Санкт-Петербургский государственный университет
Факультет прикладной математики – процессов управления
Кафедра технологии программирования

Рахимзянов Данис Ильшатович

Выпускная квалификационная работа бакалавра

Анализ вероятностной модели «Joint Sentiment-Topic Model» в задачах классификации текста по тональности

Направление 010400

Информационные технологии

Заведующий кафедрой,
кандидат физ.-мат. наук,
доцент Сергеев С. Л.

Научный руководитель,
кандидат физ.-мат. наук,
доцент Добрынин Ю. В.

Рецензент,
кандидат физ.-мат. наук,
доцент Лепихин Т. А.

Санкт-Петербург
2014

Оглавление

Введение ... 3
Постановка задачи ... 5
Обзор литературы ... 6
Глава 1. Подготовка входных данных ... 7
1.1. Парсинг англоязычной Википедии ... 8
1.2. Создание графа Википедии ... 9
1.3. Выборка коллекции документов из графа ... 10
1.4. Подготовка словаря SentiWordNet ... 12
Глава 2. Краткое описание модели JST ... 15
Глава 3. Эксперименты ... 18
3.1. Инициализация ... 18
3.2. Экспертная оценка качества классификации тем ... 19
3.3. Оценка качества классификации слов по тональности ... 24
3.4. Named-Entities Recognition ... 27
3.5. Автоматизация процесса создания словарей ... 31
Глава 4. Модификация реализации алгоритма JST ... 34
4.1. Параллельные вычисления ... 34
4.2. Использование стемминга при кодировании словаря ... 38
Выводы ... 40
Заключение ... 41
Список литературы ... 42

Введение

В современном динамично меняющемся мире человеку становится чрезвычайно важно владеть оперативной информацией независимо от вида его деятельности. В эпоху высоких технологий и Интернета, когда не выходя из дома можно купить практически всё, начиная продуктами питания и заканчивая личным бизнес-джетом, компаниям и корпорациям становится всё более важно в процессе своей инновационной деятельности знать мнение своих потребителей. Не даром одна из целей любого бизнеса – восхищать потребителя. В то время как мнения и отзывы, а также посты в блогах и социальных сетях о какой-либо продукции или услуге можно свободно получить из глобальной Сети, то встаёт вопрос об оценке этих мнений и любой текстовой информации, полученной тем или иным образом от потребителя. А учитывая объёмы этих данных становится очевидным, что единственным способом обработки этой информации остаётся машинный. Говоря об анализе данных, стоит ограничить фронт возможных действий. Понятно, что производители не в состоянии прочитать и оценить каждый отзыв на каждом сайте, поэтому на передний план выходят методы и инструменты машинного обучения, теории вероятности и математической статистики. Рассмотрим следующий индикатор – эмоциональная окраска текста. Естественно, что компании и корпорации интересуются в первую очередь отрицательными отзывами для того, чтобы лучше понять потребителя, сделать работу над ошибками и попробовать его восхитить улучшенной продукцией либо услугой. Таким образом, нас интересуют методы автоматической оценки тональности текстов. И пока не изменится способ выражения мнений и эмоций человека, эта проблема будет оставаться актуальной.

Анализ тональности текста – одно из направлений компьютерной лингвистики, достаточно хорошо исследованное. Здесь выделяются различные методы: машинное обучение с учителем и без него; подходы, основанные на правилах или словарях. Одновременно с вопросом оценки

качества и точности результата применения всех этих методов возникает также проблема непереносимости инициирующих данных и параметров (словари, правила и обучающие выборки) на другую предметную область (домен).

Наше внимание привлёк метод, который можно отнести к машинному обучению без учителя, и который опирается на элементы теории вероятностей и математической статистики, а именно вероятностная тематическая модель «Joint Sentiment-Topic Model», предложенная Ченгуа Лином (Chenghua Lin, University of Exeter) и Юланом Хэ (Yulan He, The Open University) в статьях «Joint Sentiment/Topic Model for Sentiment Analysis» [1] и «Weakly-supervised joint sentiment-topic detection from text» [2]. Отличительной особенностью данного метода являются отсутствие учителя (хотя далее мы увидим, что некоторые элементы обучения с учителем в данном методе всё-таки присутствуют), потенциальная независимость от предметной области, а также одновременная классификация текста по тональности и теме. Стоит отметить, что данный метод основывается на генеративной вероятностной тематической модели Latent Dirichlet Allocation (LDA, скрытое размещение Дирихле) из одноимённой статьи [3], где главенствующим является тезис, что каждое слово в тексте из некоторой коллекции принадлежит некоторой теме, и таким образом, документ является порождением из одной или нескольких тем. В обзоре литературы мы остановимся на этом подробнее.

Если не ограничивать себя только областью оценки мнений о продукции и услугах, то можно рассмотреть приложение вероятностной тематической модели JST к набирающему в последнее время популярность направлению «Named-Entity Recognition» (NER), где объектами исследования являются имена собственные, названия городов, стран и организаций. Начнём обо всё по порядку.

Постановка задачи

Учитывая актуальность проблемы анализа тональности текста во время всемирной глобализации и растущих объёмов данных, были поставлены следующие задачи.

Во-первых, исследовать вероятностную тематическую модель «Joint Sentiment-Topic Model» (далее – JST) на коллекции документов из англоязычной Википедии, а именно на теме «Политические выборы и Европейский Союз» используя экспертную оценку в качестве критерия оценки качества классификации слов по темам, а в качестве оценки классификации слов по тональностям – словарь SentiWordNet [4]. В качестве переменных, которыми можно оперировать, использовать количество тем, количество итераций сэмплирования по Гиббсу и различные варианты инициирующего словаря.

Во-вторых, рассмотреть приложение результатов работы алгоритма для выявления появления различных имён собственных (политических деятелей, названий стран, городов и компаний), входящих в коллекцию, в рамках различных тематических и тональных контекстов.

В-третьих, необходимо рассмотреть возможность модификации реализации алгоритма JST для уменьшения времени вычислений, а также рассмотреть влияние использования стемминга при инициализации алгоритма на качество классификации слов по темам и тональности.

Обзор литературы

Для повышения знаний в области информационного поиска важным источником является книга Маннинга К. «Введение в информационный поиск» [5]. Особенно полезными оказались глава 13 «Классификация текстов и наивный байесовский подход» и глава 14 «Классификация в векторном пространстве». Для понимания вероятностного тематического моделирования чрезвычайно полезными были лекции Воронцова К. В. [6]. Исследуемая вероятностная тематическая модель базируется на опубликованной Дэвидом Блеем (David Blei) в 2003 году статье «Latent Dirichlet Allocation». Основная идея предложенной им модели – представление коллекции документов в виде «мешка слов», то есть порядок слов в документе и в коллекции не играет роли. Далее строится предположение, что каждое слово в документе порождено некоторой темой, к которой оно относится. При этом каждый документ является порождением из нескольких тем. И выдвигается гипотеза, что распределение вероятностей принадлежности тем к документам является случайным вектором из распределения Дирихле с параметром α. В технической статье «Parameter estimation for text analysis» [7] описывается весь математический фундамент модели LDA. В 2009 и 2012 годах в [1] и [2] было предложено расширение генеративной модели LDA для одновременной классификации текстов по тональности путём добавления нового «слоя». В презентации «Named Entity Recognition and the Stanford NER Software» [8] описан принцип работы классификатора Stanford Named Entity Recognizer, который мы использовали для выявления именованных сущностей. В книге Седжвика Р. «Алгоритмы на графах» [9] даны основы теории графов и приведены основные алгоритмы. В циклах статей от Intel [10] описывается функционал и подходы к применению библиотеки OpenMP для проведения параллельных вычислений на C++. На Интернет-странице [11] приведено описание работы алгоритма Портера для стемминга слов.

Глава 1. Подготовка входных данных

Подготовка коллекции документов оказалась не самой тривиальной задачей. Есть два источника, откуда можно взять дамп Википедии: сайт WikiMedia Foundation и проект DBpedia, в котором структурированная информация из Википедии извлекается посредством языка SPARQL. Было решено использовать первый источник, в силу нескольких факторов. Во-первых, на основном сайте находятся самые актуальные дампы, в то время как в DBPedia может потребоваться процесс обновления данных. Во-вторых, это избыточность функциональности второго проекта и его инфраструктуры. Исходя из условия, что создание коллекции – это всего лишь часть данной работы, было решено своими силами извлечь необходимые данные из оригинального дампа Википедии. Для этого на языке Java было написано несколько вспомогательных программ. Можно разбить процесс создания коллекции документов на следующие логические шаги:

a) Парсинг и сохранение текста статей в базу данных;
b) Создание графа на основе имеющихся связей между статьями;
c) Получение коллекции документов из некоторой категории;

В качестве рабочей машины использовался ноутбук Sony Vaio с процессором Intel Core i5 @ 2,4Ghz и шестью гигабайтами оперативной памяти. Как выяснилось в процесс работы с большими данными, узким местом становится скорость чтения и записи на внешнюю память компьютера. Поэтому также дополнительно был приобретён SSD-накопитель. В качестве среды разработки использовались Microsoft Visual Studio 2013 для языка C# и IntelliJ IDEA Community Version 13.1.1 совместно со сборщиком проектов Apache Maven для языка Java. Для работы с базой данных использовались Connector/NET и Connector/J с официального сайта MySQL, для обоих языков соответственно.

Далее опишем каждый этап в отдельном параграфе.

1.1. Парсинг англоязычной Википедии

В качестве текстовой коллекции было решено использовать категорию из англоязычной Википедии. Необходим был механизм, посредством которого можно было бы извлекать различные коллекции в будущем, для проведения исследований в различных предметных областях.

Актуальные дампы можно скачать в Интернете по адресу www.dumps.wikimedia.org/enwiki/latest/. Последние версии статей находятся в архиве `enwiki-latest-pages-articles.xml.bz2`, после распаковки которого мы получаем один xml-файл размером 45 гигабайт. Данный файл содержит как интересующие нас статьи, так и лишнюю информацию. Нас интересуют только следующие параметры: идентификатор статьи (ID), заголовок статьи (TITLE), основной текст (TEXT), тип страницы (NAMESPACE), и ссылается ли данная статья на другую (REDIRECT). На языке Java был написан SAX-парсер, посредством которого исходный файл сохраняется в базу данных. SAX-парсер является событийно-ориентированной абстракцией, предоставляющей интерфейс для поточной обработки данных. Реализация интерфейса представлялась следующим образом: инициализируются булевы флаги, сигнализирующие о том, что в данный момент открыт интересующий нас тег. С помощью данных флагов и событий, возникающих при поточном чтении входного файла, мы можем получить, обработать и сохранить интересующие нас текстовую информацию в базу данных.

Стоит также отметить тот факт, что Википедия содержит различные типы страниц: обсуждения, профили пользователей, шаблоны, страницы помощи, страницы медиафайлов и так далее. Всего имеется 28 различных типа страниц, и он определяется параметром NAMESPACE. Для создания коллекции нам нужны только статьи и категории, их идентификаторы типа равны 0 и 14, соответственно. Таким образом, можно не сохранять все остальные типы страниц в базу данных, тем самым уменьшив время обработки

дампа Википедии и сохранив дополнительное свободное место на внешней памяти компьютера.

Следующим важным моментом является тот факт, что исходный код страниц Википедии хранится в виде wiki-разметки. Но для нашего исследования нам необходим чистый текст. Было проверено два подхода:

a) Конвертировать wiki-текст в HTML, и далее в обычный текст без разметки посредством библиотеки `org.eclipse.mylyn`;
b) Использовать методы из API проекта DBPedia, а точнее из библиотеки `org.dbpedia.extraction.wikiparser`;

Качество второго подхода оказалось заметно лучше, и первый способ был отклонён. Отметим, что в процессе парсинга не сохранялись пустые страницы и страницы с невалидной wiki-разметкой.

