граница, работа, узбекистан, гастарбайтер, место, город, азия, иногда, рабочий, жить, люди
6	казахстан, азия, регион, центральный, киргизия, таджикистан, афганистан, страна, республика, средний, узбекистан, территория, россия, киргизский, казахский
7	мигрант, страна, россия, миграция, азия, нелегал, мигрантский, таджикистан, гастарбайтер, гражданин, рабочий, работа, средний, узбекистан, таджикский, проблема, русский
7	казахстан, регион, страна, азия, республика, киргизия, русский, государственный, военный, центральный, территория, оборона, сотрудничество, русский, киргизский, афганистан

Таблица 5.2. Примеры наиболее этно-релевантных тем из различных моделей. Столбцы: # — номер модели, Темы — наборы наиболее вероятных слов в темах с указанными номерами. Расшифровка номеров моделей: 1 — Model-Plsa, 2 — Model-Lda, 3 — Model-Smooth, 5 — Model-Restricted-Dictionary, 6 — Model-Extended-Dictionary, 7 — Model-Filtered-by-Theta.

Модель	Темы	Когерентность (10)	Когерентность (20)
Model-Plsa	258	-173.6	-828.7
Model-Lda	312	-205.5	-988.0
Model-Extended-Dictionary	324	-180.2	-838.5
Model-Restricted-Dictionary	249	**-165.5**	**-789.6**
Model-Filtered-by-Theta	297	-180.9	-846.3
Model-Keyword-Documents	220	-168.5	-794.6

Таблица 5.3. Средние значения tf-idf когерентности, посчитанные по группе интерпретируемых тем для основных моделей. Столбцы: Модель — вид модели, Темы — размер группы интерпретируемых тем, Когерентность (X) — итоговое среднее значение tf-idf когерентности, посчитанной по X наиболее вероятным словам в указанных темах. Лучшие значения выделены жирным.

В каждой модели асессоры разметили темы по двум критериям: вид темы (тема этническая; тема о международных отношениях; тема и о том, и

Вид тем	Релевантность тем								
	частичная			полная			частичная или полная		
	# тем	К. (10)	К. (20)	# тем	К. (10)	К. (20)	# тем	К. (10)	К. (20)
Model-Plsa									
Этническая	5	-190.2	-904.8	12	-207.1	-996.3	17	-202.1	-969.4
Межд. отношения	20	-150.7	-733.8	19	-194.0	-946.8	39	-171.8	-837.6
Этническая и МО	20	-163.0	-784.9	25	-194.3	-938.7	45	-180.4	-870.3
Model-Lda									
Этническая	2	**-124.4**	**-646.0**	13	-190.0	-927.9	15	-181.3	-890.3
Межд. отношения	21	-158.9	-763.1	29	-225.7	-1097.5	50	-197.7	-957.1
Этническая и МО	18	-162.3	-777.7	37	-212.2	-1023.0	55	-195.9	-942.7
Model-Extended-Dictionary									
Этническая	**18**	-164.7	-798.5	30	-222.3	-1015.8	**48**	-200.7	-934.3
Межд. отношения	**33**	-142.5	-707.7	26	-207.4	-917.3	59	-171.1	-800.1
Этническая и МО	**36**	-142.0	**-695.1**	47	-211.1	-958.4	**83**	-181.1	-844.2
Model-Restricted-Dictionary									
Этническая	8	-160.5	-805.1	22	**-150.0**	**-713.8**	30	**-152.8**	**-738.2**
Межд. отношения	18	**-126.3**	**-641.1**	29	**-156.3**	**-740.8**	47	**-144.8**	**-702.6**
Этническая и МО	22	**-136.5**	-707.7	37	**-158.3**	**-741.8**	59	**-150.2**	**-729.1**
Model-Filtered-by-Theta									
Этническая	**18**	-181.3	-952.6	22	-201.8	-971.4	40	-192.6	-962.9
Межд. отношения	30	-161.6	-780.4	30	-171.4	-827.3	**60**	-166.5	-803.9
Этническая и МО	34	-161.3	-810.6	47	-180.1	-869.8	81	-172.2	-844.9
Model-Keyword-Documents									
Этническая	5	-161.1	-805.0	**37**	-175.6	-834.7	42	-173.9	-831.1
Межд. отношения	18	-138.4	-670.7	**32**	-164.3	-782.9	50	-155.0	-742.5
Этническая и МО	17	-154.3	-741.3	**52**	-165.5	-793.1	69	-162.8	-780.4

Таблица 5.4. Количество релевантных тем и средние значения tf-idf когерентности, посчитанные по этим темам для основных моделей. Столбцы: # тем — количество найденных релевантных тем, К. (X) — итоговое среднее значение tf-idf когерентности, посчитанной по X наиболее вероятным словам в указанных темах. Лучшие значения выделены жирным.

о другом; тема иная) и степень релевантности (нерелевантная; частично релевантная; полностью релевантная). Таблица 5.4 суммирует наиболее важные результаты, определяющие степень релевантности тем цели исследования.

Исходя из неё можно сделать выводы, схожие с полученными выше: Model-Restricted-Dictionary превосходит всех с точки зрения когерентности своих ре-

левантных тем, в то же время, модель Model-Extended-Dictionary содержит наибольшее количество релевантных тем. Отсюда следует ожидаемый вывод о том, что расширение словаря Q приводит к повышению числа релевантных с точки зрения асессоров тем (что часто более полезно в социологических исследованиях), но ухудшает их среднюю когерентность.

Стоит отметить, что Model-Filtered-by-Theta в целом очень похожа на модель Model-Extended-Dictionary — она тоже содержит большое количество релевантных тем, которые имеют ощутимо более плохую среднюю tf-idf когерентность, чем у тем Model-Restricted-Dictionary. Можно сделать вывод о том, что такой способ повторного построения модели оказался малоэффективным.

Модель Model-Keyword-Documents показывает неплохие результаты: уступая когерентностью только лучшей модели Model-Restricted-Dictionary, она существенно превосходит её по числу релевантных тем, особенно полностью релевантных. Тем не менее, стоит учитывать, что наборы тем, извлекаемых этими двумя моделями, получается существенно различным. Model-Restricted-Dictionary, пусть и ограниченная словарём Q, имеет возможность извлекать релевантную информацию, не содержащую этнонимы напрямую, но включающую в себя слова, часто встречающиеся с этнонимами. Полнота же модели Model-Keyword-Documents напрямую жёстко зависит от полноты Q.

Результаты переобучения после фильтрации по Φ. Естественным следствием из описанного выше набора экспериментов является идея опробовать ещё один подход, в рамках которого последовательно строятся две тематические модели. Как и в случае Model-Filtered-by-Theta, на первом этапе используется Model-Restricted-Dictionary. Отличие заключается в том, что новая коллекция формируется не на основании вероятности документов в темах, которые эксперты признали релевантными. Вместо этого производится фильтрация документов по признаку наличия в них хотя бы одного слова из множества наиболее вероятных слов всех предметных тем S. Таким образом, новая модель

#	Гиперпараметры							
	τ'_1	τ'_2	τ'_3	τ'_4	τ'_5	τ'_6	τ'_7	τ'_8
1	10^{-4}	0.01	0.05	1.0	0	10^7	1.0	0
2	0	0	0	0	0	0	100.0	0
3	2.5×10^{-4}	0.01	0.05	1.0	0	10^7	1.0	0
4	10^{-4}	0.01	0.05	1.0	0	10^7	100.0	0
5	2.5×10^{-4}	0.01	0.05	1.0	0	10^7	100.0	0
6	5×10^{-3}	0.01	0.05	1.0	0	10^7	100.0	0
7	5×10^{-5}	0.01	0.05	1.0	2×10^5	10^7	100.0	2×10^4
8	5×10^{-5}	0.01	0.05	1.0	10^5	10^7	100.0	10^4
9	5×10^{-5}	0.01	0.05	1.0	10^5	10^7	100.0	10^4

Таблица 5.5. Гиперпараметры моделей, строящихся на фильтрованной коллекции. Столбцы: # — номер модели.

идейно представляет собой гибрид Model-Filtered-by-Theta и Model-Keyword-Documents. Ещё одним отличием является то, что на втором этапе вместо обучения модели PLSA на фильтрованной коллекции предлагается построить целый набор различных моделей M-ARTM.

Размер полученной после фильтрации коллекции составил около 320 тысяч документов, количество этнонимов в Q осталось прежним. Существенно уменьшившийся объём данных позволил провести построение большого числа моделей, в данной работе рассматривается девять из них с такими наборами параметрами, которые могут привести к существенно различному результату. Таблица 5.5 демонстрирует значения этих параметров; нужно учитывать, что модель с номером 9 имеет набор параметров, аналогичный модели 8, однако 9 обучается посредством трёх проходов по коллекции, тогда как все прочие модели обучаются за один проход.

