

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий



**ДИССЕРТАЦИЯ**  
**на соискание ученой степени**  
**МАГИСТРА**

**Тема: Полуавтоматическое извлечение часто задаваемых вопросов из обращений в службу поддержки**

Студент гр. 63501/3 П.П. Жук



Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий

Диссертация допущена к защите  
зав. кафедрой

\_\_\_\_\_ В.М. Ицыксон

«\_\_\_\_» \_\_\_\_\_ 2017 г.

## **ДИССЕРТАЦИЯ на соискание ученой степени МАГИСТРА**

**Тема: Полуавтоматическое извлечение часто  
задаваемых вопросов из обращений в службу  
поддержки**

Направление: 09.04.01 – Информатика и вычислительная техника  
Магистерская программа: 09.04.01.15 – Технологии проектирования  
системного и прикладного программного обеспечения

Выполнил студент гр. 63501/3

\_\_\_\_\_ П.П. Жук

Научный руководитель,  
к. т. н., доц.

\_\_\_\_\_ В.М. Ицыксон

Научный консультант,  
м. т. т.

\_\_\_\_\_ М.Х. Ахин

Консультант по анализу данных,  
м.ф.-м.н.

\_\_\_\_\_ М.С. Давыдова

Консультант по нормоконтролю,  
ст. преп.

\_\_\_\_\_ С.А. Нестеров

Эта страница специально оставлена пустой.

# РЕФЕРАТ

Отчет, 73 стр., 11 рис., 12 табл., 20 ист., 2 прил.

## АНАЛИЗ ТЕКСТА, ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА, ТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ, ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

Часто задаваемые вопросы (ЧЗВ) содержат актуальную информацию о программном продукте и позволяют снизить нагрузку на отдел технической поддержки. Формирование ЧЗВ и поддержка их в актуальном состоянии требует существенных затрат от разработчика.

Описываемый в данной работе способ позволяет в автоматическом режиме выбрать наиболее релевантные для добавления в ЧЗВ вопросно-ответные пары, которые затем передаются эксперту для редактирования перед публикацией. Для этого применяются методы интеллектуального анализа текста и тематического моделирования.

Данный подход может быть применен и для других источников ИТ-дискуссий, таких как: форумы, вопросно-ответные системы. Практические результаты показывают, что используемый подход позволяет упростить формирование актуальных ЧЗВ.

# ABSTRACT

Report, 73 pages, 11 figures, 12 tables, 20 references, 2 appendices

TEXT MINING, NATURAL LANGUAGE PROCESSING, TOPIC  
MODELING, FREQUENTLY ASKED QUESTIONS

Frequently asked questions (FAQ) contains answers for typical user problems of the software product and helps to decrease amount of calls to user support department. Creating the FAQ and filling out it with the actual information is pretty time- and resource-consuming for the developer.

This work proposes the technique for automatic extraction of the most relevant for the adding in the FAQ question-answer pairs. Then extracted question-answer pairs should be validated or edited by the expert before publication. The technique is based on the text mining and topic modeling approaches.

It also could be applied for the other IT-discussions sources such as forums, question-answers systems and so on. Practical results show that this technique can be used to facilitate the creation of the FAQs.

# СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ</b> . . . . .	9
<b>1. АНАЛИЗ МЕТОДОВ ИЗВЛЕЧЕНИЯ ЧАСТО ЗАДА- ВАЕМЫХ ВОПРОСОВ</b> . . . . .	11
1.1. Существующие подходы к задаче извлечения ЧЗВ . .	11
1.2. Тематическое моделирование . . . . .	12
1.3. Методы построения тематической модели . . . . .	14
1.3.1. Кластеризация и классификация . . . . .	15
1.3.2. Латентно-семантическое индексирование . . . .	16
1.3.3. Вероятностный латентно-семантический анализ	19
1.3.4. Латентное размещение Дирихле . . . . .	20
1.3.5. Другие методы . . . . .	21
1.4. Сравнение тематических моделей . . . . .	22
1.5. Резюме . . . . .	22
<b>2. ПОСТАНОВКА ЗАДАЧИ ИЗВЛЕЧЕНИЯ ВО- ПРОСНО-ОТВЕТНЫХ ПАР</b> . . . . .	25
2.1. Анализируемые данные . . . . .	25
2.2. Формулирование требований . . . . .	27
2.3. Решаемые задачи . . . . .	27
2.4. Резюме . . . . .	29
<b>3. РАЗРАБОТКА ТЕХНОЛОГИИ ИЗВЛЕЧЕНИЯ ВО- ПРОСНО-ОТВЕТНЫХ ПАР</b> . . . . .	31
3.1. Обзор этапов подхода . . . . .	31
3.2. Предобработка данных . . . . .	32
3.2.1. Эвристики отображения . . . . .	34
3.2.2. Эвристики тематического моделирования . . .	37
3.2.3. Фильтрация обращений . . . . .	38
3.3. Тематическое моделирование . . . . .	39
3.3.1. Скрытое размещение Дирихле . . . . .	39
3.4. Формирование пар вопрос-ответ . . . . .	41
3.4.1. Дополнительная фильтрация . . . . .	41
3.4.2. Определение вопросов и ответов . . . . .	41
3.4.3. Удаление расфокусированных тем . . . . .	43
3.5. Резюме . . . . .	43

<b>4. РЕАЛИЗАЦИЯ АЛГОРИТМА ИЗВЛЕЧЕНИЯ ВОПРОСНО-ОТВЕТНЫХ ПАР . . . . .</b>	<b>45</b>
4.1. Используемые технологии . . . . .	45
4.2. Структура проекта . . . . .	46
4.3. Получение исходных данных . . . . .	47
4.4. Модель данных . . . . .	50
4.5. Взаимодействие с базой данных . . . . .	51
4.6. Реализация предобработки данных . . . . .	51
4.6.1. Фильтрация данных . . . . .	51
4.6.2. Эвристики предобработки . . . . .	52
4.7. Построение тематической модели . . . . .	54
4.7.1. Выбор реализации LDA . . . . .	54
4.7.2. Пакет org.jetbrains.zkb.lda . . . . .	55
4.8. Поиск вопросно-ответных пар . . . . .	57
4.9. Резюме . . . . .	59
<b>5. ОЦЕНКА ЭФФЕКТИВНОСТИ РАЗРАБОТАННОГО ПОДХОДА ИЗВЛЕЧЕНИЯ ВОПРОСНО-ОТВЕТНЫХ ПАР . . . . .</b>	<b>61</b>
5.1. Определение доли найденных ВОП . . . . .	61
5.2. Оценка влияния эвристик и параметров на качество ВОП . . . . .	62
5.3. Экспертная оценка . . . . .	64
5.4. Резюме . . . . .	65
<b>ЗАКЛЮЧЕНИЕ . . . . .</b>	<b>67</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ . . . . .</b>	<b>69</b>
<b>ПРИЛОЖЕНИЕ А. РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ . . . . .</b>	<b>71</b>
<b>ПРИЛОЖЕНИЕ Б. ПРИМЕРЫ ВОПРОСНО-ОТВЕТНЫХ ПАР . . . . .</b>	<b>73</b>



# СПИСОК ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

API	Application Programming Interface, программный интерфейс приложения
BSON	Binary JavaScript Object Notation, формат электронного обмена цифровыми данными, основанный на JavaScript
DSL	Domain-Specific Language, предметно-ориентированный язык
FAQ	Frequently Asked Questions, часто задаваемые вопросы
IDF	Inverse Document Frequency, обратная частота документа
JSON	JavaScript Object Notation, синтаксис для описания объектов и данных
JVM	Java Virtual Machine, виртуальная машина Java
LDA	Latent Dirichlet Allocation, скрытое размещение Дирихле
LSI	Latent Semantic Indexing, латентно-семантическое индексирование
PLSI	Probabilistic Latent Semantic Indexing, вероятностное латентно-семантическое индексирование
REST	Representational State Transfer, передача состояния представления
SVD	Singular Value Decomposition, сингулярное разложение
TF	Term Frequency, нормализованная частота слова в тексте
URI	Uniform Resource Identifier, унифицированный идентификатор ресурса
XML	eXtensible Markup Language, расширяемый язык разметки
ВОП	Вопросно-Ответная Пара
СУБД	Система Управления Базами Данных
ЧЗВ	Часто Задаваемые Вопросы



# ВВЕДЕНИЕ

Часто задаваемые вопросы (ЧЗВ) — список вопросов, которые часто возникают по какой-либо теме, и ответы на них, данные экспертами в соответствующей области. Программное обеспечение может сопровождаться ЧЗВ для помощи пользователям в решении распространенных проблем, например: Linux<sup>1</sup>, Apache Lucene<sup>2</sup>, Eclipse SWT<sup>3</sup>.

Основное преимущество ЧЗВ над пользовательской документацией и тематическими форумами — простота поиска необходимой информации. Однако создание качественных ЧЗВ — это нетривиальный процесс, требующий либо предугадывания потенциальных вопросов, либо ручного анализа обратной связи от пользователей. Целью данной работы является разработка метода извлечения ВОП из обращений в службу поддержки для упрощения задачи формирования ЧЗВ.

Предлагаемый способ, помимо обращений в службу поддержки, может быть использован и для других источников ИТ-дискуссий: форумов, вопросно-ответных систем. Сначала определяются часто обсуждаемые, повторяющиеся темы, для этого используется тематическое моделирование, а именно — скрытое размещение Дирихле (Latent Dirichlet allocation, LDA) [1], дополненное шагами пред- и постобработки, специфичными для ИТ-дискуссий. Далее среди обращений, относящихся к одной теме, с помощью косинусного расстояния и дополнительных фильтров проходит поиск вопросно-ответных пар (ВОП).

Способ извлечения ВОП, предлагаемый в данной работе является автоматическим. Однако перед публикацией в ЧЗВ необходимо провести дополнительный экспертный анализ, поскольку для извлеченных ВОП может потребоваться валидация, переформулирование или редактирование (например, удаление конфиденциальных данных, исправление грамматических ошибок). Таким образом, весь подход является полуавтоматическим.

Работа состоит из пяти разделов. В разделе 1 рассматриваются существующие подходы к задаче извлечения ЧЗВ, описываются различные тематические модели. Раздел 2 посвящен постановке задачи извлечения ЧЗВ из обращений в службу поддержки и описанию анализируемых данных. В разделе 3 описывается предлагаемый подход

---

<sup>1</sup> <http://tldp.org/FAQ/Linux-FAQ/index.html>

<sup>2</sup> <https://wiki.apache.org/lucene-java/LuceneFAQ>

<sup>3</sup> <http://www.eclipse.org/swt/faq.php>

к решению поставленной задачи. Представлена общая схема подхода, а также подробно рассмотрен каждый из этапов. В разделе 4 рассматривается разработка реализации алгоритма, соответствующего предложенному способу. Раздел 5 посвящен оценке эффективности полученного решения и качества ВОП. В этом разделе изучается влияние эвристик предобработки и параметров на выбранные метрики качества, а также приводятся результаты экспертной оценки.

# 1. АНАЛИЗ МЕТОДОВ ИЗВЛЕЧЕНИЯ ЧАСТО ЗАДАВАЕМЫХ ВОПРОСОВ

В данном разделе рассмотрены подходы к задаче извлечения ВОП. Дается определение процессу тематического моделирования, вводится понятие тематической модели. Рассмотрены различные тематические модели и определяется наиболее эффективная из них.

## 1.1. Существующие подходы к задаче извлечения ЧЗВ

Предварительным этапом данной работы стало изучение существующих методик извлечения вопросно-ответных пар. Были изучены различные научные статьи за период 2004–2016 гг., выявлены следующие методы:

- Методы, основанные на машинном обучении — [2], [3];
- Методы, основанные на классификации — [4], [5], [6];
- Методы на основе тематического моделирования — [7], [8], [9].

Методы, применяющие тематическое моделирование, являются наиболее предпочтительными, поскольку построение и использование тематической модели позволяет находить ВОП, у которых в вопросе и ответе используется схожая терминология. Это позволяет находить более качественные ВОП [8] по сравнению со способами, использующими машинное обучение или классификацию.

Работа [9] предлагает решать задачу извлечения ВОП в три шага: предобработка данных, тематическое моделирование, поиск вопросно-ответных пар. От других работ, использующих тематическое моделирование, эту работу отличает наличие дополнительных шагов обработки данных, специфичных для ИТ-дискуссий. При решении поставленной задачи именно эта работа применялась в качестве основной. Работа [9] также использует модель скрытого размещения Дирихле (LDA), которая более подробно рассмотрена далее в этом разделе.

Работы [7] и [8] также используют LDA для вопросно-ответных систем. Работа [7], однако, предлагает проводить тематическое моделирование в рамках одного обращения, что может дать менее качественный результат, поскольку LDA показывает лучшие результаты

на больших объемах данных. В работе [8] LDA используется для определения темы вновь поступивших вопросов, при этом для них не определяется ответ. В статьях [10] и [11] рассмотрены модификации LDA, призванные улучшить качество тематического моделирования.

В статье [2] используется размеченный корпус текстов для обучения модели, определяющей ВОП. В текущей работе отсутствуют размеченные данные, что не позволяет применить описанный в статье [2] подход. Работа [3] использует графовые модели для извлечения вопросно-ответных пар.

Работа [6] предлагает способ для нахождения лучшего ответа на вопрос среди уже предоставленных на примере размеченных данных со Stack Overflow. В текущей работе не используются размеченные данные, что позволяет получить более универсальное решение. В статье [5] представлен другой способ поиска ответов, связанный с использованием поисковой системы. Сначала комментарии разделяются на 5 классов: вопрос, уточнение, ответ, отзыв на ответ, шум. Затем используется специально настроенная поисковая система для поиска только по ответам. Основное отличие от текущей работы заключается в способе определения релевантных ответов.

## 1.2. Тематическое моделирование

**Тематическое моделирование** — способ построения модели коллекции текстовых документов, которая определяет, к каким темам относится каждый из документов [12]. Впервые задача тематического моделирования возникла в 1958 году, когда Герхард Лисовски и Леонард Рост завершили работу по составлению каталога религиозных текстов на иврите, которые должны были помочь учёным определить значения давно утраченных терминов. Затем они собрали вместе все возможные контексты, в которых появлялся каждый из терминов. Следующей задачей было научиться игнорировать несущественные различия в формах слов и выделять те различия, которые влияют на семантику. Замыслом авторов было дать возможность исследователям языка проанализировать различные отрывки и понять семантику каждого термина в его контексте.