Что касается хранения данных, принципиальной разницы в выборе СУБД не существует, и нами был использован MySQL Server. Стоит отметить, что Википедия работает на CMS MediaWiki и она использует также данную СУБД. На сайте, где располагаются дампы, можно найти множество дополнительных файлов: связи категорий и страниц, связи редиректов и прочие метаданные. Также существует возможность создания полной локальной версии Википедии, но для этого понадобится быстрая внешняя память и увеличенные настройки кэшей СУБД. Но это не входит в рамки нашего исследования.

1.2. Создание графа Википедии

Чтобы иметь возможность получить список статей определённой тематики, нам необходимо создать граф связей страниц и категорий Википедии. Стоит отметить, что существуют open-source решения данной задачи, но в них не удалялись сильные компоненты связности. Алгоритм действий выглядел следующим образом:

a) Строится транзитивное замыкание перенаправлений страниц;

b) Анализируются внутренние ссылки на каждой странице и строится ориентированный граф (например, посредством пакета WebGraph для языка Java);

Полученный граф содержит большое количество сильных компонент в подграфе категорий (выбирать предметную область мы собираемся на основе категории из Википедии). Для удачного выделения статей из некоторой предметной области нужно избавиться от компонент сильной связности в подграфе категорий:

1. Запускаем поиск в ширину из категории (Category:Contents). Для каждой вершины запоминаем кратчайшее расстояние из этой категории;
2. Используем алгоритм Тарьяна для поиска в графе категорий компонент сильной связности;
3. Удаляем в каждой компоненте сильной связности одно ребро (from, to), для которого кратчайшее расстояние для from минус кратчайшее расстояние для to принимает максимальное значение;
4. Повторяем второй пункт до тех пор, пока не избавимся от компонент сильной связности;

Таким образом, после получения и обработки графа, у нас есть возможность создания коллекции документов для изучения работы модели JST на них.

1.3. Выборка коллекции документов из графа

Для нашего исследования была выбрана тема «Политические выборы и Европейский Союз». Создание коллекции включает следующие шаги:

a) Используя поиск в ширину выделить категории и статьи, относящиеся к теме «Выборы». На выходе получаем список заголовков статей;
b) Выбрать по заголовкам статьи из базы данных, используя при этом полнотекстовый поиск вхождения фразы «European Union» в статью;

c) Сохранить выбранную коллекцию в формат, необходимый для инициализации алгоритма;

Что касается первого пункта, то действуем мы следующим образом. Запускаем алгоритм поиска в ширину из главной категории «Category:Main topic classifications» и интересующей нас категории «Category:Elections». Если из последней категории мы доходим до какой-либо категории быстрее, чем из главной, то считаем найденную категорию «подходящей» и считаем все статьи из неё также «подходящими». «Дойти быстрее» в данном контексте значит, что путь из категории «Category:Elections» до найденной категории меньше, чем из корневой категории. На втором этапе проходим по всем полученным заголовкам и сохраняем для удобства все статьи в отдельную таблицу «Elections». Далее необходимо, чтобы текст статей был проиндексирован для полнотекстового поиска. В MySQL команда, для создания такого индекса выглядит следующим образом:

```
ALTER TABLE `elections` ADD FULLTEXT INDEX(`text`);
```

Для успешного регистронезависимого полнотекстового поиска необходимо, чтобы текст хранился в кодировке «utf8_general_ci». Выборка статей, с входящей в них фразой «European Union» выглядит следующим образом:

```
SELECT * FROM `elections` WHERE MATCH(`text`) AGAINST('"european union"' IN BOOLEAN MODE);
```

Некоторая статистика касательно количества статей приведена в следующей таблице:

Таблица 1. Количество статей в Википедии

Общее количество страниц в Википедии	**Количество статей и категорий в Википедии**	**Тема «Политические выборы»**	**Статьи, содержащие фразу «Европейский союз»**
11 869 428	5 615 384	48 499	836

На следующем этапе нам нужно сохранить коллекцию в формат, пригодный для работы алгоритма. Выглядит результирующий файл следующим образом:

[d0] [word1] [word2] … [wordN$_{d0}$]

…

[dN] [word] [word2] …[wordN$_{dN}$]

Но перед этим необходимо проделать следующие технические действия. Во-первых, посредством регулярных выражений удалить из текста все числа, переводы строк и, оставшиеся после конвертации Wiki-разметки в обычный текст, HTML-теги. Во-вторых, перевести весь текст в нижний регистр. В-третьих, удалить из текста статей стоп-слова. Далее по усмотрению можно провести стемминг, например, использовать алгоритм Портера [11]. Но применять стемминг не удобно по ряду причин:

1. Некоторые однокоренные слова в английском (впрочем, как и русском) языке имеют разные тональности в словаре SentiWordNet;
2. В силу того, что слова нормализуются отсечением окончаний, будет затруднена экспертная оценка классификации слов алгоритмом;

После проведения вышеперечисленных действий, сохраняем файл коллекции в формате, указанном ранее, и приступаем к следующему этапу – подготовке инициирующего словаря.

1.4. Подготовка словаря SentiWordNet

Для инициализации алгоритма необходимо задать некоторое количество слов, которые имеют достоверную эмоциональную оценку. Наше внимание привлёк словарь SentiWordNet (далее – SWN), который базируется на словаре WordNet – тезаурусе для английского языка. В SWN каждому синонимическому ряду из WordNet даётся эмоциональная оценка по следующей шкале: позитивное, негативное и нейтральное значения; а также приводится краткий контекст в котором употребляются слова из

синонимического ряда и/или поясняется значение слова. Стоит отметить, что количество этих примеров невелико и не применимо к узким предметным областям. Поэтому нужно было исследовать потенциальную возможность автоматического составления нового словаря, в котором бы в зависимости от предметной области выделялись контексты использования того или иного слова. В нашем исследовании, как будет видно позже, из исходной коллекции выбирались Named Entities (именованные сущности), для которых впоследствии выделялись контексты использования этих слов из результатов работы основного алгоритма и объекта исследования нашей работы – JST. Но обо всём по порядку.

Рассмотрим формат словаря SWN. На каждой строчке располагается синонимический ряд, в начале строки указывается часть речи (POS), далее – идентификатор ряда из WordNet (ID), позитивная и негативная оценка (PosScore и NegScore, соответственно), синонимический ряд (SynsetTerms), с указанным через символ «#» рангом каждого слова, и контекст (Gloss). Нейтральная оценка высчитывается следующим образом:

$$ObjScore = 1 - (PosScore + NegScore).$$

Для удобства работы со словарём было решено сохранить его в базу данных. При обработке SWN поступали следующим образом. Во-первых, выбирались слова с рангом равным единице, это означает, что оценка синонимического ряда для этого слова в этой части речи первична. Во-вторых, не сохранялись слова, содержащие цифры, нижнее подчёркивание, дефис, апостроф, точку, а также слова, длина которых меньше двух символов. Далее было решено составить две версии инициирующего словаря из SWN.

Первый подход заключался в следующем. Из сохранённого в базу данных словаря SWN выбирались слова с наиболее определённой тональностью, а именно такие, у которых отсутствовала нейтральная оценка, а позитивная или негативная оценка превышала значение 0.5 и не равна единице. Из полученной выборки вручную выбиралось по 25 слов каждой

тональности, которые в той или иной мере могут быть ассоциированы с тематикой политических выборов. Список слов, образующих первый инициирующий словарь, приведён в следующей таблице.

Таблица 2. Первая версия инициирующего словаря

Позитивная тональность	**Негативная тональность**
unemotional, delicate, innocence, awesome, morality, pretty, incorrupt, unprintable, safeness, unpainful, indetermination, implacable, fabulous, justness, vice, recovered, heal, delectable, attractive, playful, disadvantageous, straightarrow, stirring, comradeship, extremeness	panic, foul, terror, loathly, insidious, degenerate, insufficient, heartless, imitation, incorruptibility, naturalistic, poor, lewd, illicit, pitiless, corruptive, fresh, heinous, enthusiasm, shifty, worst, pathos, unexchangeable, approximate, humility

Но фактически формат был следующим:

`word\tposScore\tnegScore\r\n`

Стоит отметить, что слов с наиболее определённой тональностью в SWN не так уж и много. Это факт замечательно коррелирует и с реальной жизнью: назвать несколько десятков сугубо положительных или отрицательных слов будет непростой задачей, как может показаться на первый взгляд. В связи с этим был подготовлен расширенный вариант словаря. Были выбраны все слова с выраженной положительной и отрицательной оценками и с нулевой нейтральной оценкой. Выборка включала 204 слова. Она была разбита на две части: тренировочную и тестовую.

Перейдём к краткому описанию вероятностной тематической модели JST. Наиболее полное описание можно найти в оригинальных статьях [1] и [2].

Глава 2. Краткое описание модели JST

Положим, что у нас есть коллекция документов:

$$C = \{d_1, d_2, \dots, d_D\}.$$

Каждый документ представляет собой последовательность слов, порядок которой, как мы уже отмечали ранее, не важен:

$$d = \{w_1, w_2, \dots, w_{N_d}\}.$$

А каждое слово в документе является позицией из общего словаря с индексом $i \in \{1 \dots V\}$. Обозначим общее количество тональностей переменной S, а общее количество предполагаемых тем – T. Мы будем предполагать бинарную классификацию, проще поддающуюся оценке.

Далее процесс «вывода» слова w_i сводится к следующим шагам:

a) выбрать метку тональности l из распределения тональности π_d, определённого для каждого документа;
b) выбрать тематическую метку из распределения тем $\theta_{d,l}$, зависящей от ранее выбранной метки тональности;
c) вывод слова из распределения слов, зависящего от выбранных меток тональности и тематической метки;

На Рисунке 1 изображено графическое представление модели JST.

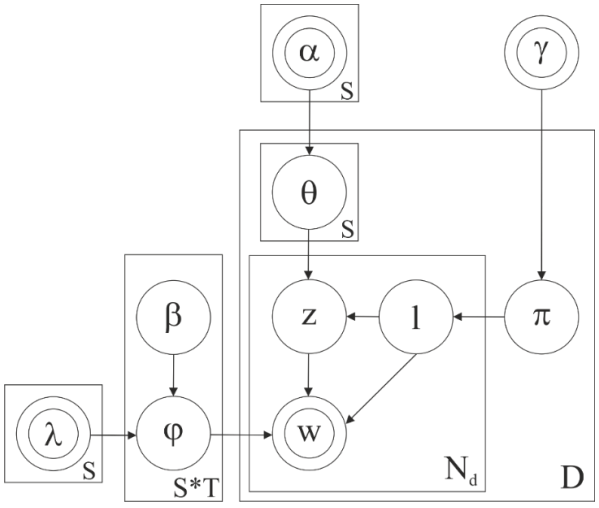


Рисунок 1. JST-модель

Здесь λ – матрица размерности $\mathrm{S} \times \mathrm{V}$, которая используется для кодирования априорной информации о тональности некоторых слов. Чтобы получить распределения π (распределение тональностей по документам), распределения θ (распределение тональностей в зависимости от тематических меток по документам) и распределение φ (распределение слов для каждой метки тональности и тематической метки по коллекции документов), необходимо вычислить апостериорное распределение тематических меток и меток тональностей.