Для обеспечения сравнимости с предыдущими результатами, все новые модели обучаются с тем же набором тем, состоящим из $|S| = 250$ и $|B| = 150$ тем. Кроме того, когерентность считается, как и раньше, с использованием счётчиков, собранных по всей коллекции.

В таблице 5.6 приводятся средние значения tf-idf когерентности новых мо-

#	Темы	Когерентность (10)	Когерентность (20)
Все темы			
1	400	-259.3	-1203.5
2	400	-215.6	-1015.7
3	400	-258.8	-1210.6
4	400	**-191.3**	**-869.8**
5	400	**-191.2**	**-870.1**
6	400	-220.0	-1011.8
7	400	-286.9	-1296.4
8	400	-264.5	-1223.8
9	400	-256.6	-1199.5
Предметные темы			
1	250	-323.1	-1505.1
2	250	-208.1	-980.3
3	250	-322.6	-1517.6
4	250	**-198.4**	**-909.6**
5	250	**-198.6**	**-912.7**
6	250	-251.9	-1171.1
7	250	-358.1	-1626.7
8	250	-320.3	-1506.1
9	250	-313.3	-1491.6
Фоновые темы			
1	150	**-152.9**	**-701.0**
2	150	-228.1	-1074.9
3	150	**-152.5**	**-699.0**
4	150	-179.6	-803.6
5	150	-178.9	-799.2
6	150	-166.9	-746.2
7	150	-168.3	-746.0
8	150	-171.5	-753.4
9	150	-162.0	-712.8

Таблица 5.6. Средние значения когерентности тем в моделях, построенных по фильтрованной коллекции. Столбцы: # — номер модели, Когерентность (X) — итоговое среднее значение tf-idf когерентности, посчитанной по X наиболее вероятным словам в темах. Лучшие значения выделены жирным.

делей. Первые девять строк представляют собой среднюю когерентность всех тем и могут напрямую сравниваться со значениями в таблице 5.1. Видно, что

лучшие модели второго этапа обучения по общей средней когерентности превосходят лучшую модель первого этапа. Сравнивая модели с номерами 8 и 9, можем заметить, что дополнительные проходы по коллекции дали не столь большое улучшение модели, как правильно подобранные коэффициенты регуляризации. Это означает, что лучше обучать несколько моделей с разным набором гиперпараметров, используя один проход по коллекции, чем делать несколько проходов для обучения какой-либо одной из них.

Таблица 5.6 также предоставляет оценки средней tf-idf когерентности отдельно для предметных и фоновых подмножеств тем. Проявляется интересный эффект: в целом у фоновых тем значение когерентности существенно лучше, чем у предметных. Такой эффект наблюдается в связи с тем, что размер предметного множества тем $|S| = 250$ оказался слишком велик для моделей, обучающихся на урезанном наборе данных. Для наглядности в таблице 5.7 показаны примеры тем из множеств S и B модели с номером 4. В множестве предметных тем присутствуют как качественные этно-релевантные темы, так и темы полностью мусорные, т.е. и не релевантные, и не интерпретируемые (номера 92 и 232 в таблице 5.7); в то же время, фоновые темы, как правило, имеют мало отношения к этническим вопросам, зато оказываются качественными с точки зрения интерпретируемости (что соответствует их более высокой когерентности).

Настройка коэффициентов регуляризации. Теория аддитивной регуляризации представляет большую гибкость в конструировании различных моделей, что наглядно продемонстрировано выше. Обратной стороной такой гибкости является растущая по мере добавления каждого нового регуляризатора сложность настройки гиперпараметров. Лучшая модель, обученная в этом разделе, Model-Restricted-Dictionary, включает в себя восемь регуляризаторов, и далеко не очевидно то, какие значения должны принимать их коэффициенты для получения даже не наиболее качественной, а хотя бы не вырожденной модели. Ниже приводится обсуждение эмпирических правил по настройке моделей

#	Темы
Примеры предметных тем	
1	ирландский, ирландия, время, день, пиво, святой, страна, друг, место, добро, жизнь
12	мигрант, узбек, русский, россия, работа, москва, место, узбекистан, занятие, страна, дворник
14	шотландский, шотландия, виски, напиток, пиво, время, бутылка, место, добро, век, английский, день, мера
173	сирийский, сирия, оружие, боевик, али, армия, дамаск, террорист, регион, гора, военный
92	мать, развернуть, клиент, ткань, олень, куприянович, путинга, ортопед, иегова, марфино, округление
232	сортировка, анб, путешествие, жан, красновка, джезве, советский, устрица, краснодар, пытка, ташкент
Примеры фоновых тем	
35	женщина, мужчина, семья, девушка, женщина, жизнь, прекрасный, жена, красный
48	цена, стоимость, недвижимость, аренда, покупка, покупатель, хороший, средний, продукт, площадь
36	больница, медицинский, операция, пациент, клиника, медицина, здоровый, лечение, здравоохранение
99	игра, команда, игра, игрок, сезон, футбол, стадион, победа, соккер, чемпионат, спорт, болельщик
101	цвет, красный, черный, зеленый, белый, синий, цветок, колор, тень, место, розовый

Таблица 5.7. Примеры наиболее тем из множеств S и B модели второго этапа с номером 4. Столбцы: # — идентификатор темы в модели, Темы — наборы наиболее вероятных слов в темах с указанными номерами.

из этого раздела, представленных исходно в работе [19], а также описание связанных с ними сложностей.

Проблема оптимальной конфигурации параметров регуляризации в общем случае не имеет полного окончательного решения. В практических экспериментах её решение сводится к перебору различных вариантов и выбору комбинаций, дающих наилучший результат. Вопрос оптимизации этого процесса обычно сводится к набору правил, позволяющих в определённых случаях сократить перебор, а также описания ситуаций, явно указывающих на неправильный выбор значения некоторого гиперпараметра. Это позволяет сосредоточить внимание на относительно узких диапазонах значений и в результате быстрее получить нужный результат.

В ходе проделанных в этом разделе экспериментов построено большое количество разнообразных вариаций тематических моделей. Для каждой из них перепробовано множество значений гиперпараметров и получены автоматическая и предварительная асессорская оценки. Лучшие из полученных таким образом моделей прошли дополнительную полноценную экспертную оценку и подробно рассмотрены ранее. Однако ценность заключается не только в них самих, но и в том процессе, результатом которого стал отбор этих моделей. Рассмотрим особенности настройки различных комбинаций регуляризаторов поотдельности.

Регуляризаторы равномерного сглаживания Φ *и* Θ. В рассматриваемой версии модели LDA используются симметричные априорные распределения Дирихле для обеих матриц параметров, что означает необходимость настройки двух гиперпараметров, τ_1 и τ_2. Выбранное для обоих коэффициентов значение 10^{-4} показало себя наилучшим образом; диапазон хороших значений включает в себя числа от 10^{-4} до 10^{-3}. Меньшие значения не оказывают на модель никакого влияния (при сравнении с моделью PLSA), а большие приводят к перерегуляризации, в результате которой темы в модели получаются чересчур общими, сильно пересекающимися и в целом бесполезными. Пример такой ситуации приводится в таблице 5.8(a).

Сглаживающий регуляризатор Φ *для частичного обучения.* Ключевым параметром модели Model-Smooth является коэффициент τ_1' регуляризатора сглаживания по этнонимам. Вновь были опробованы различные значения, с точностью до порядка выявлен диапазон хороших значений, в котором лучший результат получен при $\tau_1' = 10^{-5}$. Уменьшение этой величины приводит к падению числа релевантных тем в множестве S, а существенное увеличение — к появлению тем, состоящих только из несвязанных друг с другом по смыслу этнонимов, как демонстрирует таблица 5.8(b).

Регуляризаторы сглаживания и разреживания Θ. Коэффициенты регуляризации в случае матрицы Θ не подвергались существенному отбору, поскольку

