САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра системного программирования

Григорьев Артём Валерьевич

Оценка распределения вероятностей поисковых целей для многозначных запросов

Дипломная работа

Допущена к защите.
Зав. кафедрой:
проф., д. ф.-м. н. А.Н. Терехов

Научный руководитель:
ст. преп., к. ф.-м. н. Л.В. Грауэр

Рецензент:
доц., к. ф.-м. н. Д.С. Шалымов

Санкт-Петербург
2013

SAINT-PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Software Engineering Chair

Artem Grigorev

Estimating probability distribution of ambiguous queries search intents

Graduation Thesis

Admitted for defence.
Head of the chair:
Professor A.N. Terekhov

Scientific supervisor:
PhD L.V. Grauer

Reviewer:
PhD D.S. Shalymov

Saint-Petersburg
2013

Оглавление

Введение

Ввиду растущих объёмов информации и развития телекоммуникационных технологий поиск в сети интернет является одним из ведущих направлений научной деятельности последних лет. Аудитория поисковых систем неуклонно растёт (см. рис. 1), по данным компании Alexa[1] на май 2013 года сайт поиска «Яндекс»[2] занимает первое место по посещаемости среди всех сайтов в России, а «Google»[3] второе место в аналогичном рейтинге в мире.

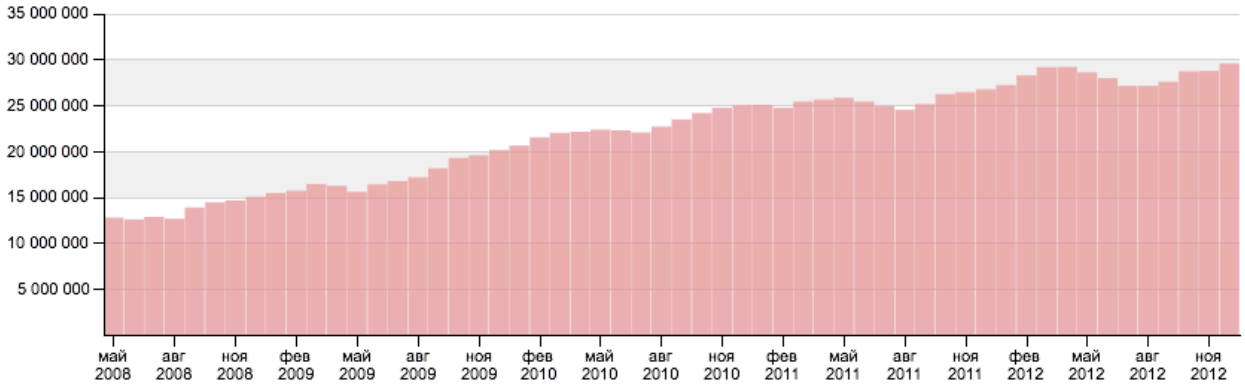


Рис. 1: Количество человек, воспользовавшихся сервисом «Поиск Яндекса», хотя бы 1 раз за месяц[5]

Основная цель поисковых систем — отвечать на запросы пользователей релевантными документами. Одним из подходов к измерению качества поиска является расчёт различных метрик[5], обычно основанных на экспертных оценках специально обученных людей (асессоров). Этот процесс может быть построен следующим образом: формируется некоторая случайная выборка из общего потока поисковых запросов (корзина), по каждому из этих запросов выкачивается выдача[6] оцениваемой поисковой системы (набор документов), релевантность каждого документа по отношению к исходному запросу оценивается асессорами,

[1] http://alexa.com

[2] http://yandex.ru

[3] http://google.com

[4] http://clck.ru/8dsPJ

[5] В этом разделе под метрикой понимается функция, сопоставляющая запросу и списку документов вещественное положительное число.

[6] В англоязычной литературе обычно используется понятие SERP (search engine response page).

после чего по каждой выдаче с использованием полученных оценок вычисляется значение метрики, усреднив эти значения для всех запросов из корзины, можно получить некоторую оценку качества поиска.

Примером метрики является *pfound* [27], оценивающая вероятность нахождения релевантного результата. Данная метрика используется в качестве основной метрики на конференции РОМИП. Pfound вычисляется по формуле:

$$pfound = \sum_{i=1}^{n} pLook_i \cdot pRel_i,$$

где $pRel_i$ — оценка вероятности того, что i-й документ окажется релевантным, может быть вычислена на основе оценок асессоров, а $pLook_i$ — оценка вероятности просмотра пользователем i-го документа, вычисляется по формуле:

$$pLook_i = pLook_{i-1} \cdot (1 - pRel_{i-1}) \cdot (1 - pBreak),$$

где $pBreak$ — вероятность того, что пользователь остановится в просмотре поисковой выдачи.

Как уже было упомянуто выше, многие метрики, в том числе и pfound, рассчитываются на основе экспертных оценок, выставленных по некоторой шкале (оценочные метрики). Перед асессором стоит задача оценить, насколько данный документ соответствует данному поисковому запросу. Для этого асессор должен как можно более точно понимать то значение запроса, которое в него вкладывал пользователь.