Совместную вероятность присваивания словам тематических и тональной меток выражается следующим образом:

$$P(w,\ z,\ l) = P(w|z,l)P(z,\ l) = P(w|z,l)P(z|l)P(l)$$

Точное вычисление данной вероятности затруднительно, поэтому используется аппроксимация (посредством сэмплирования по Гиббсу):

$$P(z_t = j,\ l_t = k | w, z^{-t}, l^{-t}, \alpha, \beta, \gamma) \propto \frac{N_{k,j,w_t}^{-t} + \beta}{N_{k,j}^{-t} + V\beta} \cdot \frac{N_{d,k,j}^{-t} + \alpha_{k,j}}{N_{d,k}^{-t} + \sum_j \alpha_{k,j}} \cdot \frac{N_{d,k}^{-t} + \gamma}{N_d^{-t} + S\gamma},$$

где $-t$ означает исключение из подсчёта слова на t-ой позиции. Переменные z^{-t} и l^{-t} являются векторами присваивания тематических и тональных меток, соответственно, для всех слов в коллекции, не включая слово на t-ой позиции в документе d. Распределения φ, θ и π вычисляются следующим образом:

$$\varphi_{k,j,i} = \frac{N_{k,j,i} + \beta}{N_{k,j} + V\beta} \quad (1)$$

$$\theta_{d,k,j} = \frac{N_{d,k,j} + \alpha_{k,j}}{N_{d,k} + \sum_j \alpha_{k,j}}$$

$$\pi_{d,k} = \frac{N_{d,k} + \gamma}{N_d + S\gamma}$$

В рамках нашего исследования реализация алгоритма была модифицирована для дополнительного вычисления распределения φ без учёта

параметра β в формуле (1), оказывающего «сглаживающее» действие. Таким образом, упрощённая формула выглядела так:

$$\varphi_{k,j,i} = \frac{N_{k,j,i}}{N_{k,j}} \quad (2)$$

Её можно интерпретировать как отношение количества того, сколько раз слово на i-ой позиции получило тематическую метку j и метку тональности k к количеству того, сколько раз всего были «выведены» тематическая метка j с меткой тональности k. Перейдём к описанию основной и главной части исследования.

Глава 3. Эксперименты

3.1. Инициализация

Для проведения тестов было решено воспользоваться авторской реализацией исследуемой вероятностной модели JST, являющейся open-source программой, написанной на C++, и распространяемой под лицензией GNU General Public License. Исходный код программы можно скачать по адресу: https://github.com/linron84/JST. В четвёртой главе нашей работы описаны эксперименты, связанные с модификацией данного алгоритма.

Программу можно скомпилировать как под ОС Windows (например, используя CygWin), так и под ОС Linux посредством утилиты make и компилятора gcc.

Далее мы создаём конфигурационный файл, в котором указываем настройки, такие как путь к коллекции, путь к инициирующему словарю, путь для сохранения результатов, количество итераций сэмплирования по Гиббсу, предполагаемое количество тем, тональных меток и значения гиперпараметров распределения Дирихле.

Было решено провести шесть экспериментов с различным количеством тем (40, 45 и 50) и с различным количеством итераций сэмплирования (800 и 1000) для каждой темы. Количество тональных меток было установлено равным двум. Использовался первый вариант инициирующего словаря. Впоследствии было проведено ещё два эксперимента для второй версии словаря, в одном из которых использовался стемминг при кодировании априорной информации. Гиперпараметры распределения Дирихле было решено установить «по умолчанию»:

$$\beta = 0.01, \qquad \gamma = \frac{(0.05 \times \text{средняя длина документа})}{S}.$$

Параметр α вычислялся методом максимального правдоподобия (встроено в авторскую реализацию). Средняя длина документа созданной нами коллекции составляла 862 слова. Среднее время вычислений в одном

эксперименте составляло 2,5 часа. Это время существенно зависит от размера коллекции и количества тем, задаваемых при инициализации. В четвёртой главе будет рассказано, как можно уменьшить это время.

Далее перейдём к описанию процесса оценки результатов работы алгоритма.

3.2. Экспертная оценка качества классификации тем

Для оценки качества классификации слов по темам было решено использовать экспертную оценку, в силу того факта, что человек сделает эту работу лучше, чем вычисление оценки качества по формулам. В статье «Reading Tea Leaves: How Humans Interpret Topic Models» [12] об этом рассказывается подробнее. В оценках результатов участвовало три эксперта. Идея состояла в следующем. Для начала берём распределение слов по темам. По окончании вычислений в нашем распоряжении находится файл, где для каждой темы и каждой тональности вычислена вероятность, с которой каждое слово из коллекции принадлежит данной теме с данной тональностью. Как было отмечено ранее, мы дополнили реализацию вычислением этих вероятностей без «сглаживания» параметром β по формуле (2). Данный файл имеет структуру, изображённую на Рисунке 2.

```
Label:0  Topic:0
0.000000474097199 0.000000474097199          ...        P(w_|V||l, t)
...
Label:0  Topic:N
0.000004704285973 0.000004704285973          ...             ...
...
Label:1 Topic:N
0.000002492695710 0.000002492695710          ...             ...
```

Рисунок 2. Файл, содержащий распределение слов по темам и тональностям

Была написана вспомогательная утилита, которая читала данный файл, а также словарь коллекции wordmap.txt, в котором содержатся все слова коллекции с присвоенными им идентификаторами, и сохраняла эти данные в память. Далее мы «склеивали» темы, разбитые по тональностям в одну: например, Label0_Topic0 + Label1_Topic0 = Topic0. При этом, если некоторое

слово входило в данную тему и с положительной и отрицательной тональностью, то ему присваивалась бóльшая вероятность из соответствующих значений. Этот процесс изображён на Рисунке 3. Далее мы сортировали слова в каждой теме по убыванию вероятностей. Для экспертной оценки каждой темы выбирались первые пять слов с наибольшей вероятностью (сверху темы) и одно слово с маленькой вероятностью (снизу темы). Если слов с одинаковой маленькой вероятностью несколько, то выбиралось одно из них случайным образом.

Рисунок 3. Процесс слияния тональностей в теме

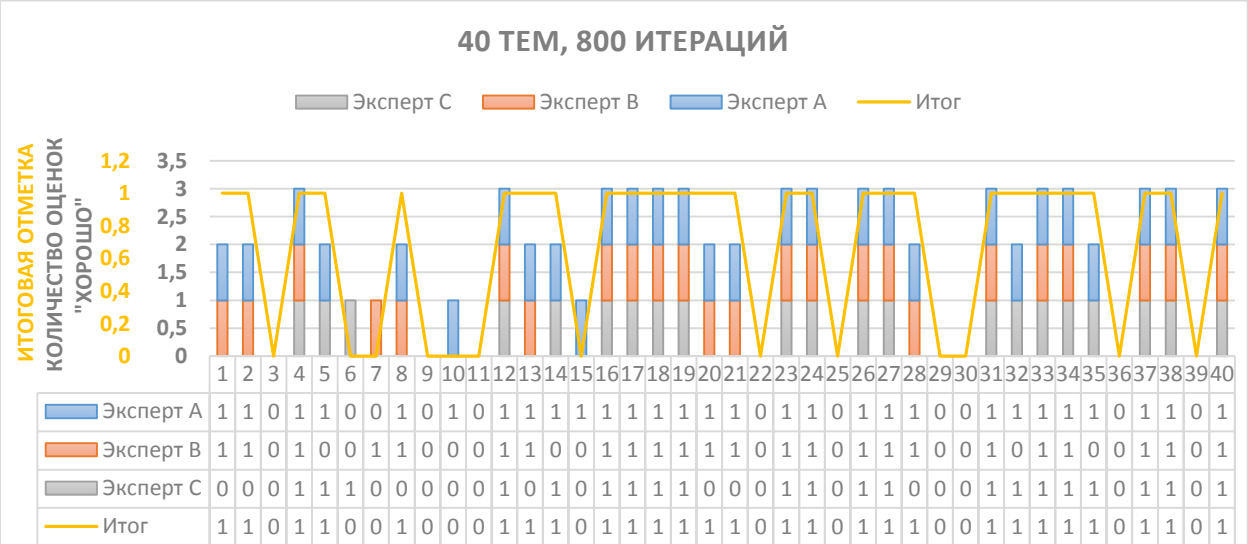
Далее из этой выборки генерировался тест для эксперта. Задача эксперта – для каждой темы из шестёрки слов исключить такое, которое по его мнению наименее ассоциировано с другими. Это слово эксперт должен был пометить знаком плюса и сохранить файл. Далее оценки трёх человек автоматически обрабатывались, и если эксперт исключил слово с маленькой вероятностью, то данная тема считалась «хорошей», ей присваивалась метка – единица, иначе – «плохая» тема и присваивалась метка 0. По окончании обработки результатов, каждой теме выставлялась итоговая оценка по простой формуле: если двое экспертов из трёх посчитали данную тему «хорошей», то она таковой и является (итоговая отметка – единица). Примеры тестовых выборок, предоставленных экспертам, изображены на Рисунке 4.

TOPIC #0	TOPIC #25	TOPIC #21
university	ukraine	pakistan
assembly	tymoshenko	iran
ireland	yanukovych	bahrain+
northern	ukrainian	khatami
wales	yushchenko	iranian
famine+	lard+	safavid
...	...	...
TOPIC #N	TOPIC #N	TOPIC #N
...	...	...

Рисунок 4. Пример строения теста для эксперта

Из крайнего правого теста видно, что экспертная оценка даётся легко не всегда. В данном примере все слова в той или иной мере связаны между собой. Поэтому эксперт должен иметь представление о предметной области, чтобы полученная оценка была наиболее объективной. Например, слово «pen» является частью имени французского политика Jean-Marie Le Pen. И будет неправильно исключать это слово из соображения, что это слово обозначает ручку для письма.

Для наглядности приведём совмещённый график количества оценок «хорошо» для каждой темы и итоговых оценок в эксперименте классификации коллекции на 40 тем при 800 итераций сэмплирования.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Эксперт A	1	1	0	1	1	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0	1	1	1	1	1	0	1	1	0	1
Эксперт B	1	1	0	1	0	0	1	1	0	0	0	1	1	0	0	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0	1	0	1	1	0	0	1	1	0	1
Эксперт C	0	0	0	1	1	1	0	0	0	0	0	1	0	1	0	1	1	1	1	0	0	0	1	1	0	1	1	0	0	0	1	1	1	1	1	0	1	1	0	1
Итог	1	1	0	1	1	0	0	1	0	0	0	1	1	1	0	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0	1	1	1	1	1	0	1	1	0	1

График 1. Оценки экспертов для первого эксперимента

Приведём такой же график для эксперимента с 50 темами и 1000 итераций сэмплирования (подписи осей идентичны осям предыдущего графика).

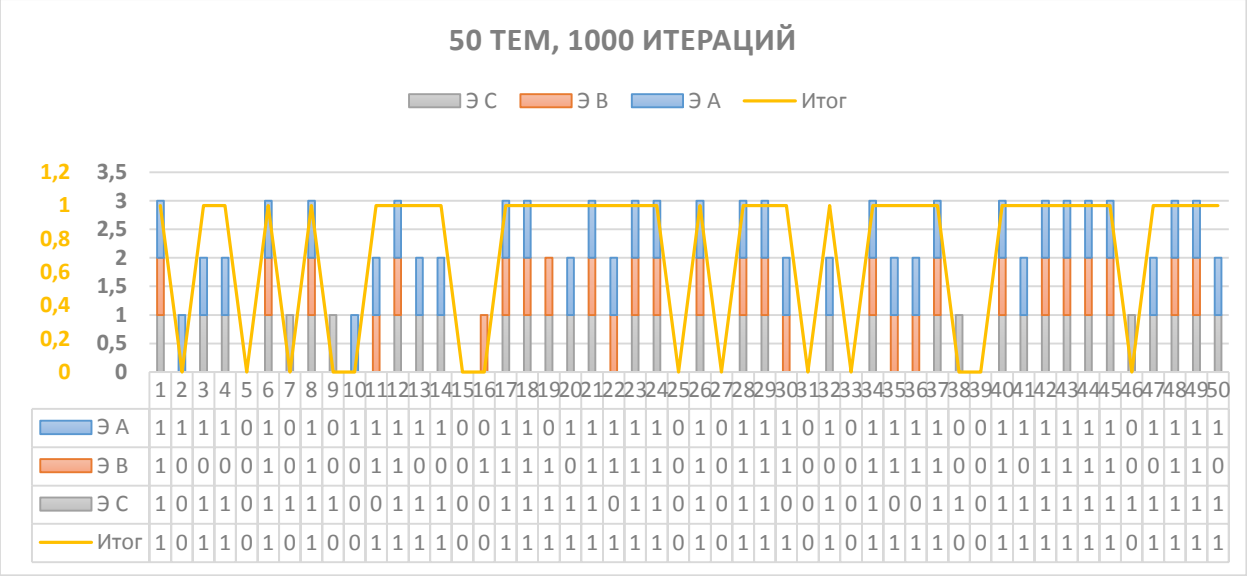


График 2. Оценки экспертов для шестого эксперимента

Во время оценки тем эксперту была известна лишь предметная область – политические выборы и Европейский Союз. В качестве экспертов выступили трое студентов старших курсов факультета Прикладной математики – процессов управления СПбГУ.