Проблемы такого рода возникают и сегодня при автоматическом анализе текстов. Одна и та же концепция может выражаться любым количеством различных терминов (синонимия), тогда как один тер-

мин часто имеет разные смыслы в различных контекстах (полисемия). Таким образом, необходимы способы различать варианты представления одной концепции и определять конкретный смысл многозначных терминов. Теоретически обоснованным и активно развивающимся направлением в анализе текстов на естественном языке, призванным решать перечисленные задачи, является тематическое моделирование коллекций текстовых документов.

Построение тематической модели может рассматриваться как задача одновременной кластеризации документов и слов по одному и тому же множеству кластеров, называемых темами. В терминах кластерного анализа тема — это результат би-кластеризации, то есть одновременной кластеризации и слов, и документов по их семантической близости. Обычно выполняется нечёткая кластеризация, то есть документ может принадлежать нескольким темам в различной степени. Таким образом, сжатое семантическое описание слова или документа представляет собой вероятностное распределение на множестве тем. Процесс нахождения этих распределений называется тематическим моделированием.

**Тематическая модель** (англ. topic model) коллекции текстовых документов определяет, к каким темам относится каждый документ и какие слова (термины) образуют каждую тему [13].

Переход из пространства терминов в пространство найденных тематик помогает эффективнее решать такие задачи, как тематический поиск, классификация и аннотация коллекций документов и новостных потоков.

Тематическое моделирование как вид статистических моделей для нахождения скрытых тем встреченных в коллекции документов, нашло свое применение в таких областях как машинное обучение и обработка естественного языка. Исследователи используют различные тематические модели для анализа текстов, текстовых архивов документов, для анализа изменения тем в наборах документов. В документах, посвященных одной теме, можно встретить некоторые слова чаще других.

Например: «собака» и «кость» встречаются чаще в документах про собак, «кошки» и «молоко» будут встречаться в документах о кошках, предлоги «и» и «в» будут встречаться в обеих тематиках. Обычно документ касается нескольких тем в разных пропорциях, таким образом, документ, в котором 10% темы составляют кошки, а 90% темы — собаки, можно предположить, что слов про собак в 9

раз больше. Тематическое моделирование отражает эту интуицию в математическую структуру, которая позволяет на основании изучения коллекции документов и исследования частотных характеристик слов в каждом документе сделать вывод, что каждый документ — это некоторый баланс тем.

Как правило, количество тем, встречающихся в документах, меньше количества различных слов во всем наборе. Поэтому скрытые переменные (темы) позволяют представить документ в виде вектора в пространстве скрытых (латентных) тем вместо представления в пространстве слов. В результате документ имеет меньшее число компонент, что позволяет быстрее и эффективнее его обрабатывать. Таким образом, тематическое моделирование также может использоваться в таком классе задач, как уменьшение размерности данных. Кроме того, найденные темы могут использоваться для семантического анализа текстов.

### 1.3. Методы построения тематической модели

Задача извлечения скрытых тем из коллекции текстовых документов имеет множество применений:

- Кластеризация, классификация, ранжирование, аннотирование и суммаризация отчётов, научных публикаций, архивов документов и т.д.;
- Тематический поиск документов и связанных с ними объектов;
- Фильтрация спама;
- Построение тематических профилей пользователей форумов, блогов и социальных сетей для поиска тематических сообществ и определения наиболее активных их участников;
- Анализ новостных потоков и сообщений из социальных сетей для определения актуальных событий реального мира и реакции пользователей на них.

Тематическое моделирование позволяет автоматически систематизировать и обрабатывать электронные архивы такого масштаба, который человек не в силах обработать. С точки зрения поставленной в данной работе задачи, тематическое моделирование используется для построения тематической модели корпуса обращений в службу поддержки. Затем в рамках одного обращения определяется пара комментариев, наиболее близкая к соответствующей данному обращению



теме. Таким образом учитывается сходство терминологии между задаваемым вопросом и полученным ответом на него.

### 1.3.1. Кластеризация и классификация

Задача определения и отслеживания тем возникла в 1996-1997 годах. В работе [14] понятие темы тесно связано с понятием события: тема — это событие или действие вместе со всеми непосредственно связанными событиями и действиями. Задача заключается в извлечении событий из потока информации. Для представления документов принято пользоваться векторной моделью, в которой каждому слову сопоставляется вес в соответствии с выбранной весовой функцией. Располагая таким представлением для всех документов, можно, например, находить расстояние между документами и тем самым решать задачу подбора — чем ближе расположены точки, тем больше похожи соответствующие документы. Классическим методом назначения весов словам является TF-IDF:

$$TFIDF(t, d, D) = TF(t, d) * IDF(t, D) \quad (1.1)$$

TF (term frequency) — нормализованная частота слова в тексте:

$$TF(t, d) = \frac{freq(t, d)}{\max_{w \in D} freq(w, d)} \quad (1.2)$$

Здесь  $freq(t, d)$  — число вхождений слова  $t$  в документ  $d$ .

IDF (inverse document frequency) — обратная частота документов:

$$IDF(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \quad (1.3)$$

Здесь в числителе — количество документов в наборе, а в знаменателе — количество документов, в которых встречается слово  $t$ . В зависимости от решаемой задачи используются различные модификации TF-IDF.

Для сравнения векторов документов в [14] применялись такие метрики, как косинус, дивергенция Кульбака-Лейблера и другие методы (взвешенная сумма компонентов документа, простые языковые модели). Всего существует более 70 способов расчёта схожести векторов [15]. В [14] рассматривается два типа задач: обнаружение событий

из набора данных за определенный период времени и обнаружение событий в режиме реального времени.

Первый тип задач заключается в разбиении исходных данных на группы, соответствующие событиям, а также в определении, описывает ли текстовый документ из набора какое-либо событие. Основной идеей всех решений было использование алгоритмов кластеризации (инкрементальная кластеризация, метод К-средних и др). При этом предполагается, что каждый кластер содержит документы, описывающие какое-либо событие. Задача второго типа — для нового документа определить, описывает ли он событие, которое уже встречалось в исходных данных. Для отслеживания событий использовались алгоритмы классификации (метод k-ближайших соседей, решающие деревья и др). Классификация производилась с использованием двух классов: YES — документ описывает событие, NO — не описывает.

Таким образом, в ранних исследованиях тема отождествлялась с событием. В реальной жизни тема может описывать иные сущности, а не только события. Поэтому в более поздних работах задачи определения событий и тем стали различаться. Еще один недостаток описанных методов состоит в том, что анализируемые документы относятся только к одной теме или событию, однако один документ может затрагивать несколько тем. К тому же, векторное представление документов не позволяет разрешать синонимию и полисемию терминов.

Для решения перечисленных проблем было предложено рассматривать набор векторов терминов из документов как общую терм-документную матрицу и применять к ней особые разложения (метод LSI).

### **1.3.2. Латентно-семантическое индексирование**

В 80-е годы прошлого столетия стали активно развиваться системы информационного поиска по коллекциям документов разнообразной природы. Первыми были реализованы подходы, основанные на поиске точных совпадений частей документов с запросами пользователей. Довольно скоро, однако, стало очевидно различие между релевантностью (соответствием) документа запросу и точным совпадением их частей. Зачастую документы, релевантные запросу с точки зрения пользователя, не содержали терминов из запроса и поэтому не отображались в результатах поиска (проблема синонимии).

С другой стороны, большое количество документов, слабо или

вовсе не соответствующих запросу, показывались пользователю только потому, что содержали термины из запроса (проблема полисемии). Самым простым решением этих проблем кажется добавление к запросу уточняющих терминов для более точного описания интересующего контекста. Однако предположение о том, что индекс поисковой системы содержит все возможные уточняющие термины, на практике выполняется довольно редко.

В 1988 г. был предложен метод латентно-семантического индексирования (latent semantic indexing, LSI), призванный повысить эффективность работы информационно-поисковых систем путём проецирования документов и терминов в пространство более низкой размерности, которое содержит семантические концепции исходного набора документов.

Основная идея метода состоит в оценке корреляции терминов путём анализа их совместной встречаемости в документах. К примеру, в коллекции всего 100 документов, содержащих термины «доступ» и/или «поиск». При этом только 95 из них содержат оба термина вместе. Логично предположить, что отсутствие термина «поиск» в документе с термином «доступ» ошибочно и возвращать данный документ по запросу, содержащему только термин «поиск». Подобные выводы можно делать не только из простой попарной корреляции терминов.

С другой стороны, анализируя корреляцию терминов в запросе, можно более точно определять интересующий пользователя смысл основного термина и повышать позиции документов, соответствующих этому смыслу, в результатах поиска.

Таким образом, при латентно-семантическом индексировании документов задача состоит в том, чтобы спроецировать часто встречающиеся вместе термины в одно и то же измерение семантического пространства, которое имеет пониженную размерность по сравнению с оригинальной терм-документной матрицей, которая обычно довольно разрежена. Элементы этой матрицы содержат веса терминов в документах, назначенные с помощью выбранной весовой функции. В качестве примера можно рассмотреть самый простой вариант такой матрицы, в которой вес термина равен 1, если он встретился в документе (независимо от количества появлений), и 0 если не встретился (таблица 1.1).

Наиболее распространенный вариант LSI основан на использовании разложения терм-документной матрицы по сингулярным значениям — сингулярном разложении (Singular Value Decomposition, SVD).

Таблица 1.1. Терм-документная матрица

	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
voyage	1	0	0	1	1	0
trip	0	0	0	1	0	1

Согласно теореме о сингулярном разложении, любая вещественная прямоугольная матрица может быть разложена на произведение трех матриц:

$$A = TSD^T \quad (1.4)$$

где матрицы  $T$  и  $D$  — ортогональные, а  $S$  — диагональная матрица, элементы на диагонали которой называются *сингулярными значениями* матрицы  $A$ . Такое разложение обладает особенностью: если в матрице  $S$  оставить только  $k$  наибольших сингулярных значений, а в матрицах  $T$  и  $D$  — только соответствующие этим значениям столбцы, то произведение получившихся матриц  $S$ ,  $T$  и  $D$  будет наилучшим приближением исходной матрицы  $A$  к матрице  $\hat{A}$  ранга  $k$ .

Если в качестве матрицы  $A$  взять терм-документную матрицу, то матрица  $\hat{A}$ , содержащая только  $k$  первых линейно независимых компонент  $A$ , отражает основную структуру различных зависимостей, присутствующих в исходной матрице. Структура зависимостей определяется весовыми функциями терминов.

Таким образом, каждый термин и документ представляются при помощи векторов в общем семантическом пространстве размерности  $k$ . Близость между любой комбинацией терминов и/или документов вычисляется при помощи скалярного произведения векторов. Для задач информационного поиска запрос пользователя рассматривается как набор терминов, который проецируется в семантическое пространство, после чего полученное представление сравнивается с представлениями документов в коллекции. Как правило, выбор  $k$  зависит от поставленной задачи и подбирается эмпирически. Если выбранное значение  $k$  слишком велико, то метод теряет свою мощностъ и приближается по характеристикам к стандартным векторным методам. Слишком маленькое значение  $k$  не позволяет улавливать различия между

похожими терминами или документами.

Следующим этапом развития тематического моделирования стали подходы, позволяющие моделировать вероятности скрытых тем в документах и терминов в темах. В отличие от дискриминативных подходов (к которым относится LSI), в вероятностных подходах сначала задаётся модель, а затем с помощью терм-документной матрицы оцениваются её скрытые параметры, которые затем могут быть использованы для генерации моделируемых распределений. Из этого следуют преимущества вероятностного моделирования документов:

- Результаты работы представляются в терминах теории вероятностей и поэтому могут быть с минимальными затратами встроены в другие вероятностные модели и проанализированы стандартными статистическими методами;
- Новые порции входных данных не требуют повторного обучения модели;
- Вероятностные модели могут быть расширены путём добавления переменных, а также новых связей между наблюдаемыми и скрытыми переменными.

### 1.3.3. Вероятностный латентно-семантический анализ

Вероятностное латентно-семантическое индексирование (Probabilistic Latent Semantic Indexing, PLSI) является одной из первых вероятностных моделей тематического моделирования, предложенной Томасом Хоффманом в 1999 году. В основе PLSI лежит так называемая аспектная модель, которая связывает скрытые переменные тем  $z \in Z = \{z_1, \dots, z_k\}$  с каждой наблюдаемой переменной — словом или документом. Таким образом, каждый документ может относиться к нескольким темам с некоторой вероятностью, что является отличительной особенностью этой модели по сравнению с подходами, не позволяющими вероятностного моделирования.

PLSI использует следующий генеративный процесс:

1. Выбрать документ  $d$  согласно распределению  $p(d)$ ;
2. Выбрать тему  $i \in \{1, \dots, k\}$  на основе распределения  $\theta_{di} = p(z = i|d)$ ;
3. Выбрать слово  $v$  — значение переменной  $w$  на основе распределения  $\phi_{iw} = p(w = v|z = i)$ ;

Совместная вероятностная модель над документами и словами определена следующим образом:

$$P(d, w) = P(d) \sum_{z \in Z} P(w|z)P(z|d) \quad (1.5)$$

Несмотря на очевидные преимущества перед более ранними подходами, модель PLSI имеет следующие недостатки. Во-первых, она содержит большое число параметров, которое растёт в линейной зависимости от числа документов. Как следствие, модель склонна к переобучению и неприменима к большим наборам данных. Во-вторых, невозможно вычислить вероятность документа, которого нет в наборе данных. В-третьих, отсутствует какая-либо закономерность при генерации документов из сочетания полученных тем. Данные недостатки устранены в модели LDA.

#### 1.3.4. Латентное размещение Дирихле

Ввиду недостатков, которыми обладала модель PLSI, её использование было весьма ограниченным. Поэтому в [1] была предложена модель латентного размещения Дирихле (latent Dirichlet allocation, LDA), лишённая недостатков PLSI. В LDA предполагается, что каждое слово в документе порождено некоторой латентной темой, при этом в явном виде моделируется распределение слов в каждой теме, а также априорное распределение тем в документе.

Темы всех слов в документе предполагаются независимыми. В LDA, как и в PLSI, документ может соответствовать не одной теме. Но LDA задаёт модель порождения как слов, так и документов, поэтому появляется дополнительная возможность оценивать вероятности документов вне текстовой коллекции с помощью алгоритма вариационного вывода и семплирования Гиббса [16].