Среди всех запросов можно выделить класс *многозначных*, сюда относятся запросы с потенциально различными поисковыми целями (интентами) пользователя. Запрос может иметь несколько возможных значений, к примеру, слово *ягуар* может означать животное, марку автомобиля или напиток. Пользователь, задавший запрос *17 мгновений весны*, может хотеть посмотреть этот фильм онлайн, скачать его или прочитать какую-то информацию о нём — все эти действия примеры различных прагматик данного многозначного запроса. Бывают также запросы со смешанным набором значений и прагматик: например, по запросу *Штирлиц* пользователь может искать фильм, книгу, анекдоты

про этого героя, а также компьютерную программу с таким названием или информацию о соционическом типе личности. Согласно пресс-релизу [25], опубликованному компанией «Яндекс», многозначные запросы составляют около 20% от всего объёма запросов к этой поисковой системе.

Многозначные запросы требуют особого подхода при оценке качества поиска. В работе [5] для расчёта метрик предлагается использовать средневзвешенную сумму значений метрик по отдельным интентам:

$$metrics_{IA}(q, D) = \sum_{i \in I} w_i \cdot metrics(q, D, i),$$

где q — исходный запрос, D — список документов в поисковой выдаче, $metrics_{IA}$ — значение IA-метрики[7], $metrics(q, D, i)$ — значение метрики, посчитанное для интента i из набора всех интентов I (при этом для оценочных метрик каждый документ оценивается в контексте каждого интента), w_i — вес интента i, в качестве которого предлагается использовать оценку вероятности того, что пользователь, задавая данный многозначный запрос, имел данный интент, $\sum_i w_i = 1$.

Постановка задачи

Целью данной дипломной работы является построение алгоритма, рассчитывающего для многозначных запросов оценки весов интентов, которые впоследствии могут быть использованы в метриках оценки качества поиска. Для достижения цели были поставлены следующие задачи:

1. Разработать и реализовать требующийся алгоритм;
2. Создать систему для запуска и анализа расчётов;
3. Оценить качество полученных результатов.

[7]Intent Aware — учитывающая интенты

Структура работы

В главе 1 делается обзор предметной области и работ в ней: разделение потока действий пользователя на логические сессии, различные графовые модели, изучение стратегий переформулирования в применении к предложению поисковой системой подсказок-уточнений исходного запроса пользователя, подходы к кластеризации поисковых запросов; также в этой главе делается краткий обзор используемых технологий.

В главе 2 даётся описание созданного решения.

В главе 3 описываются шаги построенного алгоритма без технических подробностей их реализации.

В главе 4 рассказываются наиболее интересные аспекты реализации: использование в работе алгоритма технологии MapReduce, создание тестовых наборов многозначных запросов и ручная разметка для подбора параметров и проверки качества работы алгоритма, особенности реализации созданного веб-сервиса. Также описываются вспомогательные утилиты, созданные для подготовки необходимых данных: расчёт статистики переформулировок; построение наборов запросов, имеющих переходы на общие документы в выдаче; выборка поисковых сессий, содержащих данный запрос.

В главе 5 описываются эксперименты, проведённые для оценки качества построенного алгоритма.

В «Заключении» подводится итог полученным результатам, описываются возможные направления для дальнейшего развития.

1. Обзор литературы

1.1. Выделение логических сессий

Логи (журналы серверов) поисковой системы содержат различную информацию о действиях, которые совершали пользователи по отношению к ней. Обычно такая запись содержит уникальный идентификатор пользователя, дату и время действия, параметры обращения к поисковой системе. Основные действия, записываемые в поисковые логи, это *запросы*, задаваемые через поисковую форму, и *клики* — переходы по ссылкам на документы в результатах поиска.

Записи можно сгруппировать по конкретному пользователю, получится последовательность запросов и кликов — *поисковая сессия*. В статье [18] рассматривается проблема разделения (сегментации) поисковой сессии на *логические сессии*, подпоследовательности действий пользователя, каждая из которых соответствует одному определённому интенту пользователя. Для оценки качества разделения на сессии авторы предлагают использовать метрику *Rand Index*:

$$R = \frac{n_1 + n_2}{C_n^2},$$

где n_1, n_2 — количество пар действий пользователей, которые в ручном и в проверяемом разделении принадлежат одной логической сессии, n — общее количество разделяемых действий.

Одним из самых простых способов сегментации является разделение по длительности пауз: если временной промежуток между действиями пользователя превышает некоторый наперёд заданный порог T, считается, что начата новая логическая сессия. Именно такой подход и используется в данной дипломной работе. Авторами статьи замечено, что при выборе значений $T \in [6; 45]$ минут качество сегментации меняется незначительно: $R \in [0,69; 0,72]$. Для сравнения стоит отметить, что для построенного в работе алгоритма сегментации значение $R = 0,86$.

1.2. Графовые модели

Популярным способом представления данных, получаемых из поисковых сессий, являются различные модели на графах.