Далее приведём графики, в которых можно проследить зависимости количества «хороших» тем от количества итераций сэмплирования и общего количества тем, на которое классифицируется коллекция документов.

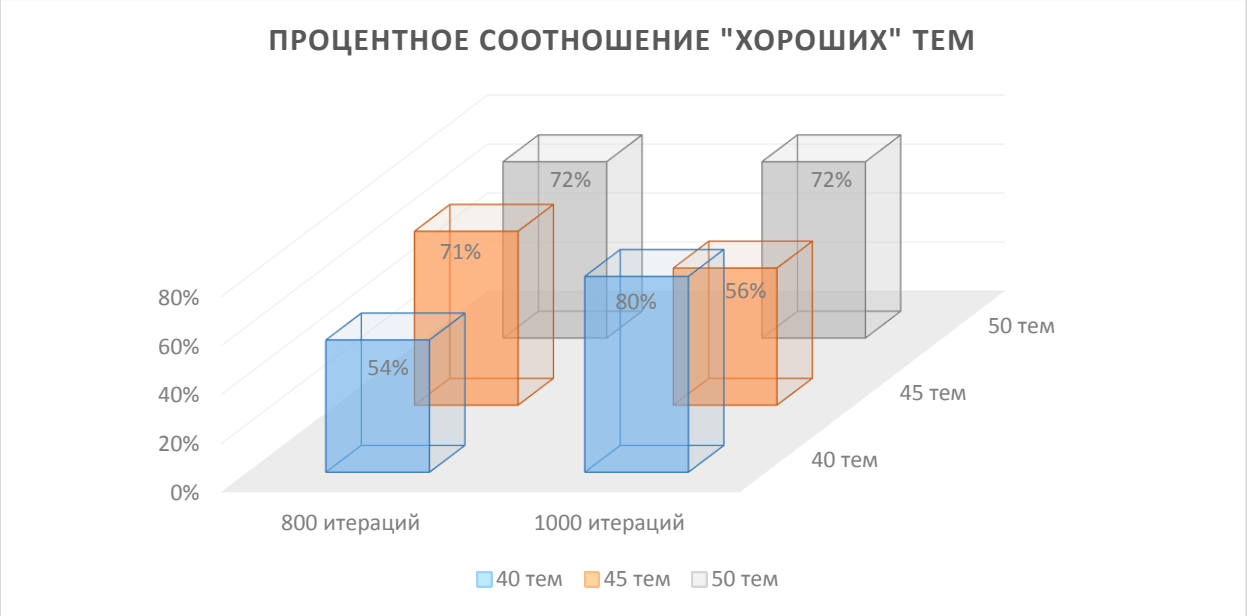


График 3. Процентное соотношение количества «хороших» тем в зависимости от параметров инициализации

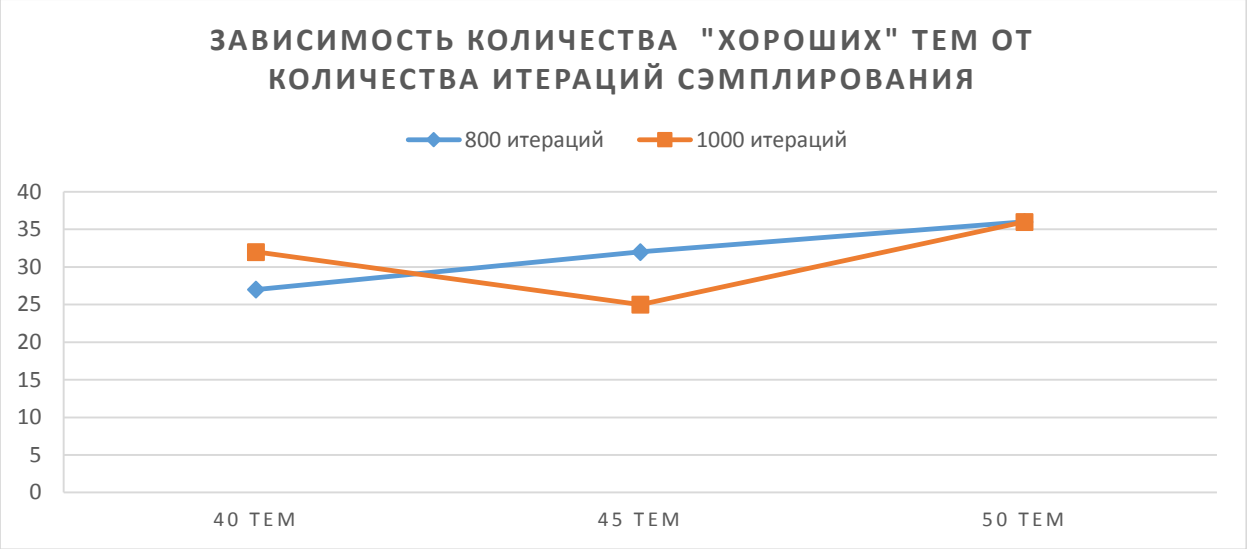
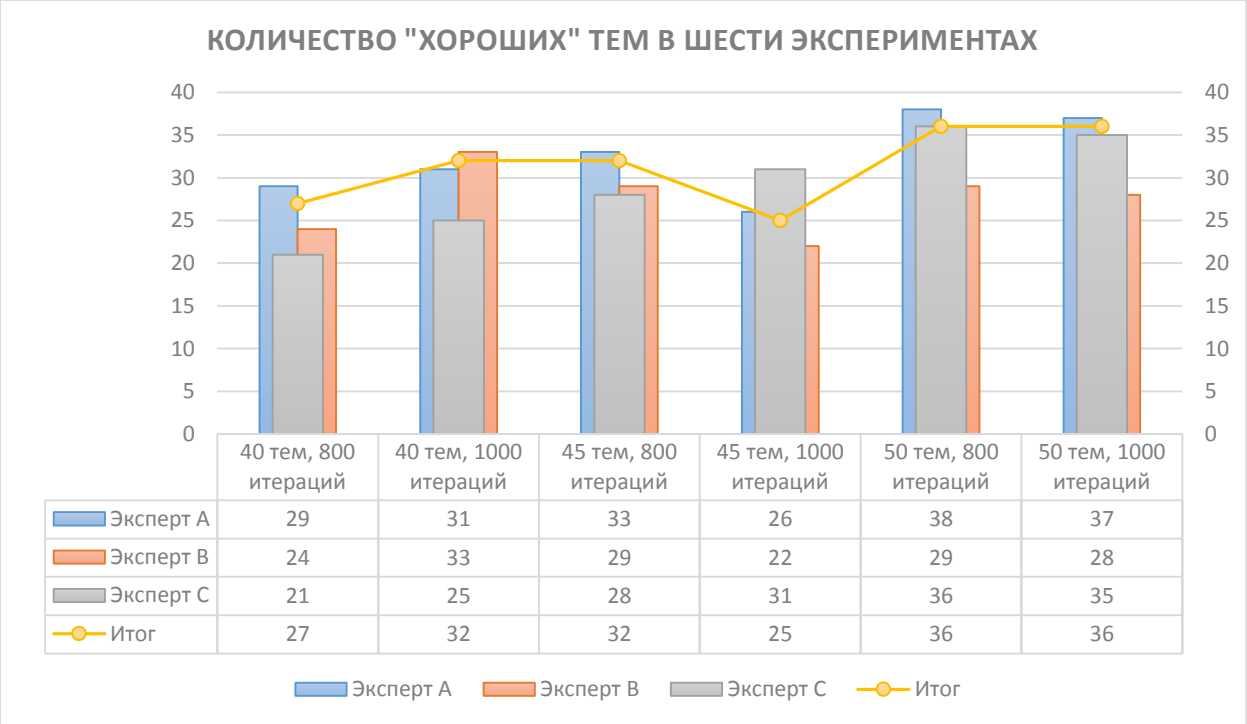


График 4. Зависимость количества «хороших» тем от итераций сэмплирования по Гиббсу

Обобщённый график количества «хороших» тем в каждом из шести экспериментов:



	40 тем, 800 итераций	40 тем, 1000 итераций	45 тем, 800 итераций	45 тем, 1000 итераций	50 тем, 800 итераций	50 тем, 1000 итераций
Эксперт А	29	31	33	26	38	37
Эксперт В	24	33	29	22	29	28
Эксперт С	21	25	28	31	36	35
Итог	27	32	32	25	36	36

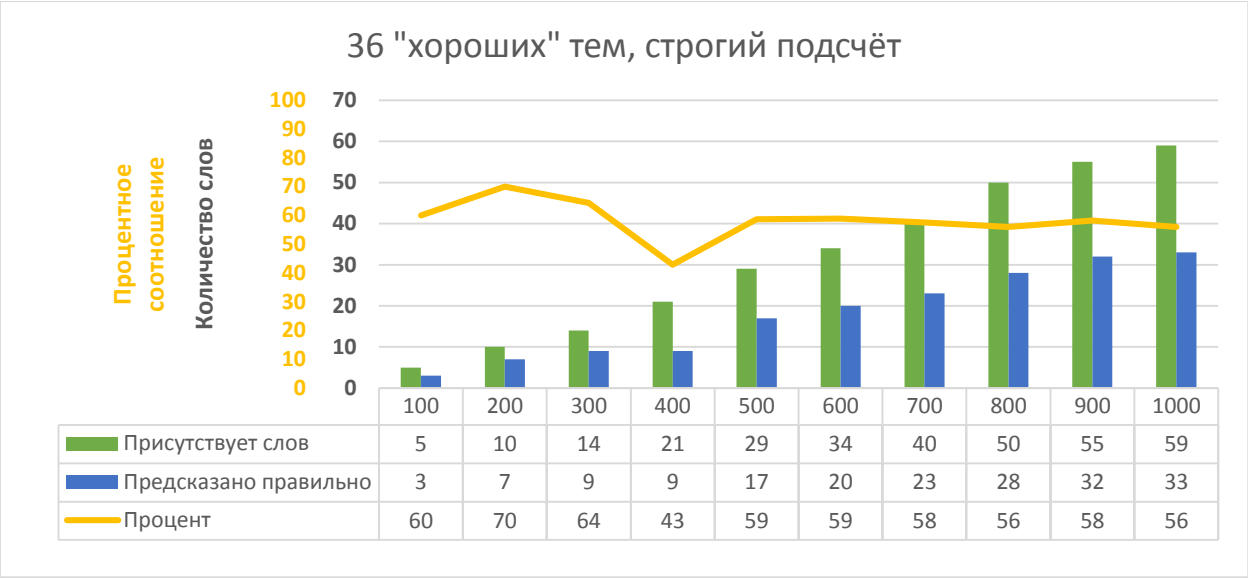
График 5. Общее количество выявленных «хороших» тем

На основании полученных результатов можно сделать некоторые выводы. По большому счёту, на основании произведённых шести экспериментов невозможно выявить влияние роста количества итераций сэмплирования на процентное соотношение получаемых «хороших» тем. Но

тенденция к росту заметна при 800 итерациях. Также можно сделать вывод, что для созданной коллекции документов классификация на 50 тем наиболее оптимальна: процент «хороших» тем получился одинаковым при разном количестве сэмплирования. Поэтому было решено далее работать с результатами классификации коллекции документов на 50 тем.