В отличие от PLSI, в LDA число параметров не увеличивается с ростом числа документов в коллекции. Многочисленные расширения модели LDA устраняют некоторые её ограничения и улучшают производительность для конкретных задач. Каждый документ генерируется независимо:

1. Случайно выбрать для документа  $d$  его распределение по темам  $\theta_d$ ;
2. Для каждого слова в документе:

- а. Случайно выбрать тему из распределения  $\theta_d$ , полученного на 1-м шаге;
- б. Случайно выбрать слово из распределения слов  $\phi_t$  в выбранной теме  $t$ .

LDA описывает каждую тему  $t$ , как вероятностное распределение по всем словам из входных данных ( $\phi_t$ ). Каждый документ  $d$  описывается вероятностным распределением по темам ( $\theta_d$ ). Цель LDA - максимизировать функцию (1.6) путем оптимизации  $\phi$  и  $\theta$ :

$$P(\theta, \phi) = \prod_{t=1}^T P(\phi_t) \prod_{d=1}^D P(\theta_d) \prod_{w=1}^{W_d} P(Z_{d,w}|\theta_d)P(N_{t,w}|\phi_t) \quad (1.6)$$

где  $T$  — количество тем,  $D$  — количество документов,  $W_d$  — количество различных слов в документе  $d$ ,  $Z_{d,w}$  — определяет принадлежность слова  $w$  к документу  $d$  и  $N_{t,w}$  — принадлежность слова  $w$  к теме  $t$ .

В модели LDA предполагается, что параметры  $\theta$  и  $\phi$  в свою очередь зависят от гиперпараметров  $\alpha$  и  $\beta$  соответственно и распределены следующим образом:  $\theta \sim Dir(\alpha)$ ,  $\phi \sim Dir(\beta)$  где  $\alpha$  и  $\beta$  — задаваемые вектора-параметры (гиперпараметры) распределения Дирихле (обычно  $\alpha$  принимается равным  $50/T$ , а параметр  $\beta = 0.1$ , увеличение  $\beta$  ведёт к более разреженным тематикам).

Для оценки параметров LDA используется сэмплирование по Гиббсу, которое состоит в том, чтобы на каждом шаге фиксировать все переменные, кроме одной, и выбирать оставшуюся переменную согласно распределению вероятности этой переменной при условии всех остальных.

### 1.3.5. Другие методы

LDA на сегодняшний день является наиболее распространенной моделью, подходящей для решения большинства задач тематического моделирования. Однако, существует ряд специализированных моделей, которые коротко представлены ниже. Более подробно данные модели рассмотрены в [12].

*Автор-тематическая модель* (author-topic model) представляет

собой расширение LDA для совместного описания документов и авторов.

*Скрытая тематическая модель гипертекста* (latent topic hypertext model, LTHM) описывает закон порождения ссылок в корпусе гипертекстов.

В рамках *композиционной модели HMM-LDA* строится совместное описание синтаксиса и семантики текста. Скрытая марковская модель (HMM) описывает локальные закономерности между соседними словами, тогда как модель LDA даёт глобальное тематическое описание документа в целом.

*Модель соответствий* (correspondence LDA, Corr-LDA) была исходно предложена для решения задачи аннотирования изображений, когда каждому изображению из обучающей выборки ставится в соответствие некоторое множество слов, и требуется сгенерировать список слов, подходящих к новому, ещё не аннотированному изображению.

Аналогичная вероятностная модель *labeled LDA* (LLDA) предложена для кластеризации генов.

*Иерархический процесс Дирихле* (Hierarchical Dirichlet Process, HDP) является Байесовской непараметрической моделью, которая может быть использована для тематического моделирования с потенциально бесконечным числом тем.

## 1.4. Сравнение тематических моделей

В таблице 1.2 приведено сравнение описанных ранее тематических моделей. В таблицу не включены различные модификации LDA, поскольку такие модели специфичны для отдельных задач и не интересны в контексте данной работы.

Модель LDA является вероятностной моделью и позволяет определять вероятность принадлежности документа к теме. Данная модель также может использоваться для определения распределения тем для новых документов без повторного обучения. Воспользуемся этой моделью для решения поставленной задачи.

## 1.5. Резюме

В ходе данного раздела были рассмотрены существующие методики извлечения часто задаваемых вопросов, описаны их преимущества



Таблица 1.2. Сравнение тематических моделей

Модель	Вероятн. модель	Преимущества	Недостатки
Класт. и классиф.	—	Может быть использована любая подходящая метрика.	Не учитывается синонимия и полисемия.
LSI	—	Снимается проблема синонимии и полисемии.	Низкая скорость работы. Результат зависит от подбора числа значимых терминов.
PLSI	+	Документ содержит более одной темы.	Не подходит для определения тем в новых документах.
LDA	+	Подходит для определения тем новых документов. Документ содержит более одной темы.	Требует указания количества тем.

и недостатки, приведены существующие научные работы в соответствующей предметной области. Для решения поставленных в данной работе задач (раздел 2) был предложен подход, использующий тематическую модель текста. Этот метод позволяет получить более качественные результаты, поскольку, в отличие от остальных подходов, учитывает сходство терминологии между вопросом и ответом.

В разделе также рассмотрены существующие тематические модели, приводится их сравнение. Для использования выбирается модель LDA, как наиболее эффективная на текущий момент.



## 2. ПОСТАНОВКА ЗАДАЧИ ИЗВЛЕЧЕНИЯ ВОПРОСНО-ОТВЕТНЫХ ПАР

Цель данной работы — разработка подхода по извлечению ВОП из обращений в службу поддержки, пригодных для добавления в ЧЗВ.

Использование такого подхода в рабочем процессе технической поддержки позволит:

1. Упростить заполнение ЧЗВ и документации;
2. Уменьшить количество типовых обращений в службу поддержки;
3. Уделять больше внимания нетривиальным обращениям и задачам.

Команда технической поддержки и команда технических писателей просматривают весь объем пользовательских обращений с целью, например, найти информацию, актуальную для добавления в FAQ, в раздел по устранению неполадок в технической документации или для поиска ответа на обращение. Добавление автоматизации в этот процесс упростит поддержание ЧЗВ и соответствующего раздела в документации в актуальном состоянии, что уменьшит количество типовых обращений в ТП и высвободит дополнительные ресурсы для нетривиальных обращений и задач.

Стоит отметить, что задача не решается полностью автоматически, поскольку текст извлеченных ВОП может содержать персональные данные или грамматические ошибки. Текст такой вопросно-ответной пары должен быть отредактирован перед публикацией. В связи с чем возникает необходимость проведения экспертного анализа — ручного этапа работы. Весь подход при этом является полуавтоматическим.

### 2.1. Анализируемые данные

В данной работе для анализа использовались обращения пользователей в техническую поддержку системы отслеживания ошибок

YouTrack<sup>1</sup>. Для взаимодействия с пользователями команда YouTrack использует Zendesk<sup>2</sup> — систему учета и обработки пользовательских обращений.

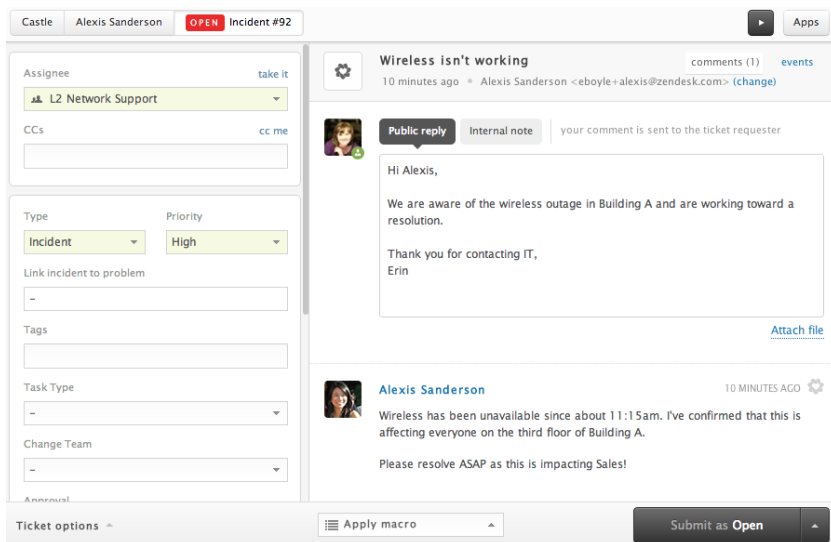


Рисунок 2.1. Пример обращения в системе Zendesk

Zendesk позволяет настраивать различные каналы для взаимодействия с пользователями: электронная почта, социальные сети, форма для прямой отправки обращений и так далее. Все собранные таким образом обращения отображаются в едином интерфейсе. На рисунке 2.1 приведен пример обращения в системе Zendesk.

Обращения состоят из комментариев и, в общем случае, представляют собой диалог между клиентом и сотрудником технической поддержки. Поскольку Zendesk агрегирует все поступающие обращения, то мы не можем делать предположений об их разбиении по темам. То есть заранее неизвестно, какие из обращений относятся, например, к проблемам администрирования YouTrack, а какие — связаны с пользовательским интерфейсом.

<sup>1</sup> <http://jetbrains.ru/products/youtrack/>

<sup>2</sup> <https://www.zendesk.com>

В Zendesk каждое обращение содержит ряд метаданных. В данной работе использовалась следующая метайнформация: статус обращения, авторство комментария. Всего в работе анализируются 6500 обращений за период с января 2016 года по март 2017 года.

## 2.2. Формулирование требований

Разрабатываемый метод должен соответствовать следующим условиям:

- Приоритет качества над количеством;
- Обработка только обращений на английском языке;
- Вопрос и ответ всегда состоят из одного комментария.

В работе уделяется большее внимание поиску качественных ВОП (точность), чем поиску всех возможных ВОП (полнота). Основная мотивация такого решения заключается в желании сократить до минимума ручную часть алгоритма — валидацию и редактирование ВОП.

Для анализа выбраны обращения на английском языке, поскольку доля таких обращений в техподдержку YouTrack составляет 89%.

В качестве ответа всегда выбирается один комментарий. Различные комментарии не комбинируются для составления ответа, поскольку если ответ на вопрос (или сам вопрос) содержится в более чем одном комментарии, то, вероятно, обращение имеет одну из следующих проблем: вопрос плохо сформулирован, вопрос слишком специфичен или ответ недостаточно полон. Такие ВОП не подходят для добавления в ЧЗВ.

Стоит отметить, что предложенный подход не ограничивается частыми вопросами и позволяет находить редкие ВОП, если они хорошо сформулированы и имеют корректный ответ.

## 2.3. Решаемые задачи

Как было установлено в разделе 1, для извлечения ВОП из большого объема текстовых данных эффективно использовать методы на основе тематического моделирования. В разделе 1 также было установлено наиболее подходящая для данной задачи тематическая модель — LDA.

Модель LDA используется для решения похожей задачи в работе [9]. В статье [9] решается задача извлечения ВОП из списков рассылки, посвященных различным программным продуктам с открытым исходным кодом. Дополнительно в этой работе применяется ряд эвристик, учитывающих ИТ-тематику анализируемых данных. Воспользуемся этой статьей в качестве опорной.

Далее представлен список решаемых в данной работе задач:

1. Предобработка данных:

- Фильтрация обращений;
- Удаление шумов;

2. Кластеризация обращений по тематикам:

- Тематическое моделирование, LDA;

3. Извлечение ВОП:

- Поиск вопросов;
- Поиск ответов;

4. Оценка эффективности:

- Экспертный анализ.

По сравнению с опорной статьей в текущей работе доработаны и расширены эвристики предобработки данных, применены дополнительные фильтры до и после тематического моделирования. Анализируемые данные — обращения в службу технической поддержки — никак не размечены, в то время как в статье [9] тот или иной список рассылки представляет собой заранее известную тему. Кроме того используется перплексия [1] для приблизительного определения количества тем при тематическом моделировании.

Этапы 1–3 подробно рассмотрены в разделах 3 и 4. Оценка эффективности и результаты экспертного анализа приведены в разделе 5.

## 2.4. Резюме

В данном разделе были сформулированы требования к разрабатываемому методу извлечения ВОП. Определены основные этапы разработки, соответствующие поставленным задачам. Были рассмотрены анализируемые в данной работе данные, которые также используются для определения эффективности подхода.





## 3. РАЗРАБОТКА ТЕХНОЛОГИИ ИЗВЛЕЧЕНИЯ ВОПРОСНО-ОТВЕТНЫХ ПАР

Этот раздел посвящен разработке технологии извлечения вопросно-ответных пар, которая включает в себя следующие этапы:

- Предобработка данных;
- Кластеризация обращений по тематикам;
- Извлечение ВОП.

Далее дается краткое описание подхода в целом, затем детально описывается каждый из этапов.

### 3.1. Обзор этапов подхода

**Этап 1 — предобработка данных.** На данном этапе происходит подготовка обращений к отображению в ЧЗВ. Сырые данные содержат много шума: HTML разметка, заголовки электронной почты (например, «01 июля 2001 г., 10:10 пользователь ... написал:»), приветствия, благодарности и так далее. Этот шум не имеет прямого отношения к проблемам пользователей, что приводит к формированию тем, не отражающих действительного содержания обращений, и заметно влияет на качество алгоритма в целом.

К шуму также относятся фрагменты программного кода, логи и трассировки стека. Для таких данных характерен часто повторяющийся, небольшой набор слов, что также ведет к формированию тем, описывающих программный код или логи, а не пользовательские проблемы.

Для повышения качества результирующих ВОП применяется ряд эвристик предобработки, которые значительно доработаны в сравнение со статьей [9]. При этом также фильтруются обращения, которые заведомо не могут содержать вопроса или ответа.

**Этап 2 — кластеризация обращений по тематикам.** Для определения кластеров связанных обращений (в дальнейшем — тем) используется скрытое размещение Дирихле. LDA описывает каждую тему как вероятностное распределение по набору всех слов и для каждого обращения определяет вероятностное распределение по темам.

Затем каждому обращению сопоставляется тема с наибольшей вероятностью. Для каждой темы определяется мешок слов (bag-of-words), как совокупность слов всех документов, принадлежащих данной теме.

Используется перплексия для приблизительного определения количества тем. Точного определения количества тем не требуется, поскольку этап 3 включает в себя удаление расфокусированных тем. В конце данного этапа темы проходят через фильтр, с целью удаления незначимых с точки зрения ЧЗВ тем.

**Этап 3 — формирование ВОП.** Для каждого комментария в рамках обращения считается метрика близости между текстом комментария и соответствующей темой. На основе этой метрики определяются хорошо сформулированные вопросы и релевантные ответы на них. Удаляются расфокусированные темы — темы, не отражающие ни одну из реальных тем анализируемых данных.

На рисунке 3.1 изображен обобщенный сценарий поиска и извлечения ВОП, включающий все описанные выше этапы, а также этап экспертной оценки.