(*a*) Пересглаженная Model-Lda, τ_1 и τ_2 больше чем 10^{-3}
хороший, дом, время, рубль, старый, сладкий, стоимость, русский, голос, работа, день, час, октябрь, слово, говорить, группа, сервер, любовь, правда
время, хорошо, жизнь, день, ребенок, дело, россия, старый, час, полный, кран, история, разный, красный, москва, кино, женщина, советский, рассказать
хороший, место, старый, день, россия, слово, сладкий, другой, стоимость, полный, дело, рука, нога, деньги, время, история, вопрос, рассказать
(*b*) Model-Smooth со слишком большим τ_1'
монголоидный, итальянский, ирландский, иудейский, немецкий, бедуинский, сербский, сомалийский, неверный, даргинский, боснийский, сингапурский, темнокожий, перуанский, русский, маньчжурский, сицилийский
черногорский, новозеландец, суахили, мари, алжир, японец, кубинец, сомалиец, серб, сингапурец, кенигсберг, венгер, немец, ногай, алан, английский, эфиоп, венесуэльец, грек, американец
новозеландец, венгр, негр, ингуш, австриец, тувинец, украинец, якут, марокканец, папуа, канадец, белорус, сицилиец, курд, суахили, скандинав, латиноамериканец, гагауз, узбек
(*c*) Модели со слишком большим коэффициентом τ_4' разреживания Θ
русский, ненецкий, новозеландский, латиноамериканский, османский, монголоидный, апач, кавказский, чилийский, индейский, иудейский, мадьярский, венесуэльский, суданский, пермякский, американский, кигрызский, ингрианский, нигерийский, немецкий
курдский, эскимосский, кхмерский, вьетнамский, черногорский, боснийский, марокканский, лезгинский, башкирский, финно-угорский, перуанский, мексиканский, крымский, катарский, кенигсбергский, шоу, ацтекский, азербайджанский, балтийский, сибирский
сирийский, карельский, кабардинский, вепсский, монгольский, хантыйский, берберский, израильский, новозеландский, абхазский, бразильский, богемский, ненецкий, итальянский, доминиканский, афроамериканский, тунгисский, алжирский, кенигсвинтер, манси
(*d*) Модели со слишком большим коэффициентом декорреляции τ_5'
врываться, ниеншанц, мизантроп, эрзурум, неполноценный, выкупить, дерзкий, спокойно, приблизительно, шапито, мессия, вокруг, затишье, революция, осман
свердловск, федеральный, пальмовед, термозащита, некультурный, формально, эцп, уралполит, иннополис, экспо, американский, алтайский, суперконденсатор, русский, формальный, китайский
рейхстаг, димитров, телестудия, жигулевск, лягушки, заткнуться, кассандра, себастьян, американец, великанов, плеханов, трехдневка, еврей

Таблица 5.8. Примеры испорченных тем, возникающих в моделях из-за неправильной настройки регуляризаторов.

на практике имеют широкие диапазоны подходящих значений. Тем не менее, перерегуляризация Θ так же возможна. Достаточно увеличить коэффициент τ_4' в пять раз, и модель испортится так, как показано в таблице 5.8(c): сильно разре-

живание обнуляет счётчики всех слов, поэтому сохраняются только этнонимы, поддерживаемые сглаживанием в матрице Φ. В результате стартовые параметры модели с перерегуляризованной Θ в условиях онлайнового алгоритма не изменяются в процессе обучения.

Регуляризаторы декорреляции и сглаживания фона в Phi. В модель ModelDecorrelated добавляются два новых гиперпараметра: коэффициент декоррелирования предметных тем τ_5' и коэффициент сглаживания фоновых тем τ_6'. Последний не требует особенной настройки, единственное требование заключается в том, что его значение должно в принципе быть очень малым (чтобы не было перерегуляризации), и точно существенно меньшим, чем τ_2': фоновое сглаживание должно быть незначительным по сравнению с разреживанием тем из B по множеству этнонимов Q.

Ключевыми гиперпараметрами, требующими настройки в этом случае, являются значения τ_5' и τ_1'. Строго говоря, эти регуляризаторы преследуют противоположные цели: декоррелирование приводит к разреживанию, в то время как регуляризатор для частичного обучения пытается увеличить веса счётчиков, соответствующих этнонимам в предметных темах. Второй регуляризатор является проблемно-ориентированным, первый же предназначен для увеличения разнообразности и интерпретируемости тем в целом. В результате возникает проблема одновременной аккуратной настройки двух гиперпараметров, которая решается путём подбора пар значений коэффициентов. Эти два регуляризатора сильно взаимосвязаны, поэтому существенная бесконтрольная вариация их коэффициентов может привести порче модели. Увеличение τ_5' в 3-4 раза приведёт к получению неинтерпретируемых тем со случайными вставками этнонимов (таблица 5.8(d)). Рост же τ_1' на один порядок приводит к сильному пересглаживанию, то есть получению неинтерпретируемых тем, состоящих только из сглаживаемых этнонимов (таблица 5.8(b)).

Полный набор регуляризаторов. Итоговая модель суммирует все предыдущие результаты и дополнительно усложняется ещё одним регуляризатором

— правдоподобием модальности этнонимов. Этот регуляризатор с коэффициентом $\tau_7' = 100$, как выяснилось, оказывает наиболее сильное влияние на модель в целом; он позволяет значительно улучшить качество моделирования и решения задачи. Взаимосвязи между регуляризаторами в этом случае далеко не очевидны, поэтому требуется совместная настройка всех гиперпараметров модели. Для регуляризатора матрицы Θ значения гиперпараметров остаются фиксированными, поскольку дают удовлетворительные результаты. То же самое относится и разреживанию фоновых тем основной модальности по этнонимам (τ_2'). Это позволяет сократить размерность пространства поиска.

Все прочие гиперпараметры настраивались совместно, чтобы сперва получить адекватную в целом модель, а затем, двигаясь небольшими шагами, подобрать достаточно оптимальную итоговую конфигурацию. Коэффициент τ_7' требует наибольшего внимания, поскольку слишком низкое его значение при фиксации прочих гиперпараметров приводит к слишком сглаженным по этнонимам предметным темам (таблица 5.8(b)). Слишком большое значение, равно как и фиксированное оптимальное при полном отключении других регуляризаторов, даёт адекватную модель, однако по качеству она уступает полным моделям Model-Restricted-Dictionary или Model-Extended-Dictionary.

5.4. Транзакционные модели T-ARTM

Обычные тематические модели текстовых коллекций описывают вхождения слов в документы. Мультимодальные модели описывают документы, в которых содержатся термы различных модальностей: слова, теги, авторы, и т. д. Во всех этих случаях модель описывает парные взаимодействия между документами и термами. В более сложных приложениях исходные данные могут описываться транзакциями (отношения, взаимосвязи, взаимодействия) между тремя и более объектами различных модальностей. Например, в сети интернет-рекламы «пользователь u кликнул объявление b на странице s»; в социальной сети «поль-

зователь u написал слово w на странице блога d»; в сети продаж «покупатель b купил у продавца s товар g»; в пассажирских авиаперевозках «клиент u вылетел из аэропорта x в аэропорт y самолётом авиакомпании a»; в рекомендательной системе «клиент u оценил фильм f в ситуативном контексте s». Ещё одной модальностью может быть дата и время транзакции.

Во всех приведённых примерах взаимодействие объектов не сводится к парным взаимодействиям. В других случаях сложные взаимодействия всё же распадаются на пары. Например, в системе рекомендаций музыки транзакция «трек r исполнителя a находится в альбоме p, вышедшем в году y» описывается, казалось бы, четвёркой объектов (r, a, p, y). Однако она распадается на парные взаимосвязи (p, r), (p, a), (p, y), которые могут быть описаны обычной мультимодальной моделью.

Для тематического моделирования транзакционных данных удобно понятие гиперграфа. Гиперграф обобщает понятие графа и отличается от него тем, что рёбрами в нём могут быть не только пары вершин, но и подмножества из трёх и более вершин. Вершины гиперграфа соответствуют термам различных модальностей, рёбра — транзакциям. Задача заключается в том, чтобы по наблюдаемой выборке транзакций восстановить неизвестные тематические распределения вершин $p(t \mid v)$. Предполагается, что вероятность транзакции тем выше, чем более схожи тематики её вершин.

Гиперграфовая модель транзакционных данных. *Гиперграф* $\Gamma = \langle V, E \rangle$ определяется множеством вершин-термов V и множеством рёбер (транзакций) E. Каждое ребро e из E образуется подмножеством вершин, $e \subset V$.

Каждая вершина $v \in V$ относится к модальности с номером $m = \mu(v)$ из конечного множества номеров модальностей M. Множество всех вершин разбивается на непересекающиеся подмножества по модальностям:

$$V = \bigsqcup_{m=1}^{M} V^m, \qquad V^m = \{v \in V : \mu(v) = m\}.$$

Пусть задано множество типов транзакций K. *Транзакционные данные* типа k — это выборка E_k независимых наблюдений $(e,t) \in 2^V \times T$, порождаемая распределением $p_k(e,t)$, своим для каждого типа $k \in K$. Каждое ребро $e \in E_k$ входит в выборку n_{ke} раз, и с каждым вхождением ребра связана своя латентная тема $t \in T$.

Будем полагать, что в каждой транзакции $e \in E$ имеется одна выделенная вершина d, называемая *контейнером*, и обозначать ребро через $e = (d, x)$, где x — множество всех остальных вершин ребра e, за исключением вершины-контейнера d. Аналогично документу, с контейнером связано распределение тем $p(t \mid d)$. Множество всех вершин-контейнеров обозначим через D.

Далее предположим, что ни распределения тем $p(t \mid d)$ в контейнере d, ни распределения вершин в темах $p(v \mid t)$ не зависят от типа ребра k. Казалось бы, на практике это предположение может не выполняться. Однако ограничение нетрудно обойти, если построить модель с тремя разными модальностями слов для этих трёх типов транзакций. Более того, механизм регуляризации позволяет связать эти распределения и сделать их похожими.