3.3. Оценка качества классификации слов по тональности

После оценки качества классификации слов по темам перейдём к описанию процесса оценки качества классификации слов по тональности. Но прежде нужно решить, какое количество слов из распределения каждой темы с отсортированными вероятностями считать ассоциированными. Данное количество было решено найти эмпирическим путём. Сама же оценка корректности классификации слов по тональностям проводилась следующим образом. Сначала из SWN выбирались все слова с наиболее выраженной тональностью (нейтральные оценка равна нулю). Всего оказалось 334 таких слова. Также далее мы рассматриваем только «хорошие» темы, определённые в предыдущем параграфе. Далее для каждого слова из выборки SWN проверялось, есть ли оно вообще в «хороших» темах, а также производился подсчёт сколько раз и в какой тональности данное слово встречалось в этих темах. Важным критерием проверки «правильности» работы алгоритма является индикатор того, что слова с наиболее выраженной тональностью из SWN встречаются часто (всегда) с соответствующей тональностью в «хороших» темах. Было проработано два варианта подхода к оценке: «строгое» и «нестрогое» (например, положительное слово из SWN равное количество раз получило обе метки тональности – считаем, что это правильное предсказание) сравнение количества попаданий в каждую тональность в «хороших» темах с эталонными значениями из SWN. Здесь и далее на оси абсцисс указывается количество берущихся первых слов каждой тональности, образующих тему (то есть реальный размер темы в два раза больше соответствующей абсциссы):



	100	200	300	400	500	600	700	800	900	1000
Присутствует слов	5	10	14	21	29	34	40	50	55	59
Предсказано правильно	3	7	9	9	17	20	23	28	32	33
Процент	60	70	64	43	59	59	58	56	58	56

График 6. Зависимость точности предсказаний от размера темы

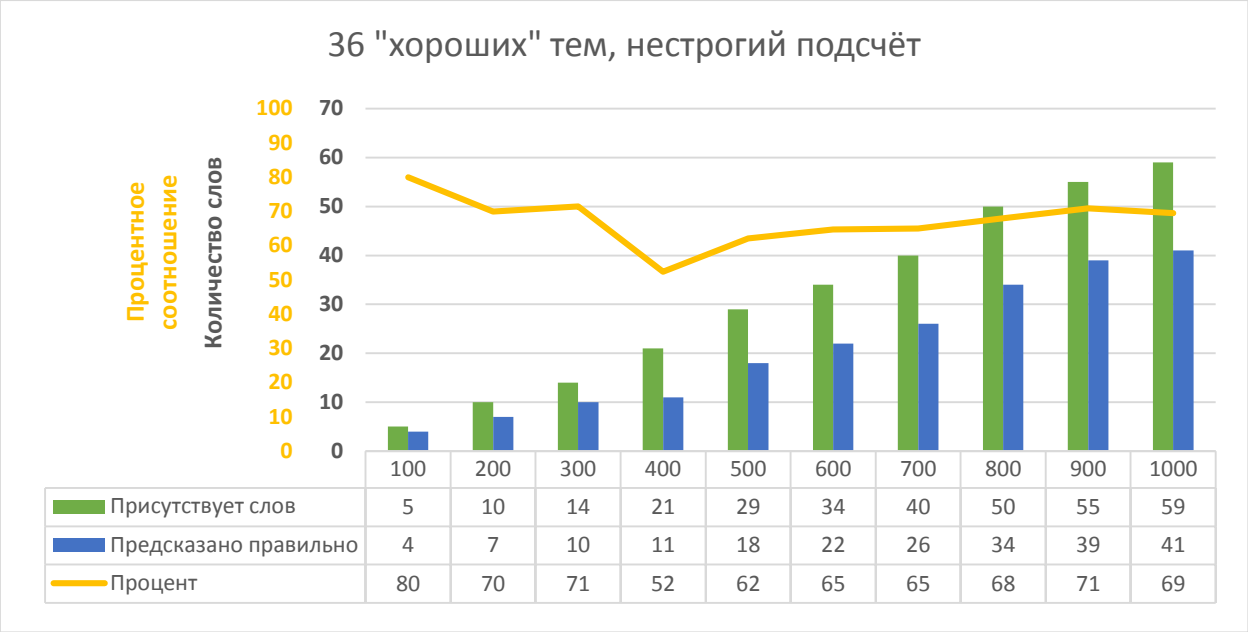
Для наглядности приведём таблицу найденных слов при размере темы в 200 слов каждой тональности (жирным шрифтом указаны слова, предсказанные «неправильно»):

Таблица 3. Слова с наиболее определённой тональностью и их частота встреч в «хороших» темах

Слово из SentiWordNet	*Сколько раз встречается в коллекции с позитивной тональностью*	*Сколько раз встречается в коллекции с негативной тональностью*	*Позитивная оценка в SWN*	*Негативная оценка в SWN*
echt	0	1	0.364	0.636
poor	1	0	0.125	0.875
vice	6	0	0.625	0.375
worst	0	1	0.0	1.0
fresh	0	1	0.375	0.625
retiring	1	0	0.25	0.75
terror	0	1	0.25	0.75
bush	3	1	0.125	0.875
congratulations	1	0	1.0	0.0
genuine	0	2	0.364	0.636

Отметим, что в нашей политической выборке слово «bush» с очень высокой долей вероятности означало распространённую английскую фамилию, то есть употреблялось не в том смысле, который указан в SWN. На

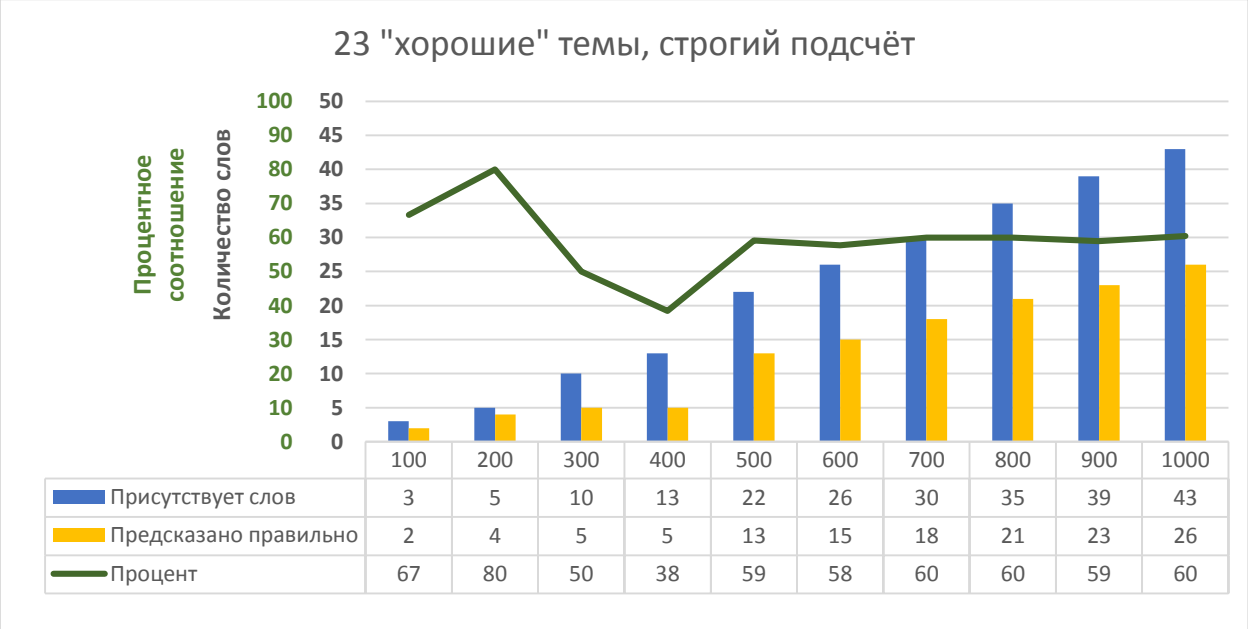
следующем графике подсчёт правильно предсказанных слов производился посредством нестрогого сравнения:



	100	200	300	400	500	600	700	800	900	1000
Присутствует слов	5	10	14	21	29	34	40	50	55	59
Предсказано правильно	4	7	10	11	18	22	26	34	39	41
Процент	80	70	71	52	62	65	65	68	71	69

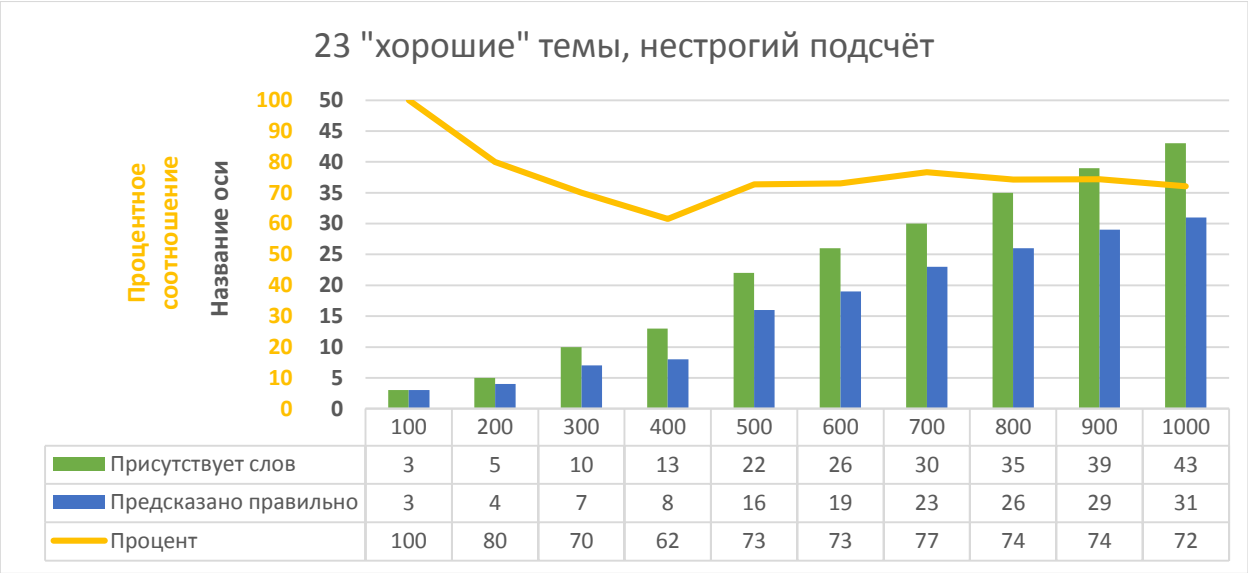
График 7. Количество правильных предсказаний тональности в зависимости от размера тем при нестрогом подсчёте

Возникло предположение, что если выбрать «самые хорошие» темы (все три эксперта дали оценку «хорошо»), то процент количества предсказанных слов может увеличиться. Проверяем гипотезу:



	100	200	300	400	500	600	700	800	900	1000
Присутствует слов	3	5	10	13	22	26	30	35	39	43
Предсказано правильно	2	4	5	5	13	15	18	21	23	26
Процент	67	80	50	38	59	58	60	60	59	60

График 8. Количество правильных предсказаний в «самых хороших» темах при строгом подсчёте



	100	200	300	400	500	600	700	800	900	1000
Присутствует слов	3	5	10	13	22	26	30	35	39	43
Предсказано правильно	3	4	7	8	16	19	23	26	29	31
Процент	100	80	70	62	73	73	77	74	74	72

График 9. Количество правильных предсказаний в «самых хороших» темах при нестрогом подсчёте

Из вышеприведённых результатов можно сделать вывод, что «назначение» тональностей при инициализации алгоритма работает, и что используя темы, оценённые всеми экспертами одновременно как «хорошие», можно увеличить процент правильно предсказанных слов.