## 3.2. Предобработка данных

Исходные данные представляют собой 6500 обращений, собранных из различных каналов поступления обращений с помощью системы автоматизации запросов клиентов Zendesk. Каждое обращение содержит ряд метаданных. В то время как использование метаданных ограничивает область применения алгоритма, это позволяет повысить его качество. В данной работе использовалась метайнформация, широко распространенная для данных такого рода: статус обращения и авторство комментария.

Предполагается, что ответ на вопрос всегда содержится в одном комментарии, поэтому несколько комментариев не объединяются для создания ответа (секция 2.2). Для анализа использовались только обращения на английском языке.

Из входных данных были отфильтрованы обращения только со статусом "закрыто" и "выполнено". Данные статусы говорят о том, что обращения имеют окончательный набор комментариев, обращения с другим статусом еще могут находиться в активном обсуждении.

Эвристики предобработки делятся на 2 категории: эвристики отображения и эвристики тематического моделирования. Первые

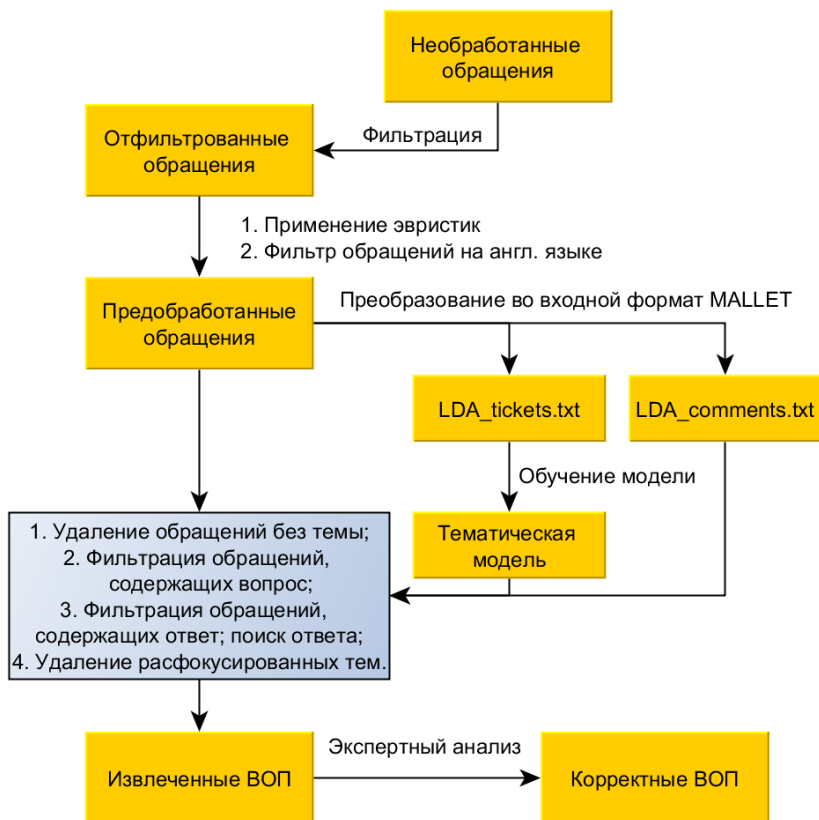


Рисунок 3.1. Обобщенная схема поиска ВОП

предназначены для приведения обращений к виду, максимально близкому к виду ЧЗВ, вторые — применяются поверх первых и создают отдельное представление, используемое в LDA.

Для наглядности в таблице 3.1 приведен простой пример необработанного комментария, полученного от пользователя через электронную почту, в таблице 3.2 для данного примера можно увидеть результаты работы эвристик обоих типов.

Таблица 3.1. Пример комментария

Hello JetBrains,  
I want to configure youtrack over SSL but not able  
to find any solution or article on the subject. Could  
you help me?

Thanks in advance

Some Name  
Director / CEO  
+12 34 567890  
mail@companyname.com  
<https://www.companyname.com/>

### 3.2.1. Эвристики отображения

*Эвристика 1 (специфичные регулярные выражения):* данная эвристика направлена на удаление фрагментов, зависящих от предметной области или используемого программного обеспечения. Например: информация, добавляемая системой управления обращениями; шаблоны оформления обращений через веб-форму, содержащие дополнительные поля (имя, e-mail, компания); и так далее.

Регулярные выражение, соответствующие этой эвристике, а также другие, используемые в данной работе регулярные выражения, представлены в приложении А.

*Эвристика 2 (удаление цитат электронной почты):* 33% обращений созданы через электронную почту. Комментарии в таких обращениях часто цитируют предыдущее сообщение. Как следствие, в тексте появляются дубликаты, которые приводят к искажению тематической модели. Механизм определения цитат, используемый Zendesk, не всегда справляется со своей задачей. Каждый четвертый комментарий содержит цитату электронной почты. Для их удаления использовалась самостоятельно разработанная библиотека email-parser<sup>1</sup>.

Библиотека email-parser позволяет удалять цитаты из электронных писем с точностью выше 97% вне зависимости от используемого почтового клиента или языка письма. Библиотека разрабатывалась

---

<sup>1</sup> <https://github.com/JetBrains/email-parser>

за рамками данного дипломного проекта, однако большая доля электронных писем в обращениях делает её использование актуальным.

*Эвристика 3 (удаление общих суффиксов):* большинство пользователей, как правило, имеют подпись, которая добавляется в конец каждого отправленного ими сообщения. Процедура для определения и удаления таких подписей приводится ниже (рисунок 3.2).

**Исходные параметры:** `commentsList` - список комментариев

**Результат:** `modifiedCommentsList` - модифицированный список комментариев

`suffixLength`  $\leftarrow$  инициализация массива длины

`commentsList.size` нулями;

**цикл**  $i \leftarrow 1$  **до** `commentsList.size - 1` **выполнять**

**цикл**  $j \leftarrow i + 1$  **до** `commentsList.size` **выполнять**

`suffix`  $\leftarrow$  определить общий суффикс для

`commentsList[i]` и `commentsList[j]`

**пока** `suffix[1]` не символ новой строки **выполнять**

        | удалить первый символ `suffix`;

**конец**

`update(suffix, i);`

`update(suffix, j);`

**конец**

**конец**

**цикл**  $i \leftarrow 1$  **до** `commentsList.size` **выполнять**

    | удалить из `commentsList[i]` последние `suffix[i]` символов;

**конец**

**функция** `update(suffix, x)` :

**если** `suffix.length`  $\geq$  `suffixLength[x]` **тогда**

    | `suffixLength[x]`  $\leftarrow$  `suffix.length`;

**конец**

**конец**

Рисунок 3.2. Описание процедуры удаления общих суффиксов

Для удаления подписей предлагается следующее:

- Для всех комментариев в исходных данных попарно посчитать общий суффикс;

- У каждого комментария удалить суффикс максимальной длины;

Суффикс определяется построчно, что позволяет избежать частичного удаления абзацев с полезной информацией.

*Эвристика 4 (короткие абзацы):* многие сообщения начинаются со слов приветствия и заканчиваются словами благодарности (рисунок 2.1, таблица 3.1). Как правило, эти фрагменты выделены в отдельные абзацы (отделены символом новой строки) и значительно короче основной части сообщения (20-25 символов против 300-500). Данная эвристика удаляет (при наличии) один короткий начальный абзац и все короткие конечные абзацы. Абзац является коротким, если он состоит из 3 или меньше слов. Это число было определено эмпирически. Дополнительно этот шаг позволяет удалить фрагменты подписи, оставшиеся после эвристики 3.

Подписи в основном состоят из нескольких коротких строк (таблица 3.1), которые в большинстве случаев попадают под данную эвристику. Поэтому для последних строчек текста данную эвристику имеет смысл применять многократно. С другой стороны, приветствие встречается в начале текста не более одного раза. Поэтому здесь эвристика применяется один раз во избежание удаления полезной информации.

Таблица 3.2. Эффект применения эвристик

Версия для ЧЗВ	Версия для LDA
I want to configure youtrack over SSL but not able to find any solution or article on the subject	configure SSL solution article subject

*Эвристика 5 (частые предложения):* данная эвристика была взята из статьи [9] и говорит о том, что предложения, встречающиеся на всем наборе обращений более 15 раз, не содержат информации, специфичной для конкретного вопроса. К таким фразам могут относиться: «Моя проблема заключается в следующем», «Благодарим вас за обращение» и так далее.

Стоит отметить, что учет предложений, состоящих из одного слова, или игнорирование регистра текста приводит к частичному удалению предложений и, как следствие, ухудшению внешнего вида ВОП.

Как результат применения описанных выше эвристик, текст комментариев часто может начинаться с нижнего регистра (ввиду удаления приветствий) и содержать лишние пустые строки, что снижает читаемость. Данные недостатки следует исправить, так как именно в таком виде ВОП будут показываться экспертам. Для этого необходимо удалить пустые строки и лишние пробелы, а также перевести первый не пробельный символ в верхний регистр.

### 3.2.2. Эвристики тематического моделирования

Эвристики из данной группы применяются с целью повышения качества LDA. Поскольку при этом теряется часть информации, необходимой для отображения ЧЗВ, необходимо сохранять две версии для каждого из комментариев, как показано в таблице 3.2.

*Эвристика 6 (удаление пользовательских данных):* пользовательские данные ухудшают качество LDA. Например, тема, включающая в себя имя некоторого пользователя, будет содержать обращения, в которых часто встречается это имя, несмотря на то, что сами обращения могут относиться к разным подсистемам программного продукта и, соответственно, иметь различные темы.

На этом этапе предлагается удалять такие фрагменты текста, как: унифицированные идентификаторы ресурса (URI), пути в файловой системе, адреса электронной почты, названия сайтов ([www.mysite.com](http://www.mysite.com)) и некоторые другие (приложение А).

*Эвристика 7 (удаление длинных абзацев):* абзацы естественной речи для ИТ-дискуссий редко превышают 800 символов, при этом длина машинно сгенерированного текста (логи, трассировки, код) часто больше этого значения.

Данная эвристика является особенно важной, поскольку именно большие объемы терминологически бедного машинно сгенерированного текста оказывают наиболее заметное негативное влияние на тематическую модель.

*Эвристика 8 (абзацы с пунктуацией):* необходима для определения коротких (менее 800 символов) фрагментов машинно сгенерированного текста. Было установлено, что абзацы длиной больше 200 символов и содержащие более 6% символов пунктуации также являются машинно сгенерированными. В качестве символов пунктуации использовался следующий набор символов:

'- = / \* + , ; : ( ) { } [ ] < > % \$ @ & \_

Ограничение на минимальную длину абзаца позволяет избежать ложных срабатываний. Это также позволяет сохранить, например, название ошибки или полное имя класса.

*Эвристика 9 (удаление стоп-слов)*: предназначена для удаления наиболее частых слов английского языка, которые не помогают в определении темы в виду своего общего назначения.

Данная задача решается средствами библиотеки MALLET [17]. Библиотека MALLET используется для решения задач тематического моделирования (секция 3.3.1) и предоставляет набор предопределенных функций для модификации данных перед построением тематической модели. Одна из таких функций предназначена для удаления наиболее распространенных английских слов (стоп-слов).

К ним стоит добавить слова, часто используемые в анализируемой области ('java' или 'class' для обсуждения разработки на Java). В рамках выбранной предметной области такие слова также являются стоп-словами и затрудняют определение тем.

### 3.2.3. Фильтрация обращений

После применения эвристик некоторые из комментариев могут оказаться пустыми, в то время как другие могут не содержать ответа от технического специалиста. Из таких обращений не удастся извлечь ВОП. Воспользуемся метаинформацией об авторстве и отфильтруем обращения, имеющие не пустой первый комментарий (в версии для LDA) и не менее одного не пустого комментария от сотрудника технической поддержки.

Обращения, содержащие длинную нить обсуждения (более 6 комментариев), вероятно, имеют одну из следующих проблем: вопрос плохо сформулирован, вопрос слишком специфичен, ответ недостаточно полон и содержится в нескольких комментариях. Поскольку мы считаем, что ответ всегда содержится в не более чем одном комментарии (секция 2.2), корректную ВОП в данном случае составить не удастся. Такие обращения не имеют ценности с точки зрения ЧЗВ, их необходимо удалить.

Специфичные для предметной области обращения, например: обращение закрытое по причине слияния с другим обращением, и так далее — также подлежат удалению.



### 3.3. Тематическое моделирование

Тематическое моделирование [12] позволяет:

- Сгруппировать схожие обращения по темам (например, одна тема может касаться почтовой интеграции, а другая — вопросов о продлении подписки для пользователей);
- Охарактеризовать каждую тему списком терминов — мешком слов (bag-of-words).

#### 3.3.1. Скрытое размещение Дирихле

В работе используется метод скрытого размещения Дирихле и его реализация на Java [17]. LDA работает с любыми текстовыми документами, поэтому далее будет использоваться термин 'документ' для описания обращения, как совокупности его комментариев.

LDA — это вероятностная тематическая модель, не требующая размеченных данных для обучения, однако требующая указания количества моделируемых тем. LDA описывает каждую тему  $t$ , как вероятностное распределение по всем словам из входных данных ( $\phi_t$ ). Каждый документ  $d$  описывается вероятностным распределением по темам ( $\theta_d$ ). Более подробно LDA описывается в секции 1.3.4.

Распределения  $\phi$  и  $\theta$  в свою очередь зависят от гиперпараметров  $\alpha$  и  $\beta$  соответственно. Реализация LDA в [17] поддерживает автоматическую оптимизацию этих параметров с использованием сэмплирования по Гиббсу [16]. Программисту остается лишь указать количество тем.

Определение количества тем — нетривиальная задача, которая не рассматривается в оригинальной статье. В данной работе для этой цели предлагается использовать *меру неупорядоченности* (perplexity, перплексия) [1]. Эта метрика показывает сходство между терминами документов и их темой (меньше — лучше), но не отражает семантическую связь. Другими словами, при оценке тематической модели учитывается частота появления слов в документах, но при этом нет возможности учесть насколько некоторая пара слов близка по смыслу между собой. Поэтому так важен этап экспертной оценки.

Тем не менее перплексия позволяет определить минимальное число тем, при котором обращения начинают разделяться на четко выраженные подтемы. Критерием является значительное замедление ско-

рости падения перплексии с ростом числа тем. В секции 3.4.3 показывается как можно избавиться от расфокусированных тем, поэтому точное определение количества тем не требуется. Основываясь на перплексии, для построения тематической модели было выбрано количество тем, равное 200 (рисунок 3.3). Существуют и другие подходы к определению оптимального количества тем. Например, в статье [10] для данной задачи используется подход, основанный на генетических алгоритмах.