Query-Flow Graph. В статье [22] следующим образом определяется понятие графа запросов: граф G_{qf} это ориентированный граф $G_{qf} = (V, E, w)$, где:

- $V = Q \cup \{s, t\}$ — множество вершин, состоящее из множества запросов Q, заданных данным пользователем поисковой системе, и специальных состояний: стартового s и терминального t, символизирующих соответственно начало и конец логической сессии;
- $E \subseteq V \times V$ — множество рёбер;
- $w : E \to (0; 1]$ — весовая функция.

Построение графа запросов производится на основе множества логических сессий $\mathbb{S} = \{S_1, \ldots, S_m\}$. Две вершины, соответствующие запросам, соединяются ребром в случае, если эти два запроса появляются последовательно хотя бы в одной логической сессии S_j, иными словами второй запрос является прямой переформулировкой первого. Множество таких рёбер T формально определяется следующим образом:

$$T = \{(q, q') | \exists S_j \in S : q = q_i \in S_j \wedge q' = q_{i+1} \in S_j\}$$

Для выбора весовой функции авторы предлагают два варианта. Первая вычислялась при помощи машинного обучения с использованием 18 текстовых, сессионных и временных признаков на основе бинарной разметки пар запросов, могут ли они принадлежать одной логической сессии или нет. Вторая основана на относительной частоте переформулировки запросов. Пусть запрос q встречается в данной выборке сессий $f(q)$ раз, $f(q, q')$ раз запрос q' встречается непосредственно после запроса q, $f(s, q)$ и $f(q, t)$ — столько раз запрос q является соответственно первым и последним в сессии, тогда весовая функция определяется следующим образом:

$$w(q, q') = \frac{f(q, q')}{f(q)}$$

Авторы статьи предлагают в качестве приложения модели графа запросов применить её для построения рекомендаций запросов (подсказок, которые зачастую показываются поисковой системой, чтобы пользователь мог уточнить исходный запрос), используя цепи Маркова и случайное блуждание по ним. Более подробно задача рекомендаций рассматривается этими авторами в их последующей работе [14]: используются различные срезы графа запросов, производится ручная оценка и анализ результатов.

В работе [13] предлагается спроецировать полученный граф запросов в некоторое евклидово пространство меньшей размерности, а затем, используя в качестве функции сходства

$$Sim(q, q') = \frac{1 + cos(q, q')}{2},$$

кластеризовать запросы. Полученные результаты также применимы для построения уточнений.

Click Graph. В работе [3] для кластеризации запросов используется отличная от QFG модель. Кликовый граф G_c — это двудольный неориентированный граф $G_c = (V, E, w)$, где:

- $V = Q \cup D$ — множество вершин, состоящее из множества запросов Q и множества документов D (в оригинальной статье в качестве документов рассматриваются изображения);
- $E \subseteq Q \times D$ — множество рёбер, символизирующих переходы на документ в поисковой выдаче по запросу;
- w — весовая функция, $w(q, d)$ — количество переходов на документ d, отображаемый в результатах поиска по запросу q.

На основе данного графа строится марковская цепь, вероятность перехода в то же самое состояние постулируется равной параметру s, а вероятность перехода между различными состояниями вычисляется как нормированный вес ребра в графе:

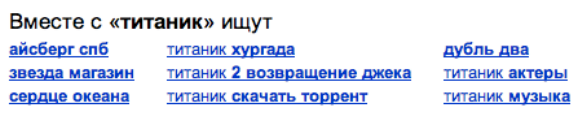
$$P_{t+1|t}(k|j) = (1 - s) \cdot \frac{w(j, k)}{\sum\limits_i w(j, i)}$$

Граф запросов и документов совмещает в себе обе вышеизложенные модели: вершины содержат как запросы, так и документы, рёбра между запросами означают переформулировку запроса, а между запросом и документом — переход. В статье [2] авторы описывают марковскую модель на основе этого графа, она используется в данной дипломной работе, поэтому её построение будет рассмотрено более подробно далее в разделе 3.2.

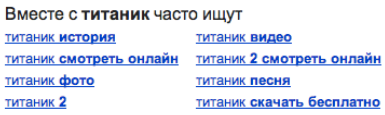
1.3. Исследования переформулировок

Большое количество работ посвящено анализу последовательности запросов, заданной пользователем в рамках поисковой сессии. Для удобства введём следующиее понятие: запрос q_2 является *(прямой) переформулировкой* запроса q_1, если запрос q_2 встречается хотя бы в одной поисковой сессии (непосредственно) после запроса q_1.

Зачастую результаты, полученные в результате исследований переформулировок, применяют к задаче построения множества уточнений исходного запроса. Многие поисковые системы предлагают после результатов поиска ссылки на другие запросы, которые могут помочь более точно раскрыть поисковую цель пользователя. На рисунках 2 и 3 приведены примеры того, как это может быть реализовано в интерфейсе.



а) «Яндекс»



б) «Google»

Рис. 2: Подсказки по запросу *титаник* в различных поисковых системах.