3.4. Named-Entities Recognition

Теперь, когда у нас коллекция классифицирована по темам и установлены тональности слов, из неё можно получить большое количество полезной информации. Было решено рассмотреть направление Named-Entities Recognition (далее – NER) – задача автоматического распознавания и извлечения структурированной информации из неструктурированных данных, а именно нахождение именованных сущностей: имён людей, географических местоположений, названий организаций, событий, числовых элементов и так далее. Возникла идея о возможности создания механизма для автоматического построения словарей, в которых бы были описаны контексты употребления с различными тональными оценками (положительный/негативный контекст) тех или иных именованных сущностей. Таким образом, необходимо было лишь выделить именованные сущности из исходных статей нашей коллекции из Википедии и составить для них словарь, где контекстом употребления выступали бы полученные ранее «хорошие» темы.

В качестве инструмента для выделения именованных сущностей была выбрана библиотека Stanford Named Entity Recognizer, предоставляющая богатый функционал, готовые обученные модели и удобное API для языка Java. Была выбрана модель классификации по трём видам сущностей: организации, места и люди. Поиск можно проводить либо в названиях статей, либо последовательно пройти по всем текстам статей из коллекции. Были опробованы оба варианта.

Далее приведём последовательность действий, необходимых для создания словаря:

a) Выделяем именованные сущности посредством Stanford NER;
b) Проходим по выделенным сущностям, приводим слово к нижнему регистру и ищем среди «хороших» тем, полученных ранее, где встречается данное слово. Если сущность состоит из нескольких слов (как большинство имён), то мы разделяем её на составные слова, приводим к нижнему регистру и для каждой части ищем темы, в которых она упоминается. Далее берётся пересечение тем всех частей слова-сущности.
c) Выводим, например, по 20-30-40 и так далее слов (с наибольшей вероятностью либо случайно) каждой тональности из найденных тем;
d) Сохраняем результат в базу данных, по желанию формируем HTML;

Приведём некоторые результаты вышеописанного процесса. В следующей таблице указаны некоторые позиции из автоматически созданного словаря для сущностей из заголовков статей на тему политических выборов с упоминанием Европейского Союза в них:

Таблица 4. Примеры из составленного контекстного словаря для именованных сущностей

Именованная сущность	**Тип**	**Контекст**
Vladimir Putin	PERSON	[TOPIC #10] [negative] ukraine, tymoshenko, yanukovych, ukrainian, yushchenko, viktor, yulia, presidential, kuchma, party, regions, stated, december, political, election, russian, rada, february, november, parliament;

		[TOPIC #14] [positive] bulgaria, bulgarian, kilroy, silk, dimitrov, vattimo, udf, sofia, prout, gianni, elles, movement, young, kostov, press, blue, beazley, binev, weak, youroukov;
Nicolas Sarkozy	PERSON	[TOPIC #31] [negative] french, france, sarkozy, nicolas, socialist, bayrou, right, left, paris, françois, hollande, jean, round, ump, chirac, marie, movement, council, jacques, wing;
Hezbollah	ORG	[TOPIC #14] [negative] hezbollah, lebanon, israel, lebanese, israeli, terrorist, syrian, syria, organization, islamic, attack, iran, resistance, attacks, movement, anti, beirut, terrorism, group, assad, july, jihad, fighters, bombing, iranian, weapons, rockets, missiles, killing, manar, targets, nasrallah, minister, hariri, border, southern, shia, arab, accused, list;
European Union	ORG	[TOPIC #34] [positive] political, time, thumb, people, country, history, left, major, years, union, european, war, power, called, popular, began, right, politics, main, period, led, post, largest, high, general, saw, started, victory, end, economic, won, second, despite, lost, economy, took, received, policy, old, joined; [TOPIC #10] [positive] council, commission, initiative, citizens, commissioner, proposal, proposed, national, initiatives, process, registered, legislation, proposals, set, organisations, european, signatures, congress, politics, draft, petition, online, statute, eci, introduced, popular, regulations, adopted, fundamental, propose, level, consultation, delimitation, context, december, procedure, charter, principles, resolution, organised; [negative] ukraine, tymoshenko, yanukovych, ukrainian, yushchenko, viktor, yulia, presidential, kuchma, party, regions, stated, december, political, election, russian, rada, february, november, parliament, yatsenyuk, verkhovna, parliamentary, prime, president, gas, bloc, january, according, october, kiev, russia, orange, european, second, leonid, revolution, tyahnybok, oleksandr, soviet; [TOPIC #11] [negative] referendum, european, constitution, treaty, union, referendums, membership, held, amendment, vote, ireland, yes, article, result, favour, voters, state, europe, government, lisbon, constitutional, approved, june, question, changes, turnout, accession, results, republic, day, majority, external, binding, reference, rejected, references, international, communities, proposal, related; [TOPIC #44] [positive] united, nations, council, security, group, states, european, non, members, elected, held, general, assembly, voting, caribbean, permanent, western, latin, seats, african, seat, asia, africa, elections, countries, american, canada, october, year, member, union, rules, japan, pacific, germany, regional, italy, america, follows, list;

		[TOPIC #30] [positive] elections, local, council, vote, seats, election, county, electoral, address, results, divisions, entitled, councillors, electors, democrats, register, student, away, registered, district, labour, english, liberal, term, office, conservatives, thursday, result, conservative, candidates, kingdom, overall, place, university, borough, working, commonwealth, government, summary, abroad; [TOPIC #15] [positive] serbia, montenegro, kosovo, democratic, independence, bosnia, tadić, serbian, european, herzegovina, albania, ethnic, dps, serb, albanian, union, srpska, republika, boris, republic, macedonia, list, đukanović, montenegrin, future, sdp, federation, stated, coalition, serbs, bosniak, albanians, state, pro, thaçi, voting, bih, macedonian, dss, integration;
Finland	LOCATION	[TOPIC #17] [negative] european, parliament, group, europe, meps, epp, pes, socialists, people, democrats, president, treaty, german, elections, lisbon, june, liberals, commission, cohn, bendit, verhofstadt, green, members, barroso, alde, belgium, alliance, italy, movement, schulz, mep, germany, france, new, brok, french, greens, states, spinelli, conservatives; [TOPIC #12] [positive] finland, finnish, finns, true, väyrynen, centre, soini, katainen, swedish, helsinki, green, halonen, åland, bailout, league, democrats, support, electoral, ncp, stubb, coalition, foreign, hautala, niinistö, kiviniemi, timo, incumbent, district, jyrki, vanhanen, sdp, work, tarja, sauli, spp, paavo, advance, media, urpilainen, sanomat;
Berlin	LOCATION	[TOPIC #0] [positive] university, affairs, foreign, member, president, minister, national, council, career, education, people, international, political, policy, vice, secretary, social, union, relations, law, assembly, deputy, politicians, school, alumni, chairman, external, democratic, appointed, born, institute, served, head, living, degree, europe, births, studies, economic, november; [TOPIC #35] [positive] moldova, mckinney, lukashenko, belarus, georgia, belarusian, osce, integration, ghimpu, georgian, timofti, green, karabakh, moldovan, nagorno, hemp, cis, september, herer, communists, transnistria, pcrm, mihai, bpf, cynthia, sannikaŭ, alexander, minsk, katrina, emo, atlanta, declared, filat, dream, ship, saakashvili, square, soviet, lupu, vlad; [TOPIC #9] [negative] hungarian, hungary, communitarianism, communitarian, communitarians, budapest, occupy, taunton, community, communities, fidesz, szegedi, lothian, mendip, jobbik, marquess, yeovil, bridgwater, kerr, responsive, socialist, áder, schmitt, frome, academic, dame, central,

		street, camp, moral, expressed, chard, burnham, etzioni, berlin, issue, shared, minorities, lmp, orbán;

Всего в заголовках статей посредством API Stanford NER было найдено 690 (343) имён людей, 62 (57) географических названий и 57 (55) организаций. В скобках указаны итоговые количества соответствующих сущностей в полученном словаре. Это связано с тем, что из каждого распределения тем мы брали по 250 слов каждой тональности (в общем случае в распределении темы присутствуют все слова из коллекции), и не все слова могут оказаться в такой выборке. К тому же, слова меньше трёх символов мы вообще не рассматривали при классификации, поэтому найти контекст для фамилии «Li» не представляется возможным. Для увеличения количества найденных слов, можно брать по 300-400-500 слов каждой тональности для темы.

Полную версию словаря можно просмотреть по адресу в Интернете: www.mylexicon.org.

3.5. Автоматизация процесса создания словарей

Из полученных в предыдущем параграфе результатов, можно сделать вывод, что применённые нами инструменты и методы позволяют создавать словари контекстов (с их эмоциональной оценкой) для именованных сущностей практически в автоматизированном режиме. В будущем, возможно даже в магистерской работе, можно создать комплекс программ, посредством которого можно было бы генерировать контекстные словари для разных предметных областей. Этапы работы такого приложения изображены на Рисунке 5.

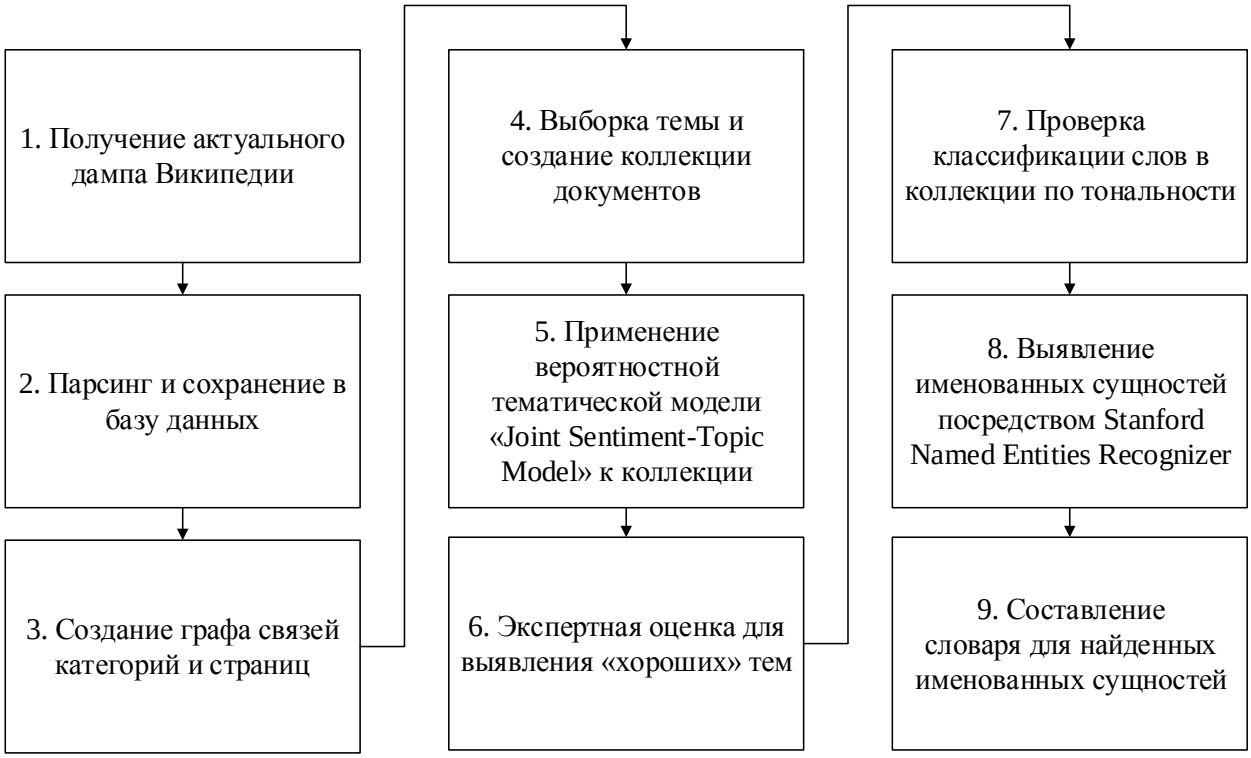


Рисунок 5. Этапы генерирования контекстных словарей

Опишем некоторые моменты и некоторые перспективы, касающиеся этой схемы. Актуализация дампа Википедии происходит раз в месяц, поэтому первые три этапа можно выделить как отдельную утилиту обновления исходных данных. Также можно использовать в качестве исходного массива данных для создания коллекций не Википедию, а, например, новостные сводки. Для шестого этапа можно создать, например, WEB-интерфейс с раздельным доступом для экспертов. Напомним, важным моментом является тот факт, что каждый эксперт должен иметь представление о предметной области. На седьмом этапе важен факт того, что назначение тональностей «работает», и критерием в нашей работе выступал словарь SWN, но можно выбрать и другие варианты кодирования априорной информации. Восьмой этап также можно изучить более глубоко: пробовать различные инструменты, обучать классификатор на своих тренировочных данных и прочее. Можно генерировать словарь не только для именованных сущностей, а, например, для всех слов. Для девятого этапа можно разработать формат файла словаря для

дистрибуции его по Интернету, либо хранить данные в базе данных и распространять словарь в качестве SQL-дампа.