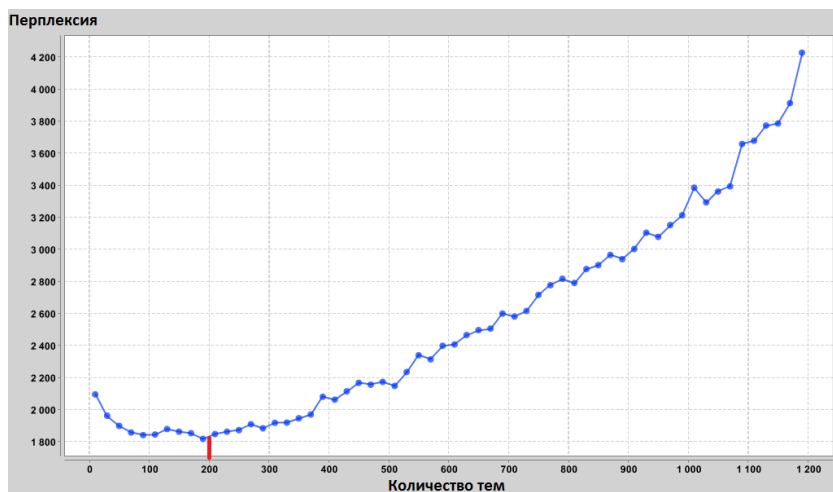


Рисунок 3.3. Зависимость перплексии от числа тем

В результате тематического моделирования для каждого документа определяется вероятностное распределение по темам  $\theta_{d,t}$ . Мы сопоставляем каждому обращению одну тему — тему с максимальной вероятностью. Однако, если для обращения  $d$  вероятность каждой темы  $\theta_{d,t} < 0.25$ , то такое обращение не имеет четко выраженной темы и для него не будет определяться ВОП.

Мешок слов каждой темы определяется, как совокупность слов всех документов, принадлежащих данной теме.

Способ построения тематической модели аналогичен предлагаемому в [9]. Дополнительно применяется перплексия для приблизительного определения количества тем.

### 3.4. Формирование пар вопрос-ответ

Процесс получения ЧЗВ из смоделированных тем состоит из трех шагов: фильтрация не информативных тем, определение пар вопрос-ответ, удаление расфокусированных тем. Первый шаг является новым, в то время как другие два — взяты из опорной статьи [9] за исключением пороговых значений.

#### 3.4.1. Дополнительная фильтрация

Данный этап не обязателен и предполагает некоторые априорные знания о данных, а также то, что алгоритм уже запускался ранее. Используемые в работе данные (обращения в техническую поддержку) могут содержать большое количество типичных, повторяющихся обращений с шаблонными ответами (просьбы о сбросе пароля; вопросы о недоступности сервиса и так далее). Такие обращения являются частыми, но не несут полезной информации для ЧЗВ.

После построения тематической модели можно найти мешки слов для таких тем. Необходимо запомнить 10 частых значимых слов для каждой из них. При последующих запусках LDA удаляются темы, для которых выполняется условие: хотя бы половина из 10 наиболее часто встречающихся слов темы совпадают с одной из фильтруемых тем. Темы проверяются на частичное совпадение, поскольку LDA недетерминирован и мешки слов могут незначительно отличаться от запуска к запуску.

Данный фильтр позволяет избавиться от шаблонных ВОП, но может негативно влиять на метрики качества (раздел 5), поскольку удаляемые таким образом темы четко выражены и содержат большое количество обращений.

#### 3.4.2. Определение вопросов и ответов

Для определения вопросов и ответов для каждого комментария вычисляется метрика близости с соответствующей темой. Для этого использовалось косинусное расстояние [18]:

$$\cos(e, t) = \frac{\sum_{i=1}^n t_i e_i}{\sum_{i=1}^n (t_i)^2 \sum_{i=1}^n (e_i)^2} \quad (3.1)$$

где  $t$  — вектор, соответствующий мешку слов темы,  $e$  — вектор, соответствующий словам комментария в представлении для LDA. Чем больше значение косинуса, тем сильнее комментарий связан с темой.

Пример получения векторов для текстов приведен в таблице 3.3. Пусть имеется два текста —  $t$  и  $e$ :

- $t$ : some short text text
- $e$ : some another text

Таблица 3.3. Пример построения векторов

	some	short	another	text
t	1	1	0	2
e	1	0	1	1

Для каждого из слов (для обоих текстов) считается сколько раз оно появилось в каждом из текстов. На основе этих значений строятся соответствующие для данных текстов векторы.

Комментарий выбирается в качестве *вопроса* при выполнении трех условий: (а) это первый комментарий в обращении; (б) косинус комментария и темы ( $\cos(Q, T)$ ) выше порога 0.15; (в) длина комментария не превышает 1000 символов (более длинный текст комментария говорит о слишком специфичном для конкретного пользователя вопросе).

Условия выбора комментария в качестве *ответа*: (а) это не первый комментарий в обращении; (б) не является комментарием инициатора обращения; (в) косинусное расстояние с темой ( $\cos(A, T)$ ) выше порога 0.15 и максимально среди других кандидатов на ответ.

Обращения в службу поддержки, типично представляют собой диалог с итеративным уточнением деталей, предоставлением дополнительной информации и попытками дать окончательный ответ. В качестве вопроса выбирается только иницирующий комментарий, поскольку все последующие комментарии пользователя не будут содержать полной информации о проблеме. Ответом считается комментарий сотрудника технической поддержки, наиболее совпадающий с темой по используемым терминам. Таким образом, обеспечивается сходство терминологии между вопросом и ответом для найденных ВОП.

### 3.4.3. Удаление расфокусированных тем

Ввиду отсутствия возможности точно определить моделируемое число тем (секция 3.3) возможны случаи, когда реальное количество тем будет меньше или больше смоделированного. В первом случае полученные после LDA темы будут слишком общими, что приведет к большим различиям между терминологией темы и принадлежащими ей обращениями и, как следствие, пониженному количеству и качеству найденных ВОП. Во втором — создадутся фантомные темы, терминология которых будет плохо совпадать с реальными данными. Удалим такие расфокусированные темы за счет введения минимальной доли ВОП (3.2) со значением 0,1.

$$\frac{|QAPairs_t|}{|tickets_t|} > threshold \quad (3.2)$$

где  $|QAPairs_t|$  — количество ВОП, извлеченных из обращений, принадлежащих к теме  $t$ ;  $|tickets_t|$  — количество обращений, принадлежащих к теме  $t$ ,  $threshold$  — минимальная доля ВОП.

Образованные ВОП упорядочиваются (в рамках каждой темы или глобально) с использованием гармонического среднего между  $\cos(Q, T)$  и  $\cos(A, T)$ . Гармоническое среднее (3.3) для получения высокого значения требует, чтобы все составляющие были высоки, таким образом, гарантируется, что и вопрос, и ответ имеют высокое качество.

$$H = \frac{n}{\sum_{i=1}^n \frac{1}{x_i}} \quad (3.3)$$

После этого ВОП передаются эксперту для валидации и редактирования перед публикацией. Данный этап описывается в разделе 5.

## 3.5. Резюме

В данном разделе представлено описание метода извлечения вопросно-ответных пар. Описаны основные составляющие этапы, такие как:

- Предобработка данных;
- Кластеризация обращений по тематикам;
- Извлечение ВОП.

Рассмотрены различные типы эвристик предобработки данных; обоснована необходимость фильтрации данных. А также описаны способы определения количества тем и удаления расфокусированных тем, определены условия, которым должны удовлетворять найденные вопросно-ответные пары.

## 4. РЕАЛИЗАЦИЯ АЛГОРИТМА ИЗВЛЕЧЕНИЯ ВОПРОСНО-ОТВЕТНЫХ ПАР

Для оценки эффективности предложенного в разделе 3 решения, реализуем его в виде программной библиотеки (реализация алгоритма). Результаты, полученные с помощью данной реализации, будут использованы в разделе 5 для проведения экспертного анализа и при оценке эффективности решения.

В данном разделе рассматриваются некоторые аспекты реализации предложенного метода извлечения ВОП из обращений в службу поддержки, описана структура проекта, определены используемые технологии. Полный текст реализации алгоритма доступен на прилагаемом к данной работе CD-диске.

### 4.1. Используемые технологии

При реализации алгоритма использовались следующие технологии: Java, Kotlin, Gradle, Git, JUnit, MongoDB, Mallet.

В качестве языка программирования используется *Kotlin*. Kotlin — это активно развивающийся язык программирования общего назначения для JVM, к достоинствам которого относятся:

- Вывод типов;
- Статическая типизация;
- Мультипарадигменность;
- Nullable типы данных;
- Выразительный синтаксис;
- Совместимость с Java кодом.

Описанные выше достоинства языка позволяют решать поставленные перед программистом задачи более эффективно, чем при использовании Java. При этом, благодаря совместимости с Java, сохраняется возможность использования большого количества существующих Java-библиотек.

На момент написания данной работы Kotlin поддерживает Gradle, Maven и Ant для автоматической сборки проектов. *Gradle* — система

автоматической сборки, предоставляющая DSL на языке Groovy, и, по сравнению с другими решениями, использующими XML, позволяет писать более компактные сценарии сборки.

*Git* — система контроля версий, используемая командой YouTrack.

*JUnit* — библиотека для модульного тестирования программного обеспечения. Совместима с Kotlin.

Для хранения данных используется *MongoDB*. Для работы с обращениями в службу поддержки команда YouTrack использует Zendesk. Загружаемые из Zendesk данные имеют JSON формат. Поскольку MongoDB использует JSON-подобные документы и схему базы данных, было решено использовать данную СУБД, вместо реляционных баз данных.

*Mallet* [17] — библиотека на Java, которая содержит реализацию необходимой тематической модели (LDA). Данная реализация написана в 2009 году, является наиболее “взрослой” и развитой реализацией LDA. Подробнее о выборе реализации LDA написано в секции 4.7.1.

## 4.2. Структура проекта

Структура проекта представляет собой набор пакетов (*packages*). Общая структура пакетов представлена на рисунке 4.1.

Каждый из пакетов выполняет соответствующую ему задачу:

- `org.jetbrains.zkb.zddownload` — загрузка данных из Zendesk и сохранение их в базу данных;
- `org.jetbrains.zkb.model` — модель данных, классы `Comment` (комментарий), `QAPair` (пара вопрос-ответ) и так далее.
- `org.jetbrains.zkb.db` — данный пакет содержит классы отвечающие за взаимодействие с базой данных;
- `org.jetbrains.zkb.filter` — содержит фильтры различного рода для анализируемых данных;
- `org.jetbrains.zkb.heuristics` — эвристики предобработки;
- `org.jetbrains.zkb.lda` — построение тематической модели анализируемых данных;
- `org.jetbrains.zkb.qa` — формирование ВОП;

Далее каждый из пакетов рассматривается более подробно.



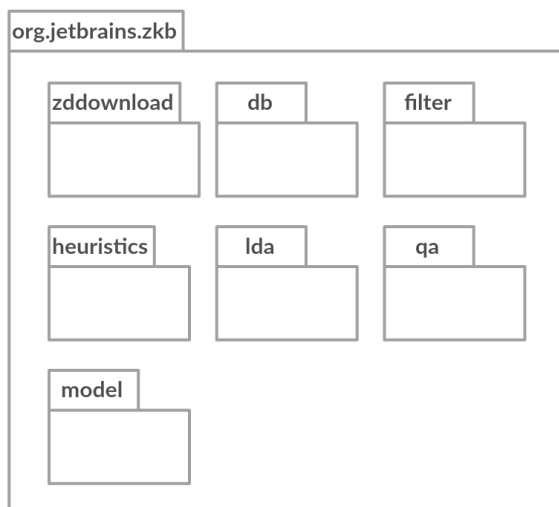


Рисунок 4.1. Обобщенная структура пакетов реализации алгоритма

### 4.3. Получение исходных данных

Пакет **org.jetbrains.zkb.zddownload** содержит программный код, отвечающий за загрузку данных из Zendesk.

Взаимодействие с Zendesk происходит через REST API [19]. В Zendesk обращение представлено такой сущностью, как “Ticket” (тикет). Тикет содержит ряд метаданных об обращении: уникальный идентификатор обращения, дата и время создания, дата и время последнего обновления, статус обращения, уникальный идентификатор инициатора и так далее — однако информация о соответствующих данному обращению комментариях отсутствует.

Комментарии представлены сущностью “Comment”, содержащей текст комментария и также некоторую дополнительную метаданную: уникальные идентификаторы комментария и автора, дата и время создания и так далее. Таким образом, для получения достаточной для дальнейшей обработки информации об одном обращении, необходимо загрузить:

1. Объект представляющий обращение (Ticket);

2. Все комментарии (Comment), принадлежащее данному обращению;

Для загрузки данных использовался клиент<sup>1</sup> для Zendesk API с открытым исходным кодом. Важное ограничение, которое нужно учитывать при взаимодействии с Zendesk — это ограничение на количество запросов в минуту. При интенсивном использовании API Zendesk может заблокировать все входящие запросы на некоторое время. Чтобы избежать данной ситуации, клиент<sup>1</sup> был модифицирован. Был добавлен учёт оставшегося количества запросов, а также интервал между запросами. Соответствующий фрагмент программы приведен ниже (листинг 4.1):

Листинг 4.1. Ограничение на использование Zendesk API

```
1 public class Zendesk implements Closeable {
2     private final AsyncHttpClient client;
3     private final Logger logger;
4
5     private Integer minRemainingApiCalls;
6     private Integer curRemainingApiCalls = null;
7     private Integer reqInterval;
8
9     // ...
10
11     // Данный метод отвечает за выполнение запроса,
12     // полученного в качестве первого параметра.
13     private <T> ListenableFuture<T> submit(Request request,
14         ZendeskAsyncCompletionHandler<T> handler) {
15         try {
16             // Ожидаем 1 минуту, если количество запросов слишком велико.
17             if (curRemainingApiCalls != null &&
18                 curRemainingApiCalls < minRemainingApiCalls) {
19                 Thread.sleep(60 * 1000);
20             } else {
21                 // Иначе ожидаем заданное время.
22                 if (reqInterval > 0) {
23                     Thread.sleep(reqInterval);
24                 }
25             }
26             } catch (InterruptedException e) {
27                 throw new ZendeskException(e.getMessage(), e);
28             }
29             return client.executeRequest(request, handler);
30     }
```

В листинге 4.2 приведен текст функции, отвечающей за загрузку данных из Zendesk. Частично загруженная информация сохраняется в

---

<sup>1</sup> <https://github.com/cloudbees/zendesk-java-client>

базу данных (функция *saveData()*) во избежание потери при разрыве соединения.