В статье [7] вводится понятие *подмены запроса*, нового запроса, сгенерированного поисковой системой и призванного заменить исходный запрос с целью показать пользователю релевантную рекламу, которой может не быть по изначальным ключевым словам. Для построения подмены предлагается находить в поисковых логах пары подменяемых по-

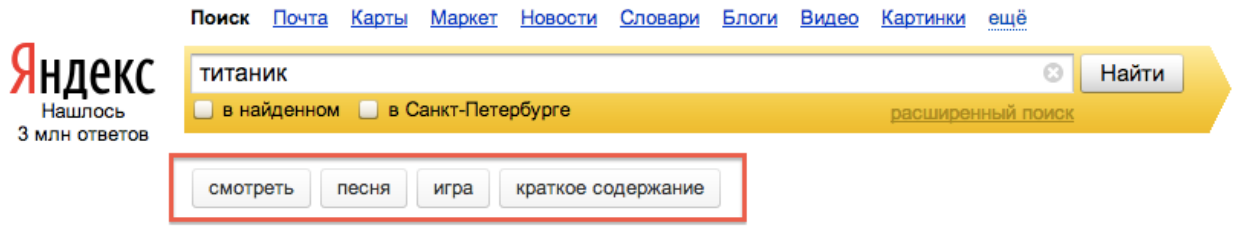


Рис. 3: Диалоговые кнопки уточнения запроса под поисковой строкой «Яндекса».

нятий, таковыми могут быть как целые запросы с переформулировками, так и некоторые их части. Пары-кандидаты ранжируются при помощи методов машинного обучения на 37 признаках: к ним относятся различные текстовые характеристики исходного запроса и его подмены, синтетические признаки, такие как редакционное расстояние, количество общих слов и т.п., похожие признаки, рассчитываемые по паре в целом.

В работе [19] рассматриваются слова и словосочетания, из которых составлены поисковые запросы, два шаблона переформулирования запроса: замена слов с использованием контекста и добавление слов, предлагаются методы для выделения этих шаблонов из поисковых сессий.

Различные модели и шаблоны переформулировок продолжают исследоваться в работах [8] и [6], авторы первой делают попытку построить формальную модель, выделив 13 типов переформулировок. Построенный ими классификатор имеет точность 98.2% и полноту 61.3%. Во второй работе рассматривается меньшее количество, а именно пять стратегий:

- **G** (generalization) — обобщение темы исходного запроса;
- **S** (specialization) — сужение темы исходного запроса;
- **C** (correction) — исправление ошибок или опечаток;
- **P** (parallel move) — переход к связанной, но другой теме;
- **X** (mission change) — кардинальная смена темы поиска.

На основе данных типов был построен классификатор с точностью принятия решений 92%. Последовательности, составленные из нескольких

шагов, называются шаблонами (например, XC, SG, GS ...), на основе которых авторы строят искомые уточнения.

В работе [16] особое внимание уделяется низкочастотным запросам. На основе шаблонов замены отдельных слов и словосочетаний при переформулировании, а также иерархии обобщения понятий авторы строят граф, состоящий из запросов и шаблонов запросов, при помощи которого, подменяя понятия на более общие, создаются уточнения исходного запроса.

1.4. Кластеризация запросов

Как будет описано далее, одним из шагов полученного в данной дипломной работе алгоритма является кластеризация запросов.

В работах [1] и [10] кластеризация производится на основе двудольго графа. В первой работе граф $G = G(V, E)$ устроен следующим образом: множество вершин V состоит из запросов Q и документов D, две вершины соединены ребром (q, d) в том случае, если документ $d \in D$ присутствует в поисковой выдаче по запросу $q \in Q$. Обозначив множество соседей вершины $v \in V$ как $\mathcal{N}(v)$, авторы вводят функцию сходства двух вершин следующим образом:

$$\sigma(x, y) = \begin{cases} \frac{|\mathcal{N}(x) \cap \mathcal{N}(y)|}{|\mathcal{N}(x) \cap \mathcal{N}(y)|}, & \text{если} |\mathcal{N}(x) \cap \mathcal{N}(y)| > 0 \\ 0, & \text{иначе.} \end{cases}$$

Для кластеризации используется агломеративный алгоритм: на каждом шаге попеременно «склеиваются» в одну вершины, соответствующие запросам и документам и имеющие наибольшее значение функции сходства σ; условие остановки — нулевые значения сходства для всех пар документов и всех пар запросов:

$$\max_{q_i, q_j \in Q} \sigma(q_i, q_j) = 0 \wedge \max_{d_i, d_j \in D} \sigma(d_i, d_j) = 0$$

Авторы второй работы сначала предлагают несколько изменить функцию сходства, заменив количество соседей на суммарное количество пе-

реходов, а затем и вовсе использовать ключевые слова из сниппетов[8] вместо документов при построении двудольного графа. Предлагается алгоритм их получения. Дальнейшие исследования посвящены персонализации поисковых подсказок.