Так как наша работа носила исследовательский характер, на данный момент разные этапы процесса были написаны на разных языках, с разной архитектурой. Но если стандартизовать все этапы и выпустить данный комплекс программ, то несомненно ему будет найдено достойное применение.

Глава 4. Модификация реализации алгоритма JST

4.1. Параллельные вычисления

В процессе изучения работы алгоритма было решено модифицировать авторскую реализацию, чтобы повысить производительность и уменьшить время вычислений. Для этих целей была выбрана библиотека OpenMP для C++. Данная библиотека снижает порог вхождения в область распределённых вычислений и предоставляет широкий набор настроек и инструментов. Основная идея заключалась в следующем. Для начала необходимо было выделить циклы, где отсутствует зависимость данных – ситуация, когда на текущей итерации цикла используются результаты предыдущей операции. В исходном алгоритме на каждой итерации сэмплирования по Гиббсу вычислялась апостериорная вероятность, посредством которой далее производилось назначение слову тональной и тематической меток. А также при обновлении значений гиперпараметров присутствовали «независимые» вычисления.

Таким образом, аналитическим путём были определены циклы, которые возможно вычислять в несколько потоков. Далее необходимо было с помощью прагм в языке C++ оставить указания компилятору в соответствующих местах и скомпилировать исходные коды с флагом `-fopenmp -Wall`. В указаниях можно было назначить количество потоков, на которое стоит разбить вычисления. Это число было установлено равным четырём.

Несомненным плюсом библиотеки OpenMP является тот факт, что мы незначительно меняем исходный код программы и такая «параллельная» версия может скомпилироваться и в обычном режиме, без необходимости удалять прагмы.

На Рисунке 6 приведён фрагмент кода, который позволяет обрабатывать циклы в несколько потоков.

```
...
omp_set_num_threads(4);

#pragma omp parallel
{
        #pragma omp for private (i, j) nowait
        for (i = 0; i < numSentiLabs; i++) {
                for (j = 0; j < numTopics; j++) {
                // Вычисления
                }
        }
}
...
```

Рисунок 6. Пример распараллеливания цикла

Далее представим результаты сравнения работы алгоритмов при классификации коллекции на сорок, сорок пять и пятьдесят тем соответственно.

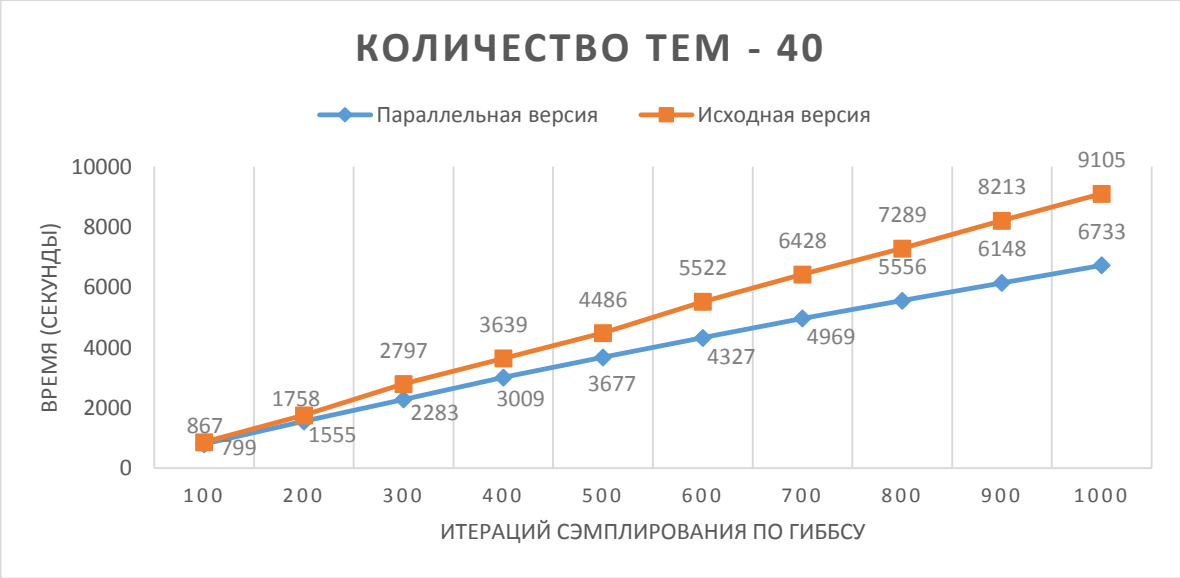


График 10. Время вычислений при классификации на 40 тем

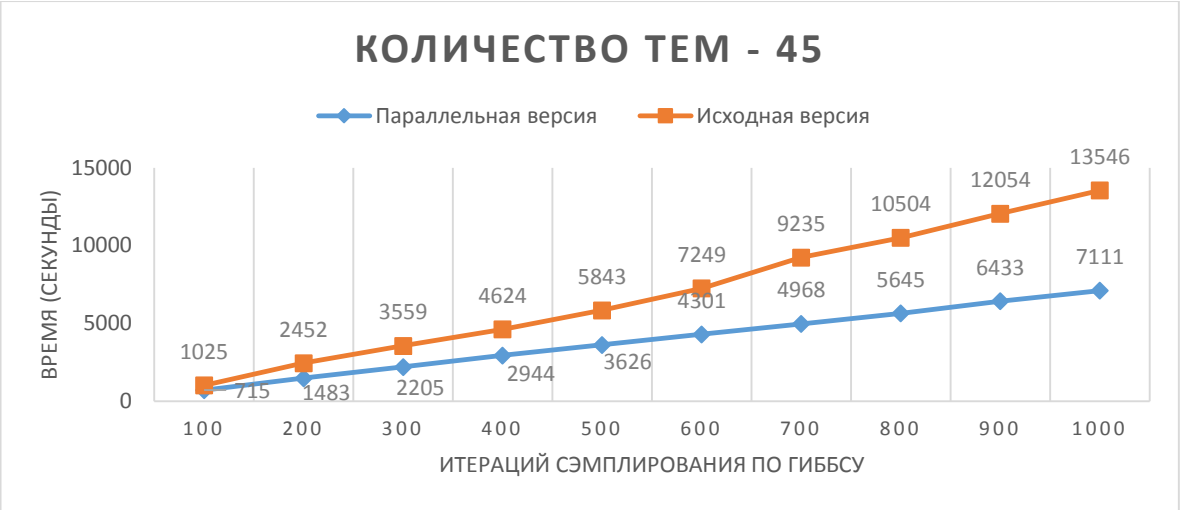


График 11. Время вычислений при классификации на 45 тем

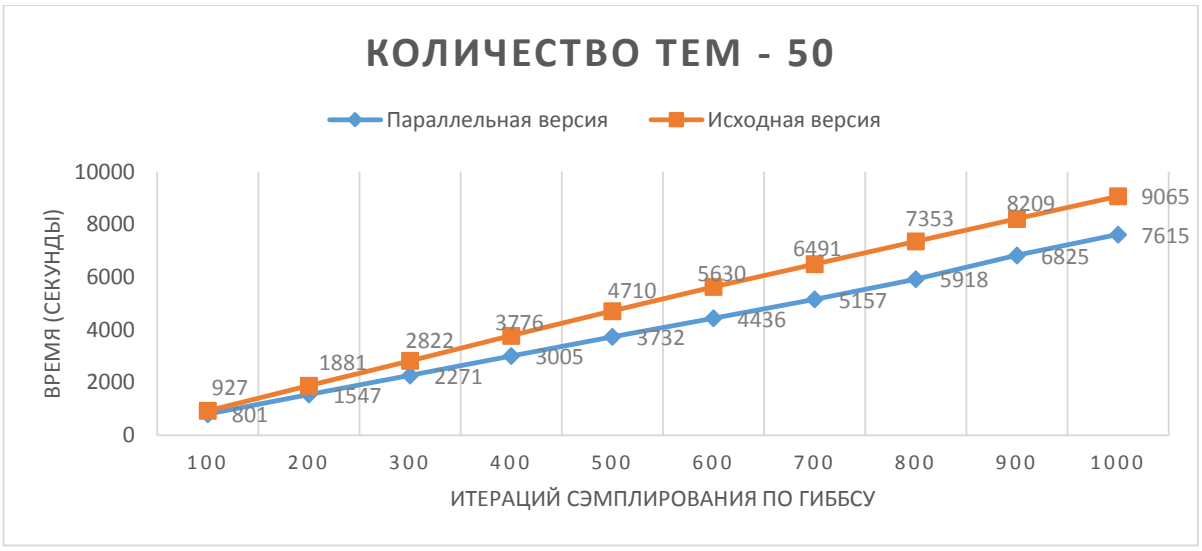


График 12. Время вычислений при классификации на 50 тем

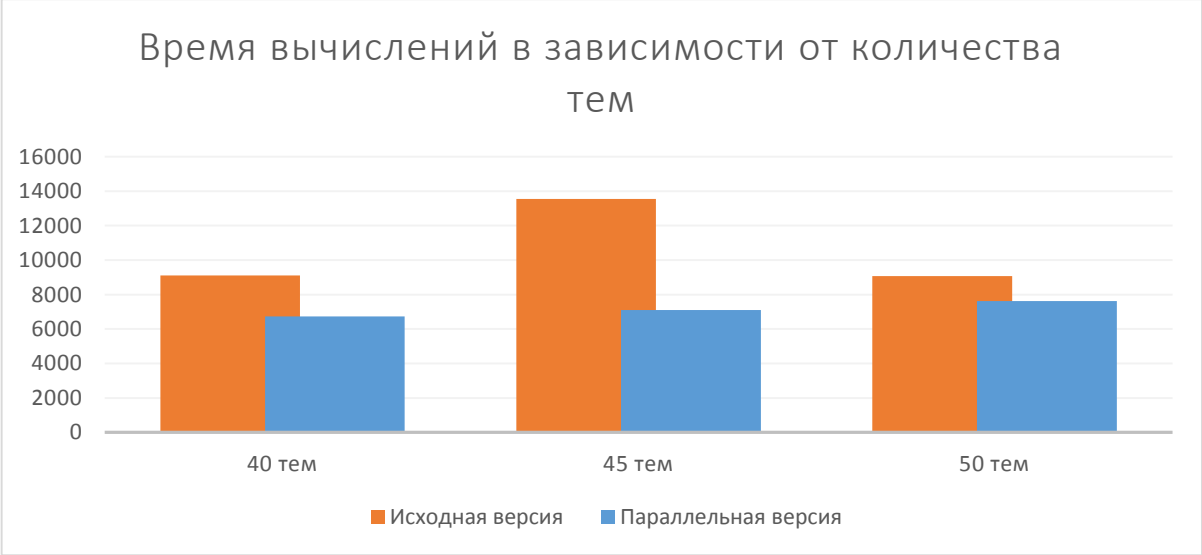


График 13. Сравнительные временные затраты при классификации на разное количество тем

Очевидно, что в зависимости от роста количества тем, увеличивается также и количество вычислений. Эта тенденция прослеживается в замерах времени работы параллельной модификации алгоритма. Некоторое удивление вызвал тот факт, что в исходной версии время выполнения вычислений «скачет». После повторного замера времени работы исходной версии для 45 тем оказалось, что время вычислений заметно уменьшилось, но по-прежнему превышало время работы параллельной модификации. Объяснить причину таких «скачков» не представляется возможным, все эксперименты проводились на одной машине и в одинаковых условиях. Можно только предположить, что время вычисления при однопоточных вычислениях состоит в некоторой зависимости от загруженности ресурсов рабочей машины.