Наконец, введём гипотезу условной независимости вершин в рёбрах (d, x):

$$p(x \mid t) = \prod_{v \in x} p(v \mid t).$$

При сделанных допущениях процесс порождения ребра $(d, x) \in E_k$ состоит из двух шагов. Сначала порождается тема t из распределения $p(t \mid d)$. Затем порождается множество вершин $x \subset V$, причём каждая вершина $v \in x$ модальности m порождается независимо от других вершин из своего распределения $p(v \mid t)$ над множеством V^m.

Тематическая модель выражает вероятности появления рёбер гиперграфа через условные распределения, связанные с их вершинами:

$$p(x \mid d) = \sum_{t \in T} p(t \mid d) \prod_{v \in x} p(v \mid t) = \sum_{t \in T} \theta_{td} \prod_{v \in x} \phi_{vt}.$$

Параметрами этой модели являются условные вероятности вершин в темах

$\phi_{vt} = p(v \mid t)$, нормированные по каждой модальности $v \in V^m$, и условные вероятности тем в контейнерах $\theta_{td} = p(t \mid d)$. В матричных обозначениях параметрами являются матрицы Φ^m, $m = 1, \ldots, M$, и Θ, как и в случае мультимодальной тематической модели.

Гиперграфовый EM-алгоритм для модели T-ARTM. Для оценки параметров модели применим принцип максимума правдоподобия. Будем максимизировать сумму логарифмов правдоподобия по всем типам рёбер k с весами τ_k и регуляризатором $R(\Phi, \Theta)$:

$$\sum_{k \in K} \tau_k \sum_{dx \in E_k} n_{kdx} \ln \sum_{t \in T} \theta_{td} \prod_{v \in x} \phi_{vt} + R(\Phi, \Theta) \to \max_{\Phi, \Theta};$$

$$\sum_{v \in V^m} \phi_{vt} = 1, \ \phi_{vt} \geq 0; \tag{5.7}$$

$$\sum_{t \in T} \theta_{td} = 1, \ \theta_{td} \geq 0. \tag{5.8}$$

Описанная выше модель называется T-ARTM (гиперграфовая или транзакционная модель ARTM). Имеет место следующая теорема:

Теорема 2. *Пусть функция $R(\Phi, \Theta)$ непрерывно дифференцируема. Точка локального максимума (Φ, Θ) задачи (5.7)–(5.8) удовлетворяет системе уравнений относительно параметров модели ϕ_{vt}, θ_{td} и вспомогательных переменных $p_{tdx} = p(t \mid d, x)$, если из решения исключить нулевые столбцы матриц Φ_m, Θ:*

$$p_{tdx} = \underset{t \in T}{\operatorname{norm}} \left(\theta_{td} \prod_{v \in x} \phi_{vt} \right). \tag{5.9}$$

$$\phi_{vt} = \underset{v \in V^m}{\operatorname{norm}} \left(n_{vt} + \phi_{vt} \frac{\partial R}{\partial \phi_{vt}} \right); \qquad n_{vt} = \sum_{k \in K} \sum_{dx \in E_k} [v \in x] \, \tau_k n_{kdx} p_{tdx}; \tag{5.10}$$

$$\theta_{td} = \underset{t \in T}{\operatorname{norm}} \left(n_{td} + \theta_{td} \frac{\partial R}{\partial \theta_{td}} \right); \qquad n_{td} = \sum_{k \in K} \sum_{dx \in E_k} \tau_k n_{kdx} p_{tdx}. \tag{5.11}$$

5.5. Детали реализации T-ARTM в BigARTM

В рамках данной работы в библиотеке BigARTM реализован дополнительный режим работы существующего EM-алгоритма, позволяющий производить обучение тематических моделей T-ARTM. Рассмотрим ряд технических деталей, связанных с внедрением этого изменения.

Формат входных данных. Данные в BigARTM могут подаваться как в виде готовых пакетов, представляющих собой бинарные файлы сериализованных сообщений protobuf, так и в виде текстовых файлов, имеющих определённый формат. Во втором случае до начала обучения пакеты создаются из этих файлов встроенным в библиотеку парсером. Рассмотрим основной формат входных текстовых данных под названием Vowpal Wabbit или VW (в силу его схожести с форматом входных данных одноимённой библиотеки). Коллекция упаковывается в файл (или несколько файлов), в котором каждому документу в наиболее общем случае соответствует одна строка следующего вида:

```
doc_id |m_1 t_11: c_11 ...t_1n: c_1n ...|m_k t_k1: c_k1 ...t_km: c_km
```

Сперва указывается идентификатор документа, затем идут блоки данных, соответствующие различным модальностям: каждый новый блок начинается с имени модальности, перед которым указывается символ «|». Внутри каждого блока находятся слова соответствующей модальности, справа от каждого слова после символа «:» указывается частота его встречаемости в документе. Такой формат без каких либо сложностей описывает и обычный текст: достаточно не указывать никаких модальностей, все термы в этом случае будут отнесены к модальности по умолчанию (основной модальности).

Обеспечение поддержки транзакционных данных требует модификации формата. Обозначим для краткости всё содержимое строки-примера после идентификатора `doc_id` за `tokens`. Тогда новая версия строки, соответствующей одному документу в формате VW, примет вид

```
doc_id ||tt_1 tokens_11 || ...tokens_1n ...||tt_k tokens_k1 || ...tokens_km
```

Здесь `tokens_ij` представляет собой одну транзакцию, то есть кортеж из нескольких термов различных модальностей, которые по смыслу связаны друг с другом. Каждая транзакция относится к некоторому типу $k \in K$ и обозначается, по аналогии с модальностями, своим именем, указанным после управляющей конструкции «||». Для того, чтобы отделять друг от друга транзакции одного типа, используется эта же управляющая конструкция. Учёт веса транзакции производится заданием веса первого входящего в неё слова, веса прочих слов игнорируются.

Полученный формат обобщает предшественника: в отсутствие указаний транзакций и их типов все слова всех модальностей считаются транзакциями длины 1, относящимися к типу транзакций по умолчанию.

Формат пакета документов. Библиотека protobuf даёт возможность описывать структурированные данные в виде сообщений на компилируемом псевдоязыке. Пакеты (`Batch`) и документы (`Item`) в BigARTM, как указано выше, являются объектами protobuf, и сообщения для них имеют следующий вид:

```
message Batch {
 repeated string token = 1;
 repeated string class_id = 2;
 repeated Item item = 3;
 optional string description = 4;
 optional string id = 5;
}
```

```
message Item {
 optional int32 id = 1;
 optional string title = 3;
 repeated int32 token_id = 4;
 repeated float token_weight = 5;
}
```

В пакете, помимо идентификатора и описания, хранятся словарь этого пакета, представляющий собой массив строковых описаний термов и соответствующий ему массив имён модальностей этих термов, и список документов. Каждый документ, в свою очередь, кроме метаданных хранит списка входящих в него термов соответствующие этим термам индексы в словаре пакета, в

другом массиве аналогичной длины хранятся счётчики встречаемости термов в документе.

Для того, чтобы иметь возможность хранить в этих сообщениях транзакционные данные, не утратив при этом обратной совместимости, предлагается изменить сообщения, добавив в них следующие поля:

```
message Batch {
...
 repeated string transaction_typename = 6;
}

message Item {
...
 repeated int32 transaction_start_index = 6;
 repeated int32 transaction_typename_id = 7;
}
```

Пакет теперь содержит список названий всех типов транзакций, которые встречаются в его документах. В каждый документ добавляется целочисленный массив `transaction_start_index`, длина которого совпадает с количеством транзакций в документе, и i-й его элемент равен стартовому индексу i-й транзакции в массиве `token_id`. А соответствующий ему целочисленный массив `transaction_typename_id` содержит для каждой транзакции индекс имени её типа в `transaction_typename`.

Модификация вычислений в ядре. Прежде, чем переходить к реализации изменений EM-алгоритма, необходимо передать модели список имён используемых типов транзакций и соответствующие им коэффициенты. Делается это простым включением двух массивов с указанной информацией в protobuf-сообщение, описывающее конфигурацию модели. Это сообщение доступно в ядре библиотеки и даёт возможность обрабатывать только транзакции нужных типов с нужными весами, игнорируя все прочие данных из пакетов.

Наиболее узким с вычислительной точки зрения местом библиотеки является модуль `Processor`, выполняющий параллельное вычисление E-шага алгоритма. Именно с ним связана значительная часть описанных в предыдущих главах оптимизаций. Предварительные эксперименты показали, что реализация обобщённого алгоритма, работающего как на транзакционных, так и на обычных мультимодальных данных, приводит росту времени обучения последних при сохранении прочих условий. По этой причине было принято решение реализовать поддержку T-ARTM в виде отдельного модуля. Если параметры модели и содержимое пакетов документов соответствуют транзакционным данным, запускается одна версия кода для подсчёта E-шага, если обычным — другая. Стоит отметить, что такой подход никак не сказывается на параллельности вычислений, она обеспечивается одними и теми же средствами в обоих случаях.