Листинг 4.2. Загрузка данных из Zendesk

```
1 private fun getData(  
2     zd: Zendesk,  
3     reader: DBReader,  
4     writer: DBWriter,  
5     startTimestamp_ms: Long,  
6     config: Config,  
7     skipSaving: Boolean  
8 ) {  
9     val startTime = Date(startTimestamp_ms)  
10    val ticketsLazyIterable = zd.getLazyTicketsSince(startTime)  
11    val lazyIterator = ticketsLazyIterable.iterator()  
12    var hasNext: Boolean = wrapZendeskAPICall(config.zdMaxRetry) {  
13        // Возможно обращение к API в:  
14        // org.zendesk.client.v2.Zendesk.PagedIterable.hasNext  
15        lazyIterator.hasNext()  
16    }  
17    while (hasNext) {  
18        val ticket = lazyIterator.next()  
19  
20        if (checkBrands(ticket, config.zdBrands) &&  
21            checkUpdatedAt(ticket, startTime)) {  
22            val comments: List<Comment> = wrapZendeskAPICall(config.  
23                zdMaxRetry) {  
24                // Обращение к API в:  
25                // org.zendesk.client.v2.Zendesk.PagedIterable.hasNext  
26                zd.getComments(ticket)  
27            }  
28            if (!skipSaving && comments.size > 1) {  
29                saveData(reader, writer, ticket, comments)  
30            }  
31        }  
32        hasNext = wrapZendeskAPICall(config.zdMaxRetry) { lazyIterator.  
33            hasNext() }  
34    }
```

Листинг 4.3. Объявление функции *wrapZendeskAPICall()*

```
1 private fun <T> wrapZendeskAPICall(maxRetry: Int, apiCall: () -> T): T
```

Все обращения к API происходят внутри функции *wrapZendeskAPICall()* (листинг 4.3), задача которой — корректная обработка исключительных ситуаций, возможных при обращении к Zendesk API.

## 4.4. Модель данных

Классы модели данных находятся в пакете **org.jetbrains.zkb.model**. Соответствующая диаграмма классов приведена на рисунке 4.2. Класс *Ticket* располагается вне данного пакета, вместо реализации данного класса используется модель, предоставляемая клиентом Zendesk (секция 4.3).

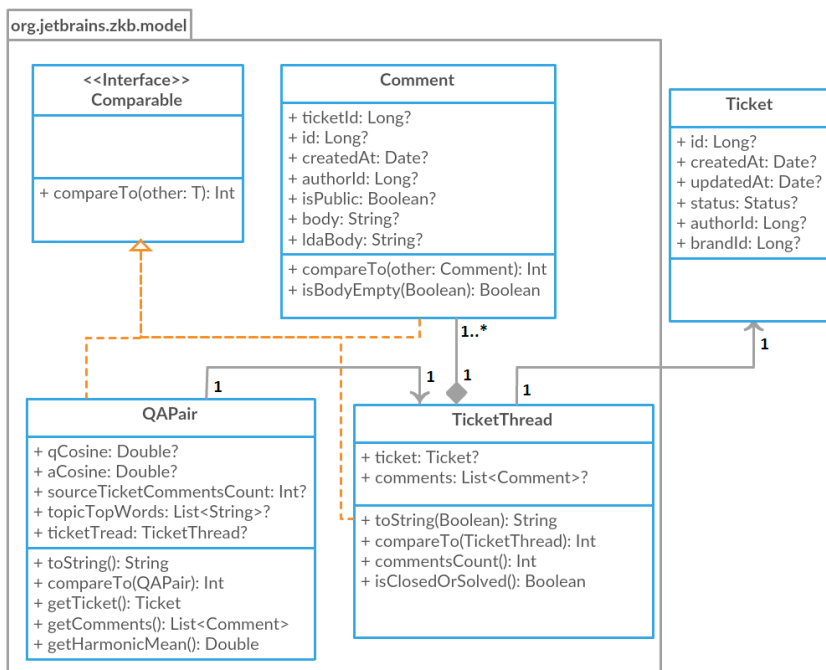


Рисунок 4.2. Диаграмма классов пакета org.jetbrains.zkb.model

Каждый из приведенных на рисунке 4.2 классов поддерживает сериализацию в формат JSON. Поддержка сериализации объектов необходима для организации взаимодействия с Zendesk и MongoDB. Для этого использовалась библиотека Jackson<sup>2</sup> с добавлением к классам специальных аннотаций. В листинге 4.4 приведен пример класса *Comment* с соответствующими аннотациями.

<sup>2</sup> <https://github.com/FasterXML/jackson>

#### Листинг 4.4. Пример использования библиотеки Jackson

```
1 @JsonIgnoreProperties(  
2     value = arrayOf("bodyEmpty")  
3 )  
4 class Comment() : Comparable<Comment> {  
5     @JsonProperty("ticket_id")  
6     var ticketId: Long? = null  
7     var id: Long? = null  
8     var body: String? = null  
9     @JsonProperty("lda_body")  
10    var ldaBody: String? = null  
11  
12    // ...  
13  
14    companion object {  
15        private val serialVersionUID = 1L  
16    }  
17 }
```

## 4.5. Взаимодействие с базой данных

Логика взаимодействия с базой данных инкапсулирована в классах *DBReader* и *DBWriter*, которые располагаются в пакете **org.jetbrains.zkb.db**. Диаграмма классов данного пакета приведена на рисунке 4.3.

MongoDB для хранения данных использует формат BSON. Класс *DBReader* реализует логику трансформации объектов в JSON формат, который затем преобразуется в BSON и сохраняется в базу данных. Класс *DBWriter* позволяет читать данные из базы данных, преобразуя их из BSON формата к объектам Kotlin.

## 4.6. Реализация предобработки данных

### 4.6.1. Фильтрация данных

За фильтрацию данных отвечает пакет **org.jetbrains.zkb.filter**. Фильтры представлены в виде набора функций, применяемых к коллекции фильтруемых объектов и возвращающих элементы, прошедшие фильтрацию. Эти фильтрующие функции поделены по файлам в соответствии с типом данных, к которым они применяются.

- *comments.kt* — удаление дубликатов; фильтрация комментариев для определенного набора обращений; фильтрация не пустых комментариев (после применения эвристик) и так далее;

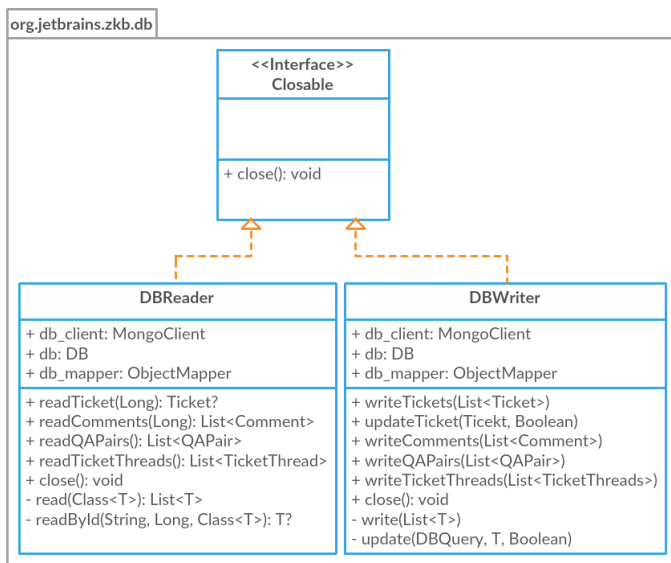


Рисунок 4.3. Диаграмма классов пакета org.jetbrains.zkb.db

- *tickets.kt* — фильтрация обращений с определенным статусом; удаление дубликатов; фильтрация обращений, относящихся к YouTrack;
- *ticketThreads.kt* — фильтр обращений с определенным количеством комментариев; фильтр обращений на английском языке и так далее;
- *filterData.kt* — применение описанных ранее функций фильтрации к необработанным данным.

Структура данного пакета показана на рисунке 4.4.

#### 4.6.2. Эвристики предобработки

Пакет **org.jetbrains.zkb.heuristics** содержит реализацию эвристик предобработки. Данный пакет реализован аналогично пакету, содержащему фильтрующие функции — эвристики предобработки представлены в виде функций, разделенных по файлам в соответствии с типом данных к которым они применяются.























TicketThreadsKt		FilterDataKt	
 <code>filterMoreThanOneComment(List&lt;TicketThread&gt;)</code>	List<TicketThread>	 <code>getFilteredTicketThreads(...)</code>	List<TicketThread>
 <code>filterEnglish(List&lt;TicketThread&gt;, boolean, boolean)</code>	List<TicketThread>	 <code>postHeuristicsFilters(...)</code>	List<TicketThread>
 <code>filterNotEmptyLdaBody(List&lt;TicketThread&gt;)</code>	List<TicketThread>	 <code>createPreparedTicketThreads(...)</code>	List<TicketThread>
 <code>filterStartCommentNotEmpty(List&lt;TicketThread&gt;)</code>	List<TicketThread>	GenericKt	
 <code>filterHasNotTopicStarterNotEmptyComment(List&lt;TicketThread&gt;)</code>	List<TicketThread>	 <code>filterDuplicatesByLd(List&lt;? extends T&gt;)</code>	List<T>
 <code>filterCommentsLessThan(List&lt;TicketThread&gt;, int)</code>	List<TicketThread>	 <code>filtrationPattern(List&lt;? extends T&gt;, String, Function1&lt;? s</code>	
 <code>filterFirstCommentWithoutAttachments(List&lt;TicketThread&gt;)</code>	List<TicketThread>		
TicketsKt		CommentsKt	
 <code>filterClosedOrSolved(List&lt;? extends Ticket&gt;)</code>	List<Ticket>	 <code>filterPublic(List&lt;Comment&gt;)</code>	List<Comment>
 <code>filterNotFollowupTickets(List&lt;? extends Ticket&gt;)</code>	List<Ticket>	 <code>filterCommentsForTickets(List&lt;Comment&gt;, List&lt;? extends Ticket&gt;)</code>	List<Comment>
 <code>filterDuplicatesByCreatedAt(List&lt;? extends Ticket&gt;)</code>	List<Ticket>	 <code>filterDuplicatesByLd(List&lt;Comment&gt;)</code>	List<Comment>
 <code>filterByBrandId(List&lt;? extends Ticket&gt;, List&lt;Long&gt;)</code>	List<Ticket>	 <code>filterDuplicatesByCreatedAt(List&lt;Comment&gt;)</code>	List<Comment>
 <code>filterDuplicatesByLd(List&lt;? extends Ticket&gt;)</code>	List<Ticket>	 <code>isBodyNotEmpty(List&lt;Comment&gt;, boolean)</code>	boolean

Рисунок 4.4. Структура пакета org.jetbrains.zkb.filter

- *comments.kt* — содержит реализацию эвристик, описанных в разделе 3;
- *ticketThreads.kt* — представляет API применения эвристик предобработки для коллекции *TicketThreads*. Содержит функцию *applyHeuristics()*, агрегирующую применение всех эвристик.

Реализация функции *applyHeuristics()* приведена в листинге 4.5.

Листинг 4.5. Применение эвристик предобработки данных

```

1  /**
2   * @param maxShortParagraphWords
3   * @param sentencesFrequency
4   * @param maxParagraphLength
5   * @param minParagraphLength
6   * @param punctuationProportion
7   * @param customActions a list of actions modifying comment body.
8   *                      It is applied before others look 'n' feel
9   *                      heuristics.
10  * @param customLdaActions a list of actions modifying comment ldaBody.
11  *                          It is applied before others LDA heuristics.
12  * @see applyRegex
13  * @see ldaApplyRegex
14  */
15  fun List<TicketThread>.applyHeuristics(
16      maxShortParagraphWords: Int,
17      sentencesFrequency: Int,
18      maxParagraphLength: Int,
19      minParagraphLength: Int,
20      punctuationProportion: Double,
21      customActions: List<CommentUpdater> = ytDefaultCustomActions,
22      customLdaActions: List<CommentUpdater> = listOf()

```

```

23 ): List<TicketThread> {
24     replaceCRLFtoLF()
25
26     // Эвристики отображения
27     applyCustomActions_(customActions)
28     removeEmailQuotes_()
29     removeShortStartParagraphs_(maxShortParagraphWords)
30     removeCommonSuffixes_(globally = true)
31     removeShortEndParagraphs_(maxShortParagraphWords)
32     removeFrequentSentences_(sentencesFrequency)
33     normalizeLookAndFeel_()
34
35     // Эвристики тематического моделирования
36     applyCustomLdaActions_(customLdaActions)
37     ldaRemoveURIs_()
38     ldaRemoveWordsWithApostrophes_()
39     ldaRemoveLongParagraphs_(maxParagraphLength)
40     ldaRemoveFilesystemPaths_()
41     ldaRemoveWWWAndEmails_()
42     ldaRemoveParagraphsWithPunctuation_(minParagraphLength,
43         punctuationProportion)
44
45     return this
46 }

```

## 4.7. Построение тематической модели

### 4.7.1. Выбор реализации LDA

В данной работе используется реализация LDA, содержащаяся в библиотеке MALLET [17]. MALLET — библиотека на Java, применяющаяся для анализа текста, классификации документов, тематического моделирования и так далее. Разработана Эндрю МакКаллум в университете города Амхерст (UMass Amherst) в 2002 году. Реализация LDA была добавлена в эту библиотеку в 2009 году. Данная реализация является наиболее “взрослой” и развитой реализацией LDA, в том числе используется авторами работы [9].

В 2015 году была добавлена реализация LDA в библиотеку scikit-learn [20], написанную на языке python. Использование данной реализации привело бы к необходимости организации дополнительного взаимодействия между Kotlin и Python, что усложнило бы процесс разработки.

Другие Java-реализации LDA:

- LDA in Mahout<sup>3</sup> — используется map-reduce подход, что приме-

---

<sup>3</sup> <https://mahout.apache.org/users/clustering/latent-dirichlet-allocation.html>



нимо для больших данных и неактуально для данной работы;

- LDA in Spark<sup>4</sup> — в документации указано, что данная реализация LDA является экспериментальной, в связи с чем нет уверенности в её стабильности;
- jLDADMM<sup>5</sup> — специфичная реализация для анализа коротких текстов.

#### 4.7.2. Пакет org.jetbrains.zkb.lda

Структура данного пакета приведена на рисунке 4.5.