Следующие две работы используют для кластеризации запросов дополнительные данные. В первом случае авторы [20] работают со специфической поисковой системой — поиском по FAQ[9]. Переформулировки запроса не рассматриваются вовсе, а понятие сессии сужается до «запрос + переходы по его результатам». В статье приводится несколько возможных функций сходства, однако они используют такие специфические для данной поисковой системы свойства запросов, как относительно большая длина, наличие вопросительного слова в начале и т.п., также в распоряжении авторов работы есть четырёхуровневая иерархия всех документов.

Во втором случае, в работе [17] авторы ставят задачу определения интента запроса, чтобы добавить в выдачу результаты из подходящего тематического поиска (к примеру, поиска по картинкам, видеозаписям или картам). К сожалению, в данной статье не рассматриваются многозначные запросы, а в качестве возможных интентов авторы рассматривают только три: *путешествия*, *имена* и *работа*. В качестве дополнительных данных используется ориентированный граф, построенный на основе графа страниц Википедии[10]. В вершинах графа находятся статьи и категории этой энциклопедии, а рёбра соответствуют гипертекстовым ссылкам между ними. Особенно отмечается, что ссылка одной статьи на другую не обязательно означает, что они содержат близкую по смыслу информацию, поэтому рёбра между ними включаются в построенный граф в обоих направлениях.

Для кластеризации запросов авторы [2] используют вектора предельных вероятностей, полученные после случайного блуждания по построенной ими на запросах и документах марковской цепи, а в качестве

[8]Здесь под *сниппетом* понимается небольшой отрывок текста страницы, найденной поисковой системой, обычно содержащий контекст исходного запроса.

[9]Frequently Asked Questions — часто задаваемые вопросы.

[10]http://wikipedia.org

алгоритма — Complete Linkage.

В работе [15] рассматривается идея использовать кластеризацию запросов для определения наборов интентов пользователя. В качестве кластеризуемого множества выступает набор запросов построенный на основе переформулировок (авторы берут $k = 10$ самых частотных переформулировок исходного запроса, добавляя к ним по k самых частотных переформулировок от каждой из них). Полученные кластера соответствуют интентам пользователя.

1.5. Обзор используемых технологий

Данная дипломная работа выполнена автором в компании «Яндекс», поэтому несколько слов стоит уделить используемым её технологиям.

Yandex MapReduce. MapReduce — это модель распределённых вычислений, предложенная компанией «Google» для упрощения процесса обработки данных на больших кластерах компьютеров [4]. Каждое вычисление в этой парадигме представляет собой преобразование входного множества пар ключ-значение в выходное множество пар ключ-значение. Для реализации вычисления пользователь должен представить его в виде двух функций `MAP` и `REDUCE`. Первая (`MAP`) принимает на вход пару ключ-значение и выдаёт на выходе множество промежуточных пар ключ-значение. Производится группировка этих пар по ключу, почле чего они передаются на вход второй операции (`REDUCE`). Пользовательская реализация операции `REDUCE` принимает на вход какой-то ключ и множество соответствующих ему значений, она необходимым образом агрегирует их и может выдать на выходе полученное значение.

В компании «Яндекс» используется несколько модифицированная реализация [23]. Все данные на кластере организованы в *таблицы*, каждая таблица состоит из *записей*, каждая запись — это тройка из *ключа*, *подключа* и *значения*. Поддерживаются две базовые операции: `MAP` и `REDUCE`, однако они выполняются по отдельности и не обязательно в связке `MAP` – `REDUCE`, как это изначально предлагалось.

Операции `MAP` на вход в произвольном порядке поступают записи из одной или нескольких таблиц, производится их обработка и на выходе результаты записываются также в одну или несколько таблиц.

Операция `REDUCE` принимает на вход ключ и все записи для данного ключа, а на выходе выдаёт произвольное множество записей для одной или нескольких таблиц. При чтении из нескольких таблиц добавляется шаг сортировки промежуточной таблицы по ключам.

Библиотека Yandex MapReduce предоставляет возможность создавать утилиты для распределённых вычислений на языке программирования C++, реализуя простые интерфейсы базовых операций.

Хранилище документов. Поисковый робот «Яндекса» регулярно обходит граф Интернета, выкачивает документы и сохраняет их содержимое в базе [26]. К сожалению, в свободном доступе нет описания этой контент-системы, которая использовалась автором для получения текстов документов.