К основным отличиям версии `Processor` для T-ARTM от обычной можно отнести наличие циклов по стартовым индексам транзакций в документах, условий, проверяющих наличие указанных пользователем весовых коэффициентов типов транзакций и модифицированный способ подсчёта переменных p_{tdw} (в данном случае p_{tdx}). Первые два изменения являются основными причинами замедления обобщённой реализации при работе с обычными данными: будучи расположенными в наиболее интенсивно используемых частях программного кода ядра, они многократно выполняются и тратят процессорное время без какой-либо пользы.

Подсчёт p_{tdx}, согласно (5.9), основывается на вычислении для каждой темы величины $\theta_{td} \prod_{v \in x} \phi_{vt}$. Вычисленный однократно набор значений сохраняется в массив для последующей нормировки. Важной технической деталью является возможность выхода результата подсчитываемого произведения за границы поддерживаемой типом (в реализации используется `double`) точности. Это не позволяет обрабатывать длинные транзакции из более чем нескольких термов. Очевидно, что в этой ситуации можно без особых сложностей использовать другой, более ёмкий тип для хранения вещественных чисел. Тем не менее, в данном

случае кажется логичным сперва решить проблему на уровне теоретического аппарата и модифицировать формулы таким образом, чтобы они позволяли учитывать транзакции произвольной длины без необходимости в специализированных технических решениях.

5.6. Экспериментальные результаты

Качество работы тематической модели в числе прочего характеризуется тем, насколько точно модель может по наблюдаемым данным (документам) восстановить параметры, из которых эти данные были сгенерированы. В следующем эксперименте производится генерация транзакционной текстовой коллекции D из некоторого известного синтетического набора параметров, после чего сравнивается качество восстановления этих параметров при использовании следующих моделей: равномерное распределение параметров, модель PLSA, модель M-ARTM, модель T-ARTM.

Генерация данных. Прежде, чем приступить к генерации документов, необходимо определить параметры модели. Для этого зафиксируем следующие величины: число тем $|T|$; набор из M модальностей; количество уникальных термов M_i в каждой из них; набор типов транзакций K. Считаем, что основной модальностью (или модальностью обычных слов) в генерируемых параметрах является модальность с индексом 1. Кроме того, требуется определить набор констант N_Θ, C_Θ, N_Φ и C_Φ, смысл которых описывается ниже.

Генерация каждого столбца матриц Φ и Θ производится независимо. Сперва все значения в столбце генерируются из равномерного распределения. Далее, в матрице Θ так же из равномерного распределения случайным образом выбираются N_Θ элементов, значение которых умножается на коэффициент C_Θ, после чего весь столбец нормируется.

Для матрицы Φ процесс выглядит схожим образом: после генерации всех

элементов столбца также происходит выбор N_Φ случайных элементов, но только среди термов, входящих в основную модальность. Эти значения умножаются на коэффициент C_Φ, после чего наборы значений в столбце, соответствующие различным модальностям, нормируются.

Такой способ генерации позволяет получить параметры, удовлетворяющие гипотезе разреженности реальных текстовых данных: каждая тема существенно характеризуется несколькими термами из словаря, каждый документ по большей части относится только к нескольким темам.

После получения матриц Φ и Θ произведём с их помощью генерацию текстовой коллекции. Для этого требуется дополнительно определить несколько параметров: число документов $|D|$; минимальное количество $K_{\min}$ транзакций каждого типа в одном документе; максимальное количество $K_{\max}$ транзакций каждого типа в одном документе; вероятность p_{ignore} проигнорировать тип транзакции при генерации текущего документа.

Отметим, что в целях упрощения эксперимента все модальности и типы транзакций в моделях используются с весом 1.0. Процесс генерации транзакционных документов, работающий в описанных условиях, представлен в Алгоритме 5.6.1.

Мультимодальные документы для M-ARTM получаются из транзакционных путём удаления информации о транзакционных связях между термами документа. Обычные документы для PLSA получаются из мультимодальных путём удаления для термов информации об их модальности и отнесения их всех к основной модальности (для этого все термы словаря как строки генерируются уникальными в рамках всего словаря, а не только своей модальности).

Метрика качества восстановления параметров. Наличие реальных значений искомых параметров модели позволяет численно оценить качество восстановления, однако есть сложность: матрицы параметров, получающиеся на выходе тематической модели, в общем случае имеют порядок тем, отличный о

Алгоритм 5.6.1. Процедура генерации транзакционных документов

Входные данные: Матрицы Φ и Θ, набор K,
гиперпараметры $|D|, p_{\text{ignore}}, K_{\min}, K_{\max}$;

Выходные данные: коллекция транзакционных документов;

```
1  цикл d = 1, ..., |D| выполнять
2      цикл k ∈ K выполнять
3          сэмплировать p^d_ignore из равномерного распределения;
4          если p^i_ignore > p_ignore тогда
5              перейти к следующему типу транзакции k;
6          сэмплировать N^k_d из равномерного распределения чисел от
           K_min до K_max;
7          цикл j = 1, ..., N^k_d выполнять
8              сэмплировать тему t из распределения θ_d;
9              цикл m ∈ k выполнять
10                 сэмплировать терм m-й модальности из распределения
                   φ^m_t;
11                 добавить полученный терм в текущую транзакцию;
12             добавить полученную транзакцию в d-й документ;
13     добавить d-й документ в коллекцию;
```

порядка в реальных матрицах. Для восстановления этого порядка используется эвристический жадный алгоритм, сопоставляющий друг другу столбцы матриц, имеющие как можно более похожую структуру.

В экспериментах измеряется близость между искомыми и построенными матрицами Φ (только часть, соответствующая основной модальности), Θ и p_{dw} (нормированная матрица «мешка слов» обучающей коллекции n_{dw}). В качестве меры близости используется расстояние Хэллингера, определяемое для веще-

ственнозначных стохастических матриц A и B размера $n \times m$ как:

$$h(A, B) = \frac{1}{m} \sum_{j=1}^{m} \sqrt{\frac{1}{2} \sum_{i=1}^{n} \left(\sqrt{a_{ij}} - \sqrt{b_{ij}}\right)^2}.$$

Для повышения стабильности результата обучение моделей в каждом эксперименте запускается 4 раза, после чего полученные результаты усредняются.

Гиперпараметры и результаты. Во всех экспериментах зафиксирован следующий набор параметров:

- количество проходов по коллекции при обучении: 40;
- количество проходов по документу при обучении: 5;
- количество параллельных потоков: 10;
- размер пакета документов: 50.

Используются следующие фиксированные значения параметров генерации:

- количество модальностей M: 4;
- количество термов в каждой из модальностей:
  $[1000, 200, 100, 100]$ для $m = [1, 2, 3, 4]$;
- количество типов транзакций $|K|$: 4;
- набор номеров модальностей в каждом из типов транзакций:
  $[(1, 2), (1, 3), (1, 2, 3), (2, 3, 4)]$ для $i = [1, 2, 3, 4]$;
- количество значений N_Φ в каждом столбце Φ, умножаемых на C_Φ: 5;
- количество значений N_Θ в каждом столбце Θ, умножаемых на C_Θ: 5;
- значение повышающего множителя C_Φ: 1000;