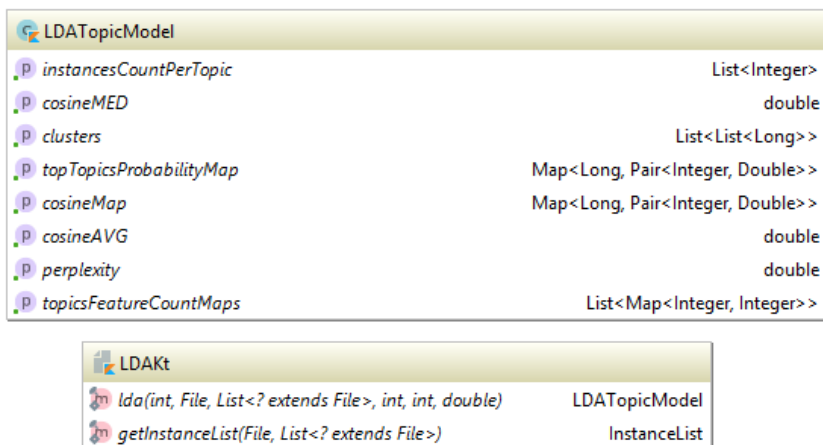


Рисунок 4.5. Структура пакета org.jetbrains.zkb.lda

Класс *LDATopicModel* наследуется от класса *ParallelTopicModel*, предоставляемого библиотекой *MALLET* и реализующего модель LDA, добавляет функциональность по вычислению перплексии полученной модели, построению кластеров обращений на основе темы с наибольшей вероятностью и некоторую другую. В листинге 4.6 представлен упрощенный код класса *LDATopicModel* без детальной реализации.

<sup>4</sup> <https://spark.apache.org/docs/latest/mllib-clustering.html>

<sup>5</sup> <http://jldadmm.sourceforge.net/>

#### Листинг 4.6. Класс LDATopicModel

```
1 import cc.mallet.topics.ParallelTopicModel
2 // ...
3
4 class LDATopicModel(
5     numberOfTopics: Int,
6     alphaSum: Double,
7     beta: Double
8 ) : ParallelTopicModel(numberOfTopics, alphaSum, beta), Serializable {
9
10     val instancesCountPerTopic: List<Int>
11     val clusters: List<List<Long>>
12     val topTopicsProbabilityMap: Map<Long, Pair<Int, Double>>
13     val topicsFeatureCountMaps: List<Map<Int, Int>>
14     val cosineMap: Map<Long, Pair<Int, Double>>
15     val perplexity: Double
16     val cosineMED: Double
17     val cosineAVG: Double
18
19     fun getTopicTopWords(
20         wordsCount: Int = 10
21     ): List<List<String>> = // ...
22
23     companion object {
24         @JvmStatic private val serialVersionUID: Long = 1L
25         @JvmStatic private val logger = LoggerFactory.getLogger(object
26             {}::class.java)
27
28         fun read(
29             modelFile: File,
30             stateFile: File? = null
31         ): LDATopicModel = // ...
32
33         override fun write(serializedModelFile: File) {
34             // ...
35         }
36
37         // ...
38     }
```

За создание и обучение тематической модели отвечает функция *lda()*, программный код которой приведен в листинге 4.7.

#### Листинг 4.7. Создание и обучение тематической модели

```
1 fun lda(
2     numTopics: Int,
3     sourceFile: File,
4     additionalStopWordsFiles: List<File> = listOf(),
5     threads: Int = 4,
6     iterations: Int = 2000,
7     alpha_t: Double = 0.1
8 ): LDATopicModel {
9     val instances = getInstanceList(sourceFile,
10         additionalStopWordsFiles)
```

```

10
11     val beta_w = 0.1
12     val model = LDATopicModel(numTopics, alpha_t * numTopics, beta_w)
13
14     model.addInstances(instances)
15     model.setNumThreads(threads)
16     model.setNumIterations(iterations)
17
18     logger.info(
19         "Estimating LDA topic model " +
20             "(topics: ${numTopics}, ins_cnt: ${instances.size},
21              " +
22              "iter: ${iterations}, threads: ${threads})..."
23     )
24     model.estimate()
25     logger.info("Done. ${model.instancesCountPerTopic.count { it > 0
26         }}/${numTopics} not empty topics.")
27
28     return model
29 }

```

## 4.8. Поиск вопросно-ответных пар













Программный код, решающий задачу поиска ВОП, расположен в пакете **org.jetbrains.zkb.qa**, структуру которого можно увидеть на рисунке 4.6.

Класс *QAFinder* инкапсулирует логику, описанную в секции 3.4. Его основными методами для публичного использования являются:

- *getQAPairsList()* — возвращает отсортированный в порядке уменьшения значения гармонического среднего список объектов *QAPair*;
- *getQAPairsByTopics()* — возвращает коллекцию списков объектов *QAPair*, где каждый список содержит ВОП, принадлежащие к одной теме.

Расчет косинусного расстояния вынесен за пределы класса *QAFinder*, поскольку помимо этого класса функция расчета косинуса используется в классе *LDATopicModel*.

Функция *getVectors()* вычисляет векторы, соответствующие текстам, между которыми необходимо посчитать расстояние (листинг 4.8). Результат данной функции затем передается в функцию *cosine()* для определения косинусного расстояния (листинг 4.9).

QAFinder	
 minAnswerLength	int
 commentsMap	Map<Long, List<Instance>>
 cosineAnswerThreshold	double
 cosineQuestionThreshold	double
 QAPairsPercentageThreshold	double
 topicProbabilityThreshold	double
 QAPairsByTopics	Map<Integer, List<QAPair>>
 model	LDATopicModel
 ticketThreads	List<TicketThread>
 maxQuestionLength	int
 QAPairsList	List<QAPair>
 highProbabilityTopicGroups	Map<Integer, List<TicketThread>>









CosineSimilarityKt	
 getVectors(Map<Integer, Integer>, Map<Integer, Integer>)	Pair<List<Integer>, List<Integer>>
 vectorLength(List<Integer>)	double
 cosine(List<Integer>, List<Integer>)	double
 checkIsAllPositive(List<Integer>, List<Integer>)	void
 checkNoZeroPairs(List<Integer>, List<Integer>)	void
 checkSizes(List<Integer>, List<Integer>)	void
 cosine(Map<Integer, Integer>, Map<Integer, Integer>)	double
 topicCommentCosine(LDATopicModel, int, Instance)	double

Рисунок 4.6. Структура пакета org.jetbrains.zkb.qa

Листинг 4.8. Построение векторов

```

1 internal fun getVectors(
2     wordCountMapText: Map<Int, Int>,
3     wordCountMapTopic: Map<Int, Int>
4 ): Pair<List<Int>, List<Int>> {
5     checkIsAllPositive(
6         wordCountMapText.keys.toList(),
7         wordCountMapText.values.toList()
8     )
9     checkIsAllPositive(
10        wordCountMapTopic.keys.toList(),
11        wordCountMapTopic.values.toList()
12    )
13 }

```

```

14     val mixed: MutableMap<Int, Pair<Int, Int>> = mutableMapOf()
15
16     wordCountMapTopic.forEach {
17         mixed[it.key] = Pair(0, it.value)
18     }
19     wordCountMapText.forEach {
20         if (it.key in mixed) {
21             val pair = mixed[it.key]!!
22             mixed[it.key] = pair.copy(first = it.value)
23         } else {
24             mixed[it.key] = Pair(it.value, 0)
25         }
26     }
27     return mixed.map { it.value }
28         .unzip()
29 }

```

Листинг 4.9. Вычисление косинусного расстояния

```

1 fun cosine(textVector: List<Int>, topicVector: List<Int>): Double {
2     checkSizes(textVector, topicVector)
3     checkIsAllPositive(textVector, topicVector)
4     checkNoZeroPairs(textVector, topicVector)
5
6     val nominator = textVector.zip(topicVector)
7         .sumBy { it.first * it.second }
8     val denominator = vectorLength(textVector) * vectorLength(
9         topicVector)
10    return nominator / denominator
}

```

## 4.9. Резюме

В данном разделе рассмотрены основные аспекты реализации предложенного метода извлечения ВОП из обращений в службу поддержки. Описаны используемые технологии, структура проекта. Приведены листинги некоторых функций и классов, UML-диаграммы классов соответствующих пакетов.



## 5. ОЦЕНКА ЭФФЕКТИВНОСТИ РАЗРАБОТАННОГО ПОДХОДА ИЗВЛЕЧЕНИЯ ВОПРОСНО-ОТВЕТНЫХ ПАР

В данном разделе проводится оценка эффективности разработанной технологии извлечения ВОП и оценивается качество самих ВОП. Эффективность оценивается путём изучения влияния эвристик предобработки и параметров на выбранные метрики качества, а качество ВОП — при помощи экспертной оценки.

### 5.1. Определение доли найденных ВОП

На рисунке 5.1 изображена помесечная гистограмма результатов, показывающая соотношение количества исходных обращений, предобработанных и отфильтрованных обращений, а также найденных ВОП.

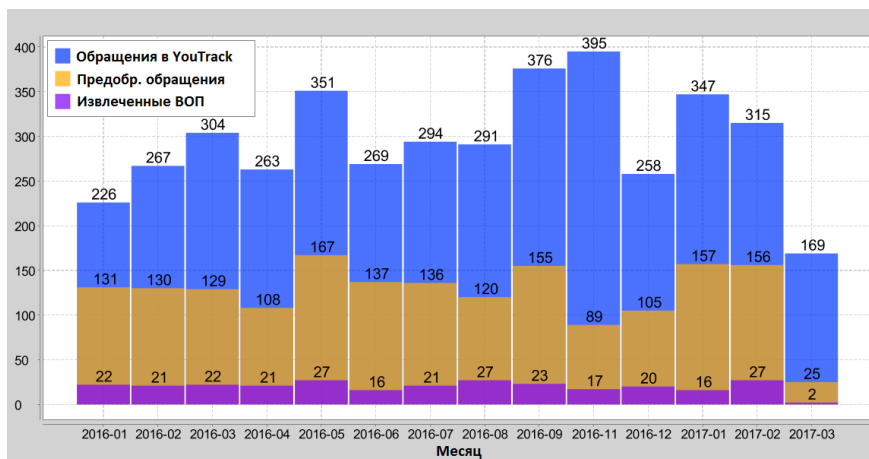


Рисунок 5.1. Гистограмма извлеченных ВОП

По графику видно, что медианное значение количества вопросов, на которое может быть расширен раздел с ЧЗВ, составляет 21 вопрос в месяц. В среднем доля ВОП составляет 6,8%.

Как упоминалось ранее, анализируемые данные не являются предварительно размеченными, в связи с чем отсутствует возможность посчитать полноту и точность решения. Воспользуемся другими способами оценки качества найденных ВОП, которые рассмотрены далее.

## 5.2. Оценка влияния эвристик и параметров на качество ВОП

Характеристики анализируемых данных включают:

- 6500 обращений за период с января 2016 года по март 2017;
- 19700 комментариев;
- 45000 различных слов;
- Тематическое моделирование проводилось для 200 тем.

В работе [9] установлено, что ВОП, признанные экспертами подходящими для публикации в ЧЗВ, имеют более высокие значения косинусного расстояния [9], секция VI–С). На основе этого для оценки качества разработанного способа извлечения ВОП воспользуемся следующими метриками: косинусное расстояние между вопросом и темой, косинусное расстояние между ответом и темой. Таблица 5.1 показывает влияние этапов предобработки и параметров алгоритма на значения метрик.

В первой строке показаны значения метрик для оптимальных параметров алгоритма. Последующие строки описывают, какое изменение было внесено и как оно влияет на выбранные метрики качества по сравнению с оптимальными параметрами. Качество извлеченных ВОП считается тем выше, чем выше косинусное расстояние для вопроса и ответа.

По таблице 5.1 видно, что применение эвристик предобработки и фильтров (за исключением фильтра тем) положительно влияет на косинус вопросов и ответов. На основании этого можно сделать вывод о том, что применение соответствующих этапов ведет к повышению качества извлеченных ВОП.

Помимо этого из таблицы 5.1 можно сделать вывод о том, что изменение параметров в большую сторону позволяет находить меньшее количество более качественных, с точки зрения косинусного расстояния, ВОП.



Таблица 5.1. Влияние эвристик и параметров на медианные значения метрик

Исследуемый параметр	Старое значение	Новое значение	Перплексия	Количество ВОП	$\cos(Q, T)$	$\cos(A, T)$
Оптимальные параметры	-	-	1864	357	0.396	0.413
Эвристики отображения (3.2.1)	вкл	выкл	1971	361	0.371	0.395
Эвристики тем. модел. (3.2.2)	вкл	выкл	2016	384	0.369	0.391
Фильтр обращений (3.2.3)	вкл	выкл	1924	403	0.370	0.388
Фильтр тем (3.4.1)	вкл	выкл	1791	472	0.393	0.416
Порог выбора темы LDA (3.3.1)	0.25	0.0	1853	376	0.366	0.404
Порог выбора темы LDA (3.3.1)	0.25	0.4	1871	302	0.399	0.421
Мин. значение косинуса (3.4.2)	0.15	0.0	1860	394	0.337	0.359
Мин. значение косинуса (3.4.2)	0.15	0.3	1864	288	0.387	0.419
Мин. доля ВОП (3.4.3)	0.1	0.0	1869	376	0.384	0.407
Мин. доля ВОП (3.4.3)	0.1	0.2	1850	324	0.391	0.417

### 5.3. Экспертная оценка

Экспертная оценка проводилась только для оптимального набора параметров. Из данных, описанных в предыдущей секции, было извлечено 358 ВОП. Некоторые из характеристик полученных вопросно-ответных пар приведены в таблице 5.2.

Таблица 5.2. Характеристики извлеченных ВОП

<b>Количество ВОП</b>	358
<b>Ср. длина вопроса (симв.)</b>	323
$avg \cos(Q, T)$	0.401
$avg \cos(A, T)$	0.412
<b>Гармоническое ср.</b>	0.406
<b>Ср. количество комментариев в исх. обращении</b>	3.6

При проведении экспертной оценки использовались индивидуальные оценки, основанные на мнениях независимых друг от друга экспертов. Способ оценки — ранжирование. Ранжируемое свойство — корректность вопросно-ответной пары. Под корректностью ВОП подразумевается:

- Пригодность вопроса для добавления в ЧЗВ;
- Соответствие ответа задаваемому вопросу;
- “Внешний вид” ВОП — наличие шумов (секция 3.1), необходимость в редактировании.