2. Обзор решения

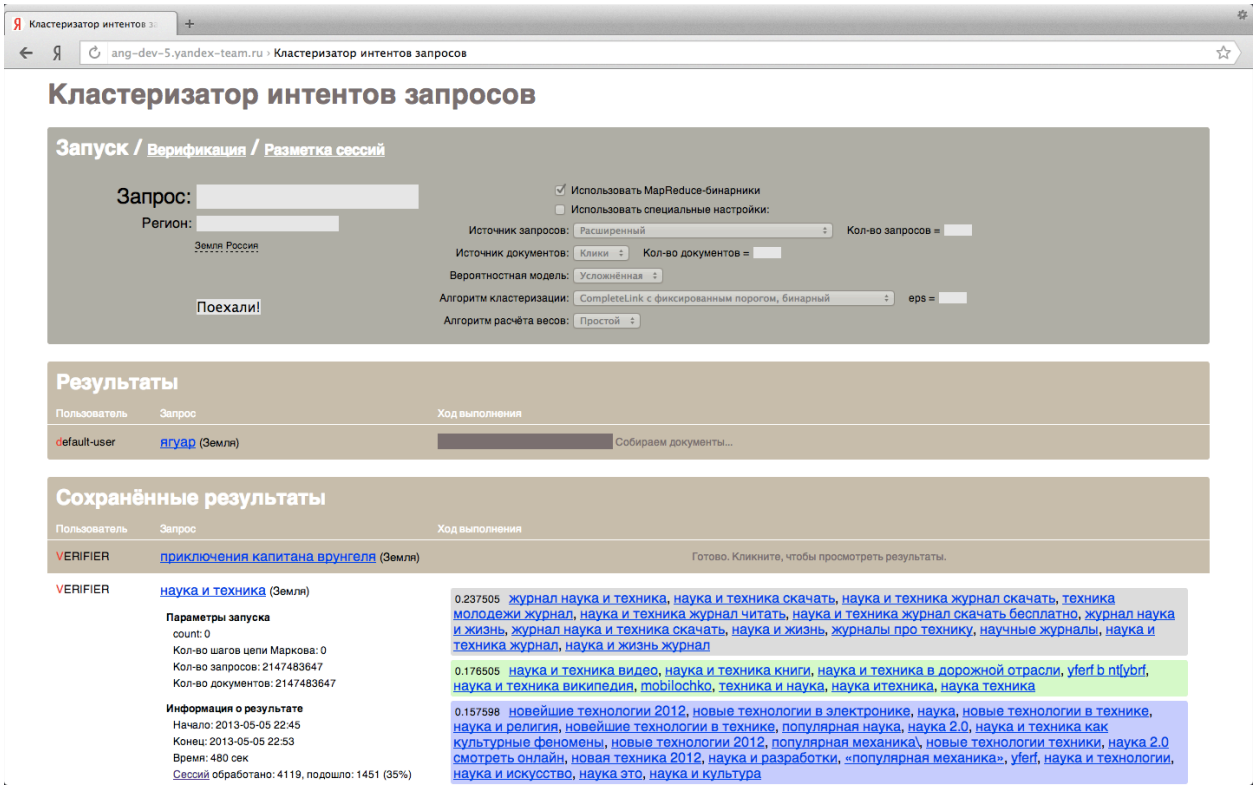


Рис. 4: Главная страница веб-сервиса.

Одной из задач данной дипломной работы было создание сервиса для запуска расчётов весов интентов и анализа полученных результатов. Предполагается, что основными пользователями продукта будут поисковые аналитики компании «Яндекс». В качестве решения была выбрана клиент-серверная архитектура по следующим причинам:

- Расчёты занимают продолжительное время, требуют больших вычислительных мощностей, доступ к кластеру MapReduce.
- Для работы с веб-сервисом не требуется установка дополнительного программного обеспечения на компьютер пользователя, достаточно веб-браузера.
- Программы-утилиты, участвующие в расчётах, ввиду наложенных сторонним программным обеспечением ограничений, являются платформенно-зависимыми.

- Архитектура веб-сервиса является внутренним стандартом компании для решения таких задач.

Взаимодействие пользователя с системой расчётов происходит следующим образом:

- Пользователь заходит на главную страницу сервиса (см. рис. 4), где в специальную форму вводит текст запроса и регион, из которого он был задан, затем запускает расчёт.
- На серверной стороне этот расчёт ставится в очередь, которая обрабатывается параллельно в несколько потоков. За прогрессом выполнения можно следить в списке запущенных расчётов.
- После завершения расчёта пользователь может посмотреть его результаты (см. главу 3), случайные выборки сессий, вспомогательную информацию о запуске.

Также в веб-сервис были встроены системы для ручной разметки тестовых данных и для оценки качества полученных результов (см. главу 5).

3. Общий алгоритм

Данная глава содержит общее описание построенного алгоритма без технических подробностей его реализации. На вход алгоритму поступает запрос, на выходе получается набор кластеров с их весами, каждый кластер содержит набор запросов, соответствующих интенту, определение которого производится уже пользователем. Алгоритм можно представить в виде последовательного выполнения следующих шагов:

1. Построение набора связанных запросов
2. Построение графа запросов и документов
3. Построение марковской модели
4. Представление запросов в векторном пространстве
5. Кластеризация набора запросов
6. Классификация логических сессий

Далее в этой главе каждый шаг будет подробно рассмотрен.

3.1. Построение набора связанных запросов

Первым шагом алгоритма является построение набора запросов, связанных с исходным. Именно это множество будет впоследствии разбито на кластеры, соответствующие интентам.