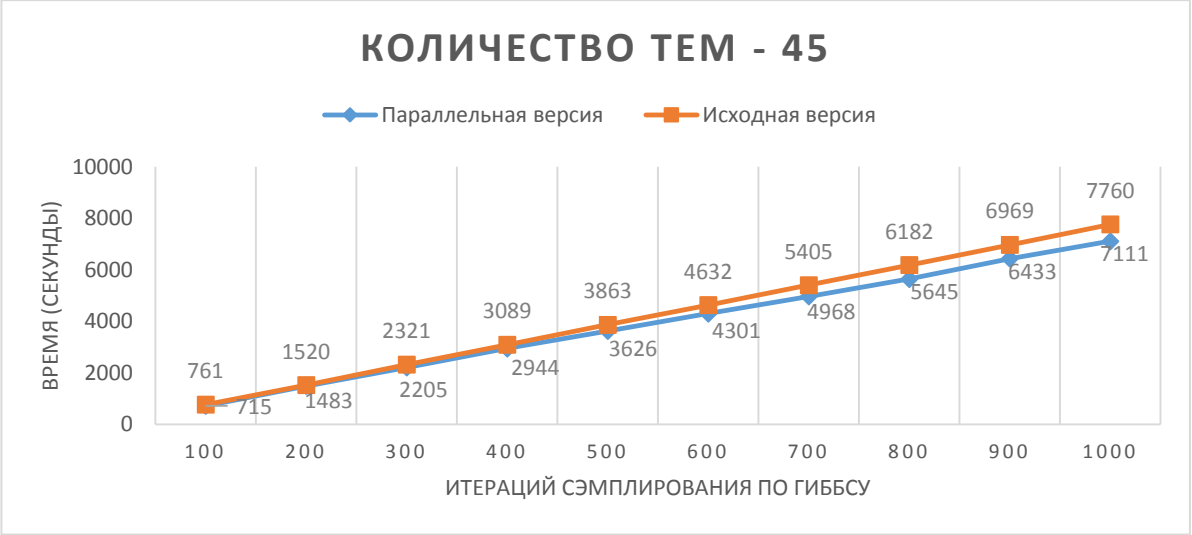


График 14. Время вычислений при повторной классификации на 45 тем

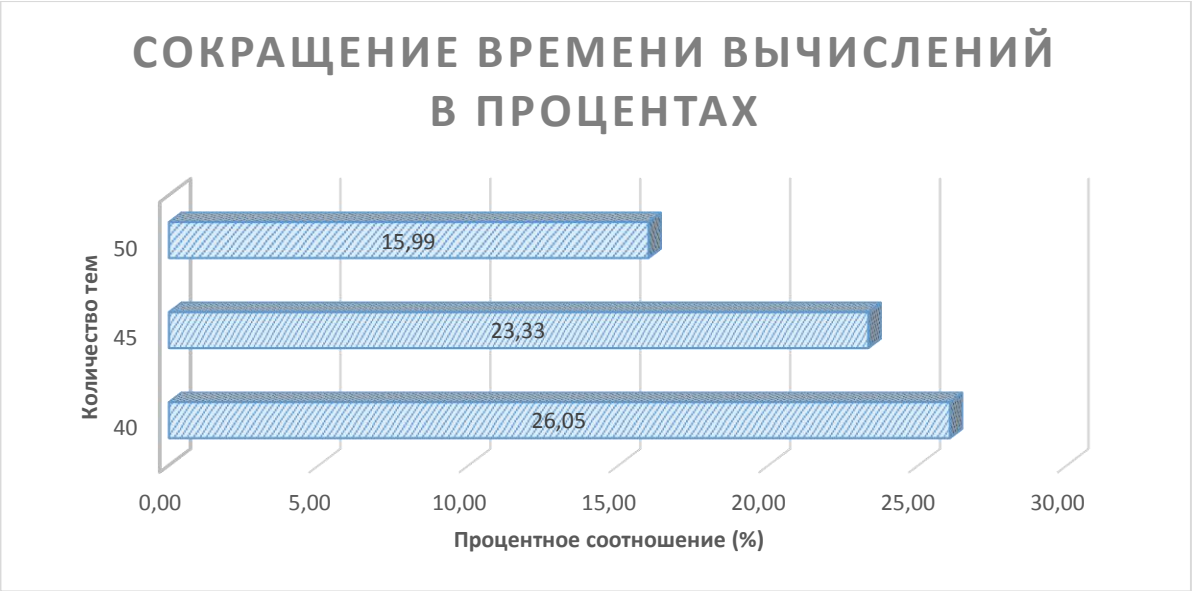


График 15. Процентное соотношение уменьшения времени вычислений при использовании параллельной версии

Таким образом, посредством параллельной модификации можно в среднем сократить время вычислений на 23%. Но стоит отметить, что при однопоточных вычислениях не удалось получить однозначного представления, от чего зависит время обработки всех данных.

4.2. Использование стемминга при кодировании словаря

С целью увеличить количество слов, которые получат априорную информацию о тональности при инициализации работы алгоритма, с учётом необходимости возможности последующей экспертной оценки (сохранения читабельности слов) была произведена модификация инициализации алгоритма. Для этого в авторскую реализацию была добавлена open-source имплементация улучшенного алгоритма Портера (Porter2) [13].

Далее действовали следующим образом. При загрузке инициирующего словаря производился стемминг, и сохранялись оба варианта словаря. При кодировании априорной информации происходит проверка слова из текста на наличие в словаре. При этом мы производили стемминг слова из текста «на лету» и искали его в словаре. Таким образом все «однокоренные» слова получали одну и ту же тональную оценку. Ожидалось, что качество классификации текста по темам и тональности от этого возрастёт, но как

оказалось, оно несколько снизилось. В Таблице 5 приведены данные экспертной оценки для двух экспериментов с числом тем равным 50, проведённой по тому же принципу, который был описан ранее. Использовался второй вариант инициирующего словаря, состоящая из тренировочной и тестовой выборок. Кратко приведём результаты экспериментов:

Таблица 5. Результаты оценки качества классификации слов для второй очереди экспериментов

Эксперимент	**Количество «хороших» тем из 50**	**Процентное соотношение количества «хороших» тем**	**Всего слов (правильно предсказанных) слов из тестовой выборки при размере темы в 1000 слов каждой тональности**	**Процентное соотношение правильно предсказанных слов из тестовой выборки**
Без стемминга	35	70%	19 (10)	52%
Со стеммингом	29	58%	14 (6)	42%

Из результатов видно, что использование стемминга не дало прироста количества «хороших» тем, а наоборот, привело к их снижению. Процент правильно предсказанных слов также оставляет желать лучшего.

Выводы

Подводя итог проделанной работе, отметим, что все поставленные задачи удалось выполнить. Был показан способ составления коллекции документов для темы политических выборов (с упоминанием Европейского Союза в них) из англоязычной Википедии. Показанный принцип позволяет создавать коллекций документов и для других предметных областей. Далее на этой коллекции была протестирована вероятностная тематическая модель «Joint Sentiment-Topic Model» и предложена последовательность действий для экспертной оценки качества классификации слов из коллекции по темам. Была показана зависимость количества «хороших» тем от выбора их общего количества, и от количества итераций сэмплирования. Словарь SentiWordNet был использован в качестве критерия корректности классификации слов из коллекции по тональности. Были произведены модификации авторской имплементации JST для уменьшения времени вычислений, а также исследовано влияние использования стемминга при кодировании априорной информации о тональностях. Далее было показано, как можно использовать результаты классификации текстов посредством модели JST в области Named Entities Recognition. Был составлен словарь для именованных сущностей, выделенных из заголовков статей коллекции, содержащий контекст упоминания для каждой такой сущности, с соответствующей тональной оценкой. Был описан потенциальный комплекс программ, позволяющий практически в автоматизированном режиме генерировать новые контекстные словари.

Из всего этого можно сделать вывод, что вероятностная тематическая модель «Joint Sentiment-Topic Model» может успешно применяться в задачах классификации текстов по тональности и имеет большой потенциал в прикладных задачах, таких как составление контекстных тональных словарей.

Заключение

Исследование показало, что посредством математической статистики и теории вероятностей можно успешно выделять практически в автоматизированном режиме интересную информацию из большого массива данных. Показанная в работе эффективность вероятностной модели JST в такой прикладной области, как составления контекстных тональных словарей, говорит о том, что имеет смысл разработать качественный комплекс программ, который бы позволял проделывать всю работу, описанную в исследовании, в несколько «кликов». В свою очередь словари, полученные посредством такого комплекса программ, могут использоваться в различных прикладных областях: в контекстной рекламе, рекомендательных системах, оценке отзывов и многих других. Создание такого комплекса программ может стать объектом магистерской диссертации.

Список литературы

1. Lin C., He Y. Joint Sentiment/Topic Model for Sentiment Analysis // CIKM '09 Proceedings of the 18th ACM conference on Information and knowledge management, 2009. P. 375-384.
2. Lin, C., He, Y., Everson R., Rüger S. Weakly-supervised joint sentiment-topic detection from text // IEEE Transactions on Knowledge and Data Engineering, Volume 24 Issue 6, June 2012. P. 1134-1145.
3. David M. Blei, Andrew Y. Ng, Michael I. Jordan. Latent dirichlet allocation // The Journal of Machine Learning Research, Volume 3, 3/1/2003. P. 993-1022.
4. Esuli. A, Sebastiani F. SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining // Istituto di Scienza e Tecnologie dell'Informazione
5. Маннинг К. Д., Рагхаван П., Шютце Х. Введение в информационный поиск. М. : ООО «И.Д. Вильямс», 2001. 528 с.
6. Воронцов К. В. Вероятностное тематическое моделирование. http://www.machinelearning.ru/wiki/images/f/fb/Voron-ML-TopicModels.pdf
7. Heinrich G. Parameter estimation for text analysis // Technical Note, University of Leipzig, Germany, 2009.
8. Named Entity Recognition and the Stanford NER Software. http://nlp.stanford.edu/software/jenny-ner-2007.pdf
9. Седжвик Р. Алгоритмы на графах. 3-е изд. Россия, Санкт-Петербург: «ДиаСофтЮП», 2002. 496 с.
10. Введение в OpenMP: параллельное программирование на C++. https://software.intel.com/ru-ru/blogs/2011/11/21/openmp-c
11. The English (Porter2) stemming algorithm. http://snowball.tartarus.org/algorithms/english/stemmer.html

12. Chang J., Gerrish S., Wang C., Boyd-Graber J. L., Blei D. M. Reading Tea Leaves: How Humans Interpret Topic Models // Advances in Neural Information Processing Systems 22, 2009. P. 288-296.
13. Open-source имплементация алгоритма Porter2. https://bitbucket.org/smassung/porter2_stemmer