Экспертам было необходимо отнести каждую из оцениваемых ВОП к одной из следующих категорий:

1. Подходит для публикации без редактирования;
2. Подходит для публикации с редактированием вопроса и/или ответа;
3. Не подходит для публикации. Некорректный вопрос;
4. Не подходит для публикации. Некорректный ответ;

Корректными считаются ВОП, относящие к категории 1 или 2.

В качестве экспертов выступили команда технической поддержки и команда технических писателей YouTrack — 4 человека. Для отображения ВОП и получения оценок использовался веб-интерфейс. Экспертам предлагалось последовательно оценивать по одной ВОП в случайном порядке. Экспертам были доступны только текст вопроса и ответа, дополнительные характеристики ВОП, как, например, косинусное расстояние между вопросом и темой, были скрыты. Ограничения на максимальное количество ВОП, оцениваемых одним экспертом установлено не было. Результаты представлены в таблице 5.3.

Таблица 5.3. Результаты экспертной оценки

Категория ВОП	Количество	Доля, %
<b>Общее количество</b>	358	100
1. Подходит для публикации без редактирования.	262	73
2. Подходит для публикации с редактированием вопроса и/или ответа.	17	5
3. Не подходит для публикации. Некорректный вопрос.	17	5
4. Не подходит для публикации. Некорректный ответ.	62	17

Доля подходящих для публикации ВОП составила 78% (категории 1 и 2). В приложении Б представлены примеры ВОП, принадлежащих к каждой из категорий.

## 5.4. Резюме

В данном разделе проведена оценка качества разработанной технологии извлечения ВОП. Определяется численная метрика качества ВОП: косинусное расстояние между вопросом и темой, косинусное расстояние между ответом и темой.

Оценка влияния эвристик и параметров на качество ВОП показала, что используемые эвристики повышают качество извлекаемых ВОП. Повышение значений параметров алгоритма приводит к умень-

пению количества извлекаемых ВОП, при этом повышаются значения выбранных метрик качества.

Описана методика проведения экспертного оценивания. Вводится понятие корректных ВОП. По результатам экспертной оценки, проведенную работу можно считать успешной, поскольку:

- Достигнутый результат (78%) превосходит результаты работы [9] (37%), использующей похожее решение;
- Команда YouTrack удовлетворена достигнутыми результатами, планируется внедрение данного решения в рабочий процесс YouTrack.

## ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы была разработана технология извлечения вопросно-ответных пар из обращений в службу поддержки. Предложенный способ учитывает специфику ИТ-дискуссий и, помимо обращений в службу поддержки, может быть также применен к таким источникам как: вопросно-ответные системы, форумы.

Для решения данной задачи был предложен подход с использованием тематического моделирования. Для каждого обращения определяется тема, которая представляет собой мешок слов (секция 3.1). Вопросно-ответные пары затем определяются на основе косинусного расстояния между отдельно взятым комментарием и темой обращения. Данный подход основан на сходстве используемой терминологии между вопросом и ответом на него.

В работе были рассмотрены существующие на сегодняшний день тематические модели (раздел 1). Проведено их сравнение, на основе которого выбрана модель LDA, как наиболее эффективная. Важными особенностями данной модели является то, что она не требует размеченных входных данных для обучения и позволяет определять тему для нового документа без необходимости повторного обучения.

В разделе 2 описаны анализируемые данные и их источник. Формируются решаемые задачи:

1. Предобработка данных;
2. Кластеризация обращений по темам;
3. Извлечение ВОП;
4. Оценка эффективности разработанного подхода.

Первые три шага подробно описаны разделе 3. Предобработка данных позволяет избавиться от обращений, заведомо не содержащих ВОП. Из оставшихся обращений удаляются шумы, которые негативно влияют на качество тематической модели и затрудняют определение вопроса или ответа. Применяются специфичные для ИТ-дискуссий эвристики, связанные с удалением машинно-сгенерированного текста.

LDA используется для кластеризации обращений по темам. Каждому обращению присваивается тема с наибольшей вероятностью, после чего происходит определение ВОП.

В разделе 4 приведена реализация соответствующего предложенному решению алгоритма. Эта реализация используется в разделе 5 для определения качества извлекаемых ВОП и эффективности решения.

Оценка влияния эвристик и параметров на качество извлекаемых ВОП показала, что используемые эвристики влияют на их качество положительно. Повышение значений параметров алгоритма приводит к уменьшению результирующего количества ВОП, при этом повышаются значения выбранных метрик качества.

По результатам экспертной оценки, 78% из найденных ВОП признано корректными, команда YouTrack заинтересована во внедрении данного решения в рабочий процесс.

К достоинствам описанного в данной работе подхода можно отнести то, что помимо часто задаваемых вопросов могут быть также найдены и редкие ВОП, если они хорошо сформулированы и имеют корректный ответ.

К ограничениям можно отнести: анализируются только обращения на английском языке; в качестве ответа выбирается только один комментарий.

Если ответ на вопрос (или сам вопрос) содержится в более чем одном комментарии, то, вероятно, вопрос плохо сформулирован или ответ недостаточно полон. Однако комбинирование нескольких комментариев для составления вопроса или ответа является одной из возможных тем для дальнейших исследований. К другим направлениям для дальнейших исследований относятся:

- Увеличение полноты решения;
- Улучшение качества определения машинно-сгенерированного текста;
- Обработка мультязычных данных:
  - Моноязычные обращения;
  - Мультязычные обращения.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Blei David M., Ng Andrew, Jordan Michael. Latent Dirichlet allocation // Journal of Machine Learning Research. — 2003. — Vol. 3.
2. Shrestha Lokesh, McKeown Kathleen. Detection of Question-answer Pairs in Email Conversations // Proceedings of the 20th International Conference on Computational Linguistics. — COLING '04. — Stroudsburg, PA, USA : Association for Computational Linguistics, 2004. — URL: <https://doi.org/10.3115/1220355.1220483>.
3. Finding Question-answer Pairs from Online Forums / Gao Cong, Long Wang, Chin-Yew Lin et al. // Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. — SIGIR '08. — New York, NY, USA : ACM, 2008. — P. 467–474. — URL: <http://doi.acm.org/10.1145/1390334.1390415>.
4. Hong Liangjie, Davison Brian D. A Classification-based Approach to Question Answering in Discussion Boards // Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval. — SIGIR '09. — New York, NY, USA : ACM, 2009. — P. 171–178. — URL: <http://doi.acm.org/10.1145/1571941.1571973>.
5. Gottipati S, Lo D, Jiang J. Finding relevant answers in software forums // Proceedings of the Automated Software Engineering Conference. — 2011.
6. Tian Qiongjie, Zhang Peng, Li Baoxin. Towards Predicting the Best Answers in Community-Based Question-Answering Services // Proceedings of the International AAAI Conference on Web and Social Media. — 2013.
7. Celikyilmaz A, Hakkani-Tur D, Tur G. Lda based similarity modeling for question answering // Proceedings of the NAACL HLT 2010 Workshop on Semantic Search. — 2010.
8. Liu M, Liu Y, Yang Q. Predicting best answerers for new questions in community question answering // Proceedings of the 11th International Conference on Webpage Information Management. — 2010.
9. Henß Stefan, Monperrus Martin, Mezini Mira. Semi-automatically Extracting FAQs to Improve Accessibility of Software Development

- Knowledge // Proceedings of the International Conference on Software Engineering (ICSE). — 2012.
10. How to Effectively Use Topic Models for Software Engineering Tasks? An Approach Based on Genetic Algorithms / Annibale Panichella, Bogdan Dit, Rocco Oliveto et al. // Proceedings of the 2013 International Conference on Software Engineering. — 2013. — P. 522–531.
  11. Source-LDA: Enhancing probabilistic topic models using prior knowledge sources / J. Wood, P. Tan, A. Das et al. // ArXiv e-prints. — 2016. — Jun. — 1606.00577.
  12. Коршунов Антон, Гомзин Андрей. Тематическое моделирование текстов на естественном языке // Труды Института Системного Программирования РАН. — 2012. — Т. 23.
  13. Воронцов Константин. Вероятностное тематическое моделирование. — 2013.
  14. Topic Detection and Tracking Pilot Study Final Report / James Allan, Jaime Carbonell, George Doddington et al. // In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop. — 1998. — P. 194–218.
  15. Choi Seung-Seok, Cha Sung-Hyuk, Tappert Charles C. A survey of binary similarity and distance measures // Journal of Systemics, Cybernetics and Informatics. — 2010. — Vol. 8, no. 1. — P. 43–48.
  16. Heinrich Gregor. Parameter estimation for text analysis // Technical report, Fraunhofer IGD. — 2005.
  17. McCallum Andrew Kachites. MALLET: A Machine Learning for Language Toolkit. — 2002. — <http://mallet.cs.umass.edu>.
  18. Goma Wael H., Fahmy Aly A. A Survey of Text Similarity Approaches // International Journal of Computer Applications. — 2013. — Vol. 68.
  19. Inc. Zendesk. REST API Documentation. — 2017. — URL: [https://developer.zendesk.com/rest\\_api](https://developer.zendesk.com/rest_api) (online; accessed: 2017-05-22).
  20. Scikit-learn. Latent Dirichlet Allocation Documentation. — 2017. — URL: <http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html> (online; accessed: 2017-05-28).



# ПРИЛОЖЕНИЕ А

## РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

Листинг А.1. Регулярные выражения для эвристик отображения

```
1 @Language("Regexp")
2 private const val MERGED_TICKET_REGEX =
3     """"This request was closed and merged into request #\d+""""
4 @Language("Regexp")
5 private const val YT_INCLOUD_SUBSCRIPTION_CANCEL_REGEX =
6     """"I would like to cancel my YouTrack InCloud subscription""""
7 @Language("Regexp")
8 private const val REQUEST_FOR_DETAILS_REGEX =
9     """"could you please ((provide)|(attach)|(send))""""
10 @Language("Regexp")
11 internal const val ORIGINAL_TICKET_ID =
12     """"\b(original_ticket_id_)\d*""""
13 @Language("Regexp")
14 internal const val TEMPLATE_TICKET_PRODUCT =
15     """"~((\{*2\}?Product(\{*2\}[p{C}\p{Z}\s])?: ).*$""""
16 @Language("Regexp")
17 internal const val TEMPLATE_TICKET_QUESTION =
18     """"~(Question: ).*$""""
19 @Language("Regexp")
20 internal const val TEMPLATE_TICKET_QUESTION_MARKUP =
21     """"~(\{*2\}Question\{*2\}[p{C}\p{Z}\s]: ).*$""""
22 @Language("Regexp")
23 internal const val TEMPLATE_TICKET_MERGED =
24     """"Request #\d+ ".*" was closed and merged into this request. Last
        comment in request #\d+:""""
```

Листинг А.2. Регулярные выражения для эвристик тематического моделирования

```
1 @Language("Regexp")
2 internal const val ATTACHMENT_REGEX =
3     """"!\[.*\]\([~\s]*\)""""
4 @Language("Regexp")
5 internal const val WWW_REGEX =
6     """"(\b)www\.\S+\.\S+""""
7 @Language("Regexp")
8 internal const val EMAIL_REGEX = """"\S+@\S+""""
9 @Language("Regexp")
10 internal const val APOSTROPHE_REGEX =
11     """"\b\w{1,5}[''\u00B4\u2018\u2019]\w+\b""""
12 @Language("Regexp")
13 internal const val PATH_REGEX =
14     """"[~\p{C}\p{Z}\s]*[/\\](\((program )|(Program Files ))?[~\p{C}\p{Z}\s]
        s+[/\\][~\p{C}\p{Z}\s]*""""
15 @Language("Regexp")
16 internal const val URI_REGEX =
17     """"[~\p{C}\p{Z}\s]+://[~\p{C}\p{Z}\s]*""""
```



## ПРИЛОЖЕНИЕ Б

### ПРИМЕРЫ ВОПРОСНО-ОТВЕТНЫХ ПАР

Таблица Б.1. Пример ВОП, подходящей для публикации без редактирования

<b>Вопрос:</b>
My company is currently experiencing an issue where a sprint created in one agile board will show up in multiple boards. It appears that in new boards, selecting "create independent copy" in the sprint field will fix this for the future, but I would like to fix this in our existing boards. Please let me know what steps we can take!
<b>Ответ:</b>
If you have multiple boards connected to the same custom field with the same values bundle, then this behaviour is as expected. Creating independent copies of values bundles should help avoid such synchronization. In existing boards you can fix this only by deleting unnecessary sprints manually.

Таблица Б.2. Пример ВОП, подходящей для публикации с редактированием вопроса

<b>Вопрос:</b>
It is so basic an issue yet I do not find any help for this. Could you send me a pointer to this. I have already projects and issues based on the project. Now I start a scrum and want to add issues to this scrum which are already existing. How do I do this? ....
<b>Вопрос после ручного редактирования:</b>
Could you send me a pointer to this. I already have projects and issues based on the project. Now I start a sprint and want to add issues to this sprint which are already existing. How do I do this?
<b>Ответ:</b>
You can add issues from the issue full screen "Board" section on the bottom right under custom field. Also it's possible to add them from the backlog on the board or with the query: 'Board <boards name> <sprint name>'.

*Красным цветом выделен текст, подлежащий редактированию.*

Таблица Б.3. Пример ВОП с некорректным вопросом

<b>Вопрос:</b>
I just installed YouTrack on a RedHat 7 server and I'm attempting to run it from the command line with the following command: "java -Xmx1g <some long command>" The console shows the following: "OpenJDK 64-Bit Server VM warning: <some long output>" I tried the default suggestion (see my call to launch YouTrack) but it isn't resolving this issue. Can someone help me to resolve this please?
<b>Ответ:</b>
I believe it to be a parameter ordering issue. Please try the following command and let me know if it helps: java <some long command>

Таблица Б.4. Пример ВОП с некорректным ответом

<b>Вопрос:</b>
I am setting up a site (asp.net) for my company. Let's call that www.mycompany.com and it will be hosted by IIS. I just installed Youtrack and it's running at localhost:8082 Everything is working great. So I am wondering if you have any documentation in order to setup Youtrack behind an IIS proxy? i.e. www.mycompany.com/youtrack
<b>Ответ:</b>
Sorry for a short delay! "Default Web Site" is an entry point on your IIS configuration. As far as I investigated your case, it could be done, please refer to <a href="http://&lt;longurl&gt;">http://&lt;longurl&gt;</a> . Also, this guide includes several other helpful instructions, check the table of contents. Thank you and let me know if have any additional questions, I'll try to help you with configuring IIS.