Параметры			Матрица p_{dw}				Матрица Φ				Матрица Θ			
$\|T\|$	$\|D\|$	$K_{m/m}$	U.	P.	M.	T.	U.	P.	M.	T.	U.	P.	M.	T.
20	200	50/100	0.67	0.42	0.29	**0.28**	0.73	0.48	0.37	**0.36**	0.65	0.49	0.3	**0.23**
		100/200	0.67	0.42	0.27	**0.24**	0.73	0.47	**0.34**	0.35	0.65	0.48	0.3	**0.23**
		250/500	0.68	0.41	0.25	**0.2**	0.75	0.45	**0.3**	**0.3**	0.65	0.5	0.3	**0.22**
	1000	50/100	0.68	0.42	0.26	**0.22**	0.74	0.45	**0.29**	0.3	0.65	0.53	0.3	**0.25**
		100/200	0.67	0.42	0.27	**0.22**	0.74	0.44	0.31	**0.3**	0.65	0.53	0.34	**0.28**
		250/500	0.68	0.41	0.26	**0.17**	0.74	0.42	0.28	**0.22**	0.65	0.52	0.35	**0.24**
	5000	50/100	0.68	0.42	0.25	**0.21**	0.74	0.42	**0.22**	0.24	0.65	0.53	0.31	**0.28**
		100/200	0.67	0.42	0.24	**0.16**	0.74	0.44	0.22	**0.16**	0.65	0.56	0.33	**0.21**
		250/500	0.67	0.41	0.25	**0.16**	0.74	0.41	0.24	**0.16**	0.65	0.54	0.35	**0.24**
100	200	50/100	0.67	0.46	**0.4**	0.5	0.74	0.55	**0.54**	0.63	0.81	0.57	**0.47**	0.6
		100/200	0.67	0.44	**0.32**	0.38	0.74	0.53	**0.43**	0.5	0.81	0.56	**0.39**	0.41
		250/500	0.67	0.45	0.3	**0.28**	0.74	0.54	**0.41**	**0.41**	0.81	0.61	0.42	**0.31**
	1000	50/100	0.67	0.45	**0.34**	**0.34**	0.74	0.5	0.42	**0.4**	0.81	0.62	0.45	**0.39**
		100/200	0.67	0.45	0.32	**0.29**	0.74	0.5	0.41	**0.37**	0.81	0.64	0.46	**0.35**
		250/500	0.67	0.44	0.29	**0.25**	0.74	0.48	0.36	**0.34**	0.81	0.64	0.43	**0.35**
	5000	50/100	0.67	0.45	**0.32**	**0.32**	0.74	0.47	**0.37**	**0.37**	0.81	0.66	0.45	**0.4**
		100/200	0.67	0.44	0.31	**0.27**	0.74	0.46	0.36	**0.32**	0.81	0.64	0.45	**0.37**
		250/500	0.67	0.44	0.3	**0.25**	0.74	0.45	0.34	**0.28**	0.81	0.65	0.45	**0.37**
200	200	50/100	0.67	0.48	**0.41**	0.53	0.74	**0.63**	0.68	0.73	0.85	0.67	**0.63**	0.71
		100/200	0.67	0.46	**0.36**	0.45	0.74	0.61	**0.6**	0.67	0.85	0.63	**0.54**	0.61
		250/500	0.67	0.43	**0.3**	0.33	0.74	0.59	**0.49**	0.55	0.85	0.61	0.45	**0.44**
	1000	50/100	0.67	0.47	**0.35**	0.39	0.74	0.52	**0.43**	0.44	0.85	0.67	0.48	**0.45**
		100/200	0.67	0.46	**0.32**	0.33	0.74	0.53	0.42	**0.41**	0.85	0.67	0.47	**0.41**
		250/500	0.67	0.45	0.3	**0.27**	0.74	0.52	0.4	**0.37**	0.85	0.68	0.48	**0.38**
	5000	50/100	0.67	0.48	**0.35**	0.36	0.74	0.51	0.42	**0.4**	0.85	0.71	0.51	**0.45**
		100/200	0.67	0.47	0.33	**0.31**	0.74	0.51	0.42	**0.36**	0.85	0.7	0.51	**0.41**
		250/500	0.67	0.45	0.31	**0.27**	0.74	0.5	0.39	**0.33**	0.85	0.7	0.5	**0.4**

Таблица 5.9. Значения расстояния Хэллингера между истинными и построенными матрицами распределений для различных моделей. Столбцы: $|T|$ — число тем в модели, $|D|$ — число документов в коллекции, $K_{m/m} = K_{\min}/K_{\max}$ — нижняя и верхняя границы числа транзакций одного типа в документе, U. — равномерные распределения, P. — модель PLSA, M. — модель M-ARTM, T. — модель T-ARTM. Лучшие значения выделены жирным.

- значение повышающего множителя C_Θ: $10 \times |T|$;
- вероятность p_{ignore} проигнорировать тип транзакции при генерации: 0.5.

Значения следующих гиперпараметров в экспериментах перебираются по сетке:

- количество тем $|T|$: $\{20, 100, 200\}$;
- количество документов $|D|$: $\{200, 1000, 5000\}$;
- нижняя и верхняя границы количества транзакций одного типа в документе $K_{\min}$ / $K_{\max}$: $\{50/100, 100/200, 250/500\}$.

В указанной конфигурации данные получаются полноценно транзакционными, что позволяет соответствующим моделям проявить свои предполагаемые свойства.

Таблица 5.9 демонстрирует результаты обучения моделей при различных условиях. Можно видеть, что в большинстве случаев T-ARTM восстанавливает матрицы лучше других методов (особенно в случае Θ). Однако главным выводом является то, что во всех случаях качество восстановления параметров моделью T-ARTM растёт и обходит конкурентов с ростом числа обучающих данных, то есть количества документов и их длины в транзакциях. Это подтверждает гипотезу о том, что T-ARTM лучше адаптируется под данные транзакционной природы и может использовать информацию о связях термов в транзакциях, которая недоступна более простым моделям.

5.7. Основные выводы

M-ARTM. В рамках теории мультимодальных тематических моделей предложена комбинация регуляризаторов, решающая задачу поиска в коллекции текстов информации, заданной неполным списком ключевых слов. Проведены обширные исследования различных тематических моделей в реальной задаче анализа этно-социального дискурса. Демонстрируется превосходство предложенного подхода над всеми прочими как с точки зрения полноты извлекаемой

релевантной информации, так и с точки зрения её интерпретируемости. Предлагаются эмпирические рекомендации по настройке гиперпараметров предлагаемой комбинация. Рассматриваются и оцениваются различные подходы к построению моделей на урезанных данных, которые получаются в результате предварительной фильтрации с помощью моделей, обучаемых на полных данных. В ходе экспериментов, связанных с выделением этнических тем, получен следующий набор основных выводов.

1. Модели с предлагаемой комбинацией регуляризаторов позволяют получить наилучший результат с точки зрения числа релевантных и интерпретируемых тем, при этом в зависимости от размера и качества словаря ключевых слов результат получается различным: более чистый и маленький словарь приводит к уменьшению числа релевантных тем при росте их интерпретируемости (Model-Restricted-Dictionary), более объёмный, но менее качественный словарь приводит к обратному эффекту (Model-Extended-Dictionary).

2. Идея построения модели PLSA на основе данных, обогащённых путём фильтрации по Θ при помощи экспертных оценок релевантности тем, оказалась неудачной.

3. Обучение модели на коллекции, обогащаемой путём фильтрации по ключевым словам приводит к хорошему результату, однако её полнота зависит от размера и качества Q значительно сильнее, чем полнота любой из моделей, обучающихся на всех имеющихся данных.

4. Обогащение коллекции путём фильтрации по наиболее вероятным словам предметных тем при повторном обучении хорошо настроенной модели позволяет улучшить среднюю интерпретируемость тем.

5. Экспериментально подтверждается, что частые обновления модели и многочисленные проходы по каждому документу позволяют онлайновому ал-

горитму ограничиваться одним проходом по данным даже в случае коллекции из нескольких сотен тысяч документов.

6. Совместное использование регуляризаторов сглаживания и декорреляции тем возможно и может давать нужный результат в случае правильно подобранных коэффициентов регуляризации.

7. Регуляризация на основе лог-правдоподобия модальности способна существенно улучшить качество решения внешней задачи с минимальными затратами на подбор её гиперпараметра, в том случае, если эта модальность привносит в модель какую-то информацию, релевантную решаемой задаче.

T-ARTM. В ходе работы на этой частью главы в библиотеке BigARTM реализована поддержка обучения транзакционных аддитивно регуляризованных тематических моделей. Проведена серия экспериментов на синтетических данных по восстановлению параметров распределений, из которых были порождены обучающие текстовые данные. Эмпирически подтверждена гипотеза о том, что гиперграфовая модель T-ARTM лучше использует информацию о транзакционных связях между термами документа, чем модели-предшественники.

Результаты данной главы опубликованы в работах [18, 19].

Заключение

Начало данной работе положила идея о том, что исследователь в области анализа текстов должен получить единый универсальный инструмент для обучения как можно большего числа различных тематических моделей. На момент появления библиотеки BigARTM уже существовало немало реализаций алгоритмов обучения тематических моделей, однако все они, вне зависимости от своей эффективности или архитектуры, предназначены для обучения байесовской модели LDA или её многочисленных, но узконаправленных расширений. Аддитивная регуляризация тематических моделей обобщила значительную часть этих расширений и дала теоретическую базу для создания относительно универсального и гибкого решения. Следующим шагом стала первая версия программной реализации, сразу показавшая неплохую производительность и масштабируемость. Построенная на основе предварительного обзора существующих в научной литературе подходов, она использовала параллельные и асинхронные вычисления, позволяла обучать модели в онлайновом режиме. Однако ключевым её достоинством стало наличие встроенного механизма регуляризации с набором предопределённых регуляризаторов и возможностью добавления новых.

Развитие производительности BigARTM выполнялось итеративно. Прежде всего был предложен и реализован новый алгоритм для онлайнового обучения моделей на больших текстовых коллекциях DetAsync. По сравнению со своим предшественником, он стал быстрее, проще в настройке и устойчивее к плохому набору значений параметров своей работы. Кроме того, он стал детерминированным, что существенно важно для воспроизводимости экспериментов. Следующим шагом стало использование свойства разреженности, часто встречающегося в тематических моделях, особенно с разреживающими регуляризаторами. Было предложено и реализовано обновление ядра библиотеки, которое позволяет в случае разреженных матриц параметров и счётчиков достигать существенного ускорения обучения. В случае алгоритма DetAsync это изменение

дополнительно уменьшает пиковый объём используемой при обучении оперативной памяти.