Во-первых, важно то, как пользователь переформулирует многозначный запрос, возможно добавляя уточняющие слова. Поэтому в множество связанных запросов следует включить прямые переформулировки исходного запроса. Чтобы избежать шума, учитывались только те переформулировки, которые сделали хотя бы два человека, этот экспериментально полученный порог позволяет устранить некоторые редкие ненужные переформулировки, при этом не принимая за шум низкочастнотные, но релевантные.

Во-вторых, один и тот же документ может присутствовать в поисковой выдаче по различным запросам. Если предположить, что тематика

одного документа едина и мало изменяется во времени, то запросы, имеющие общие документы в выдаче, похожи. В множество связанных запросов были также включены все те, что имеют хотя бы один общий документ в выдаче с исходным.

Технические подробности сбора этой статистики по поисковым сессиям приведены в разделе 4.1.

Для устранения шума был составлен фиксированный список из 100 самых часто задаваемых запросов (к примеру, в него попали запросы *вконтакте*, *одноклассники*, *гугл* и др.). Из множества связанных запросов удаляются все запросы, попавшие в составленный стоп-лист.

3.2. Построение графа запросов и документов

Следующим шагом алгоритма является построение графа запросов и документов, аналогичного описанному в работе [2].

Здесь и далее $N_{click(q_i,d_j)}$ — количество переходов на документ d_j как результат в поисковой выдаче по запросу q_i, $N_{ref(q_i,q_j)}$ — количество переформулировок из запроса q_i в запрос q_j, $\epsilon \in (0,1)$ — параметр, определяющий выбор дальнейшего действия пользователя, в работе он был выбран равным 0.5 (согласно рекомендациям, приведённым в статье [2]). Итак, $G = (V, E, w)$ — это ориентированный взвешенный граф, где:

- $\mathbf{V} = Q \cup D$ — множество вершин состоит из множества связанных запросов Q, построенного в предыдущем разделе, а также множества документов D, в него попадают все документы, на которые были переходы хотя бы с одного запроса $q \in Q$.

- $V \times V \supseteq \mathbf{E} = E_{click} \cup E_{ref} \cup E_{self}$ — множество рёбер состоит из ребёр трёх типов, для них определяется весовая функция $\mathbf{w} : E \to [0,1]$:

  - $E_{click} \subseteq Q \times D$ — рёбра, соответствующие переходам с запросов на документы, с весами, пропорциональными количеству переходов на данный документ d_j как результат в поисковой выдаче по данному запросу q_i, нормированному на общее ко-

личество переходов с данным запросом.

$$w(q_i, d_j) = P_{click}(q_i, d_j) = \frac{\epsilon \cdot N_{click(q_i,d_j)}}{\sum\limits_{d_k \in D} N_{click(q_i,d_k)}}, \quad \text{где } q_i \in Q, d_j \in D \tag{1}$$

– $E_{ref} \subseteq Q \times Q$ — рёбра, соответствующие переформулировкам, с весами, пропорциональными количеству переформулировок из запроса q_i в запрос q_j, нормированному на общее количество переформулировок из данного запроса.

$$w(q_i, q_j) = P_{ref}(q_i, q_j) = \frac{(1-\epsilon) \cdot N_{ref(q_i,q_j)}}{\sum\limits_{q_k \in Q} N_{ref(q_i,q_k)}}, \quad \text{где } q_i, q_j \in Q \tag{2}$$

– $E_{self} = \{(d_i, d_i) \mid d_i \in D\}$ — петли с единичным весом во всех документах, их назначение будет объяснено в разделе 3.3.

$$w(d_i, d_i) = 1, \quad \text{где } d_i \in D \tag{3}$$

На рисунке 5 приведён пример такого графа для пяти запросов и шести документов, вершина q_0 символизирует исходный многозначный запрос.

3.3. Построение марковской модели

Построенный граф можно рассматривать как граф переходов однородной марковской цепи $\{X_n\}_{n \geq 0}$, а вероятность перехода определить через весовую функцию рёбер:

$$\mathbf{P}_{xy} = P\{X_{n+1} = y \mid X_n = x\} = w(x, y) \quad \forall n \geq 0,\ x, y \in V \tag{4}$$

Для удобства будем считать, что все вершины в графе пронумерованы, а в обозначениях использовать элементы множеств запросов и документов как индексы, подразумевая под ними номера соответствующих вершин. Покажем, что полученная матрица переходных вероятностей $\mathbf{P}$ действительно стохастическая. Согласно определению (3) для