Теоретический аппарат, лежащий в основе BigARTM, продолжил развитие, одним из его результатов стала возможность игнорирования операции нормировки при подсчёте вспомогательных переменных в EM-алгоритме за счёт использования быстрого E-шага. В работе были предложены и реализованы различные стратегии комбинирования обычных и быстрых E-шагов для оффлайнового и онлайновых алгоритмов, экспериментально определены стратегии, позволяющие и ускорять процесс обучения, и улучшать качество итоговых моделей с точки зрения сходимости и/или интерпретируемости.

Другим следствием развития теории ARTM стало появление её обобщения, модели M-ARTM, которая была реализована в BigARTM вскоре после появления библиотеки. В данной работе эта реализация была использована для экспериментального обоснования эффективности комбинации регуляризаторов, предназначенной для выявления тематик специфической направленности из корпусов текстов. Используя небольшой, подготовленный экспертами словарь, предложенная модель лучше аналогов справляется с извлечением релевантных и интерпретируемых тем. В результате исследования совместно с экспертами-социологами была решена практическая задача анализа этно-социального дискурса.

Модель M-ARTM получила дальнейшее развитие в виде теории гиперграфовых тематических моделей, или T-ARTM, предназначенных для анализа текстовых данных, представленных в виде наборов транзакций. В ходе модернизации BigARTM в рамках данной работы библиотека получила возможность обработки транзакционных данных с помощью моделей T-ARTM. Реализация была использована в эксперименте по восстановлению параметров порождающей модели на синтетических данных, и показала преимущество T-ARTM по сравнению с более простыми моделями при обработке документов, имеющих транзакционную природу.

Говоря о направлениях дальнейших исследований, стоит отметить, что предварительные эксперименты по совмещению разреживающей регуляризации и стратегий смешивания обычных и быстрых E-шагов оказались безуспешными, но детально этот вопрос изучен не был. Другим полезным продолжением работы является более тщательное изучение в задаче поиска специфических тематик подхода с повторным построением модели на обогащённых по матрице Φ данных. В частности, стоит провести эксперименты по подбору оптимального числа тем, используя экспертные оценки релевантности и значения tf-idf когерентности. Наконец, открытым остаётся вопрос об использовании различных подходов к разреженной инициализации тематической модели с целью ещё большей минимизации пикового потребления оперативной памяти алгоритмами без существенных потерь качества.

Список литературы

1. *T. Hoffmann.* Probabilistic latent semantic indexing // Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. — New York, NY, USA: ACM, 1999. — Pp. 50–57.
2. *T. Hoffmann.* Probabilistic latent semantic analysis // Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, – UAI'99. — San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. — Pp. 289–296.
3. *D. Blei, A. Ng, M. Jordan.* Latent Dirichlet allocation // *Journal of Machine Learning Research.* — 2003. — Vol. 3. — Pp. 993–1022.
4. *M. Steyvers, T. Griffiths.* Finding scientific topics // Proceedings of the National Academy of Sciences. — Vol. 101. — 2004. — Pp. 5228–5235.
5. *K. Vorontsov.* Additive regularization for topic models of text collections // *Doklady Mathematics.* — 2014. — Vol. 89, no. 3. — Pp. 301–304.
6. Statistical topic models for multi-label document classification / Rubin T., Chambers A., Smyth P., Steyvers M. // *Machine Learning.* — 2012. — Vol. 88, no. 1-2. — Pp. 157–208.
7. *А. Янина, К. Воронцов.* Мультимодальные тематические модели для разведочного поиска в коллективном блоге // *Машинное обучение и анализ данных.* — 2016. — Vol. 2, no. 2. — Pp. 173–186.
8. Improving summarization quality with topic modeling / Litvak M., Vanetik N., Liu C. et al. // Proceedings of the 2015 Workshop on Topic Models: Post-Processing and Applications. — New York, NY, USA: ACM, 2015. — Pp. 39–47.
9. Evolutionary hierarchical Dirichlet processes for multiple correlated time-varying corpora / Zhang J., Song Y., Zhang C., Liu S. // Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. — 2010. — Pp. 1079–1088.
10. *W. Cui, S. Liu, L. Tan et al.* TextFlow: Towards Better Understanding of Evolving Topics in Text // *Transactions on Visualization and Computer Graphics.*

— 2011. — Vol. 17, no. 12. — Pp. 2412–2421.

11. Mining multilingual topics from wikipedia / Ni X., Sun J., Hu J., Chen Z. // In Proceedings of the 18th International Conference on World Wide Web, WWW '09. — 2009. — Pp. 1155–1156.
12. Polylingual topic models / Mimno D., Wallach H., Naradowsky J. et al. // In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: EMNLP '09. — Vol. 2. — 2009. — Pp. 880–889.
13. *W. Smet, M. Moens.* Cross-language linking of news stories on the web using interlingual topic modelling // In Proceedings of the 2Nd ACM Workshop on Social Web Search and Mining, SWSM '09. — New York, NY, USA: ACM, 2009. — Pp. 57–64.
14. Non-bayesian additive regularization for multimodal topic modeling of large collections / Vorontsov K., Frei O., Apishev M. et al. // Proceedings of the CIKM 2015 Workshop on Topic Models: Post-Processing and Applications. — New York, NY, USA: ACM, 2015. — Pp. 29–37.
15. Библиотека для тематического моделирования BigARTM. — URL: `https://bigartm.org`.
16. BigARTM: Open Source Library for Regularized Multimodal Topic Modeling of Large Collections / Vorontsov K., Frei O., Apishev M. et al. // AIST'2015, Analysis of Images, Social networks and Texts. Communications in Computer and Information Science (CCIS). — Switzerland: Springer International Publishing, 2015. — Pp. 370–384.
17. *O. Frei, M. Apishev.* Parallel non-blocking deterministic algorithm for online topic modeling // AIST'2016, Analysis of Images, Social networks and Texts. Communications in Computer and Information Science (CCIS). — Vol. 661. — Switzerland: Springer International Publishing, 2016. — Pp. 132–144.
18. Mining Ethnic Content Online with Additively Regularized Topic Models / Apishev M., Koltcov S., Koltsova O. et al. // *Computación y Sistemas.* — 2016. — Vol. 20, no. 3. — Pp. 387–403.

19. Additive Regularization for Topic Modeling in Sociological Studies of User-Generated Texts / Apishev M., Koltcov S., Koltsova O. et al. // Advances in Computational Intelligence, 15th Mexican International Conference on Artificial Intelligence, MICAI 2016. Proceedings, Part I. Lecture Notes in Artificial Intelligence. — Vol. 10061. — Cancún, Quintana Roo, Mexico: 2016. — Pp. 166–181.
20. Fast and modular regularized topic modelling / Kochedykov D., Apishev M., Golitsyn L., Vorontsov K. // Proceeding Of The 21St Conference Of FRUCT Association. The seminar on Intelligence, Social Media and Web (ISMW). — 2017. — Pp. 182–193.
21. *M. Apishev, K. Vorontsov.* Learning Topic Models With Arbitrary Loss // Proceedings of the 26th Conference of FRUCT Association. — 2020. — Pp. 30–37.
22. *М. Апишев.* Эффективные реализации алгоритмов тематического моделирования // *Труды ИСП РАН.* — 2020. — Vol. 32, no. 1. — Pp. 137–152.
23. *М. Апишев.* Реализация мультимодальных регуляризованных тематических моделей в библиотеке с открытым кодом BigARTM // Сборник тезисов XXII Международной научной конференции студентов, аспирантов и молодых учёных «Ломоносов-2015», секция «Вычислительная математика и кибернетика». — Москва: МАКС Пресс, 2015. — Pp. 91–92.
24. *М. Апишев.* Аддитивная регуляризация тематических моделей в задаче анализа этносоциального дискурса // Сборник тезисов XXIII Международной научной конференции студентов, аспирантов и молодых учёных «Ломоносов-2016», секция «Вычислительная математика и кибернетика». — Москва: МАКС Пресс, 2016. — Pp. 117–119.
25. *И. Жариков, М. Апишев, К. Воронцов.* Гиперграфовые многомодальные вероятностные тематические модели транзакционных данных // Интеллектуализация обработки информации (ИОИ-2018): Тезисы докл. — Москва: Торус Пресс, 2018. — Pp. 148–149.
26. *Y. Xing, A. James.* A Comparative Study of Utilizing Topic Models for Infor-

mation Retrieval // *Advances in Information Retrieval.* — 2009. — Vol. 5478. — Pp. 29–41.

27. *I. Vulic, W. Smet, Moens M.* Cross-language information retrieval models based on latent topic models trained with document-aligned comparable corpora // *Information Retrieval.* — 2012. — Pp. 1–38.
28. *E. Zavitsanos, G. Paliouras G. Vouros.* Non-Parametric Estimation of Topic Hierarchies from Texts with Hierarchical Dirichlet Processes // *Journal of Machine Learning Research.* — 2011. — Vol. 12. — Pp. 2749–2775.
29. *J. Shoaib, L. Wai.* An N-Gram Topic Model for Time-Stamped Documents // 35th European Conference on Information Retrieval, ECIR-2013, Lecture Notes in Computer Science (LNCS). — Springer, 2013. — Pp. 292–304.
30. Knowledge discovery through directed probabilistic topic models: a survey / Daud A., Li J., Zhou L., Muhammad F. // *Frontiers of Computer Science in China.* — 2010. — Vol. 4, no. 2. — Pp. 280–301.
31. *K. Vorontsov, A. Potapenko.* Additive regularization of topic models // *Machine Learning, Special Issue on Data Analysis and Intelligent Optimization with Applications.* — 2015. — Vol. 101, no. 1. — Pp. 303–323.
32. *A. Dempster, N. Laird, D. Rubin.* Maximum likelihood from incomplete data via the EM algorithm // *J. of the Royal Statistical Society, Series B.* — 1977. — no. 34. — Pp. 1–38.
33. *A. Tikhonov, V. Arsenin.* Solution of ill-posed problems. — 1977.
34. *Y. Tan, Z. Ou.* Topic-weak-correlated latent Dirichlet allocation // 7th International Symposium Chinese Spoken Language Processing (ISCSLP). — 2010. — Pp. 224–228.
35. *Y. Teh, D. Newman, M. Welling.* A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation // NIPS. — 2006. — Pp. 1353–1360.
36. *K. Vorontsov, A. Potapenko.* EM-like algorithms for probabilistic topic modeling // *Machine Learning and Data Analysis.* — 2013. — Vol. 1, no. 6. — Pp. 657–686.

37. On Smoothing and Inference for Topic Models / Asuncion A., Wekking M., Smyth P., Teh Y. // Proceedings of the International Conference on Uncertainty in Artificial Intelligence. — 2009. — Pp. 27–34.
38. Distributed algorithms for topic models / Newman D., Asuncion A., Smyth P., Welling M. // *The Journal of Machine Learning Research*. — 2009. — Vol. 10. — Pp. 1801–1828.
39. PLDA: Parallel latent Dirichlet allocation for large-scale applications / Wang Y., Bai H., Stanton M. et al. // Proceedings of the 5th International Conference on Algorithmic Aspects in Information and Management. — 2009. — Pp. 301–314.
40. *A. Asuncion, P. Smyth, M. Welling.* Asynchronous distributed estimation of topic models for document analysis // *Statistical Methodology*. — 2010. — Vol. 8, no. 1. — Pp. 3–17.
41. PLDA+: Parallel latent dirichlet allocation with data placement and pipeline processing / Liu Z., Zhang Y., Chang E., Sun M. // *Transactions on Intelligent Systems and Technology*. — 2011. — Vol. 2, Issue 3, no. 26.
42. *A. Smola, S. Narayanamurthy.* An architecture for parallel topic models // Proceedings of the VLDB Endowment. — Vol. 3, Issue 1-2. — 2010. — Pp. 703–710.
43. *R. Rehurek, P. Sojka.* Software framework for topic modeling with large corpora // Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. — 2010. — Pp. 45–50.
44. Mr. LDA: a flexible large scale topic modeling package using variational inference in MapReduce / Zhai K., Boyd-Graber J., Asadi N., Alkhouja M. // Proceedings of the 21st international conference on World Wide Web. — 2012. — Pp. 879–888.
45. Gibbs collapsed sampling for latent Dirichlet allocation on Spark / Qiu Z., Wu B., Wang B., Yu L. // Proceedings of the 3rd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications. — Vol. 36. — 2014. — Pp. 17–28.
46. Peacock: learning long-tail topic features for industrial applications / Wang Y., Zhao X., Sun Z. et al. // *Transactions on Intelligent Systems and Technology*

(TIST) — Regular Papers and Special Section on Intelligent Healthcare Informatics. — 2014. — Vol. 9, no. 4, Article 39.

47. LightLDA: Big topic models on modest computer clusters / Yuan J., Gao F., Ho Q. et al. // WWW. — 2015. — Pp. 1351–1361.
48. ZenLDA: Large-scale topic model training on distributed data-parallel platform / Zhao B., Zhou H., Li G., Huang Y. // *Big Data Mining and Analytics.* — 2018. — Vol. 1, Issue 1. — Pp. 57–74.
49. *J. Zeng, Z. Liu, Cao X.* Fast online EM for big topic modeling // *Transactions on Knowledge and Data Engineering.* — 2016. — Vol. 28, Issue 3. — Pp. 675–688.
50. *M. Hoffmann, D. Blei, F. Bach.* Online learning for latent Dirichlet alocation // Proceedings of the 23rd International Conference on Neural Information Processing Systems. — Vol. 1. — 2010. — Pp. 856–864.
51. Библиотека для машинного обучения Vowpal Wabbit. — URL: `https://github.com/JohnLangford/vowpal_wabbit`.
52. *T. White.* Hadoop: The definitive guide. — O'Reilly Media, Inc., 2012.
53. *M. Zaharia, M. Chowdhury, M. Franklin.* Spark: cluster computing with working sets // HotCloud'10 Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing. — 2010. — P. 10.
54. Forum Message Passing Interface. MPI: A Message-Passing Interface Standard, Version 3.0 // ACM SIGOPS operating systems review / High Performance Computing Center, Stuttgart (HLRS). — ASM, 2012.
55. Система распределённого хранения данных Memcached. — URL: `https://memcached.org`.
56. *J. Dean, S. Ghemawat.* MapReduce: simplified data processing on large clusters // OSDI'04. — 2004.
57. Платформа для реализации алгоритмов машинного обучения Petuum. — URL: `http://www.petuum.com`.
58. Библиотека GraphX для работы с графами в фреймворке Spark. — URL: `https://spark.apache.org/graphx`.

59. *B. Nelson.* Remote Procedure Call // PARC CSL-81-9, Xerox Palo Alto Research Center, PhD thesis. — 1981.
60. SaberLDA: Sparsity-Aware Learning of Topic Models on GPUs / Li K., Chen J., Chen W., Zhu J. // Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems.
61. Библиотека для описания и сериализации структурированных данных. — URL: `https://developers.google.com/protocol-buffers`.
62. Коллекция статей New York Times. — URL: `https://archive.ics.uci.edu/ml/datasets/Bag+of+Words`.
63. Коллекция статей англоязычной Википедии. — URL: `https://s3-eu-west-1.amazonaws.com/artm/vw.wiki-en.txt.zip`.
64. Коллекция аннотаций медицинских статей Pubmed. — URL: `https://s3-eu-west-1.amazonaws.com/artm/docword.pubmed.txt.gz\(vocab.pubmed.txt)`.
65. Реализация и объяснение CSR-формата в библиотеке SciPy. — URL: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html`.
66. Evaluating topic models for digital libraries / Newman D., Noh Y., Talley E. et al. // Proceedings of the 10th annual Joint Conference on Digital libraries ser. JCDL '10. — New York, NY, USA: ACM, 2010. — Pp. 215–224.
67. Optimizing semantic coherence in topic models / Mimno D., Wallach H., Talley E. et al. // Proc. EMNLP'11. — 2011. — Pp. 262–272.
68. *D. Blei, M. Jordan.* Modeling annotated data // In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval. — New York, NY, USA: ACM, 2003. — Pp. 127–134.
69. *D. Cohn, T. Hofmann.* The missing link – a probabilistic model of document content and hypertext connectivity // In NIPS. — 2000. — Pp. 430–436.
70. *D. Newman, C. Chemudugunta, P. Smyth.* Statistical entity-topic models // In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge

Discovery and Data Mining, KDD '06. — New York, NY, USA: ACM, 2006. — Pp. 680–686.

71. *X. Si, M. Sun.* Tag-lda for scalable real-time tag recommendation // *Journal of Information & Computational Science.* — 2009. — no. 6. — Pp. 23–31.
72. *L. Dietz, S. Bickel, T. Scheffer.* Unsupervised prediction of citation influences // In Proceedings of the 24th international conference on Machine learning, ICML '07. — New York, NY, USA: 2007. — Pp. 233–240.
73. *J. Jagarlamudi, H. Daume, R. Udupa.* Incorporating lexical priors into topic models // In: Proc. EACL'12. — 2012. — Pp. 204–213.
74. *M. Paul, M. Dredze.* Discovering health topics in social media using topic models // *PLoS ONE.* — 2014. — Vol. 9, no. 8.
75. Interval semi-supervised LDA: Classifying needles in a haystack / Bodrunova S., Koltsov S., Koltsova O. et al. // Proc. MICAI 2013, LNCS. — Vol. 8625. — Springer, 2013. — Pp. 265–274.
76. *S. Nikolenko, O. Koltsova, S. Koltsov.* Topic modelling for qualitative studies // *Journal of Information Science.* — 2015.
77. *C. Chemudugunta, P. Smyth, M. Steyvers.* Modeling general and specific aspects of documents with a probabilistic topic model // *Advances in Neural Information Processing Systems.* — 2007. — Vol. 19. — Pp. 241–248.