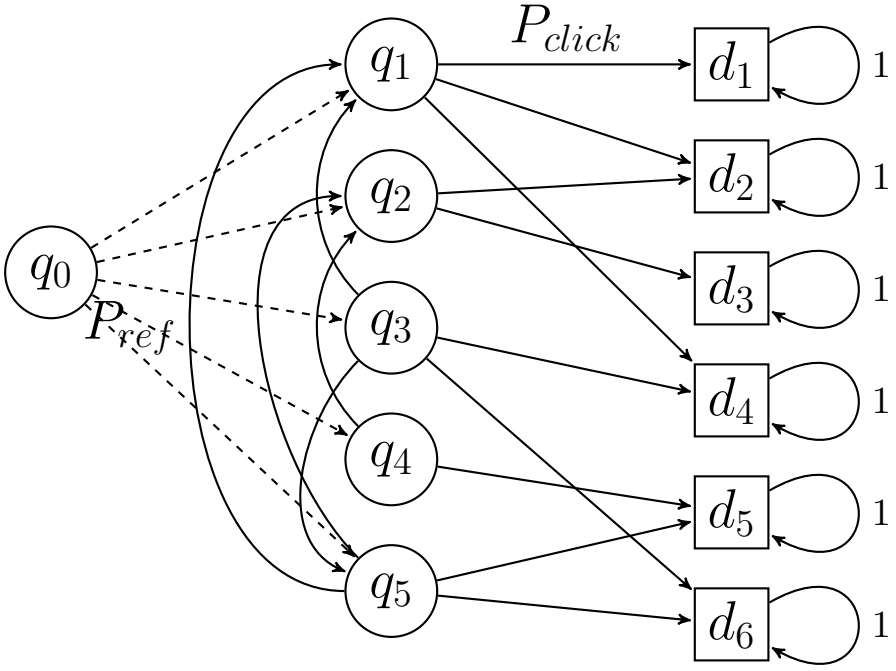


Рис. 5: Граф переходов марковской цепи (вершина q_0 в него не входит).

вершины-документа есть только одно исходящее ребро и это петля с весом 1, рассмотрим сумму вероятностей переходов из вершин-запросов:

$$\sum_{v \in V} \mathbf{P}_{qv} = \sum_{q' \in Q} \mathbf{P}_{qq'} + \sum_{d \in D} \mathbf{P}_{qd} = \sum_{q' \in Q} \frac{(1-\epsilon) \cdot N_{ref(q,q')}}{\sum\limits_{q'' \in Q} N_{ref(q,q'')}} + \sum_{d \in D} \frac{\epsilon \cdot N_{click(q,d)}}{\sum\limits_{d' \in D} N_{click(q,d')}} \quad (5)$$

$$= (1-\epsilon) \cdot \frac{\sum\limits_{q' \in Q} N_{ref(q,q')}}{\sum\limits_{q'' \in Q} N_{ref(q,q'')}} + \epsilon \cdot \frac{\sum\limits_{d \in D} N_{click(q,d)}}{\sum\limits_{d' \in D} N_{click(q,d')}} = (1-\epsilon) + \epsilon = 1 \quad (6)$$

Второй переход в выкладке (5) выполнен согласно определению переходной вероятности (4) и определениям весовой функции (1) и (2).

3.4. Представление запросов в векторном пространстве

Полученную марковскую модель можно проинтерпретировать следующим образом. Переход между состояниями-запросами — это переформулировка, а переход на состояние-документ — это клик по резуль-

тату поисковой выдачи. Случайное блуждание, начатое в одном из состояний, соответствующем запросу, по пути через несколько переформулирований так или иначе заканчивается в поглощающем состоянии, соответствующем документу. Параметр модели ϵ в данном случае отвечает за выбор следующего действия пользователя: продолжит он переформулировки или завершит блуждание переходом на документ. В дальнейшем данный параметр был принят равным 0.5.

Рассмотрим случайное блуждание, начатое в состоянии $q \in Q$, т.е. вектор начального распределения p_0 имеет только одну ненулевую компоненту на позиции, соответствующей этому запросу. Согласно уравнению Колмогорова—Чепмена матрица переходных вероятностей за n шагов является степенью исходной матрицы: $\mathbf{P}^n$. Чтобы получить распределение вероятностей после n шагов нужно умножить начальную вектор-строку на степень матрицы переходных вероятностей:

$$x^n = x^0 \cdot \mathbf{P}^n \tag{7}$$

С учётом особенности начального распределения и тождества (7), чтобы получить распределение вероятностей «оказаться» за n шагов в каждом из документов (вектор $P_{doc}(q, n)$), можно взять элементы с индексами, соответствующими документам, из соответствующей запросу строки матрицы P^n:

$$P_{doc}(q,n)^\top = \left(p^n_{qd_1}, p^n_{qd_2}, \ldots, p^n_{qd_k}\right) = \left((0,0,\ldots,0,1,0,\ldots,0)\cdot\mathbf{P}^n\right)_D = \left(\mathbf{P}^n\right)_{D\times\{q\}} \tag{8}$$

Таким образом для каждого запроса получено его представление в векторном пространстве размерности $|D|$, которое можно использовать для применения различных видов кластеризации. Эксперименты с различными значениями параметра n рассматриваются в главе 5.

В данной работе предлагается ещё один способ получения векторного представления запроса, заключается он в добавлении ещё одного шага. Текст каждого документа представляет собой набор слов W_d. Объединив их по всем документам, получим множество слов: $W = \bigcup_{d\in D} W_d$ (под *словом* здесь понимается несколько подряд идущих букв, произ-

водится приведение слов к нижнему регистру). Введём функцию $freq : W \times D \to [0, 1]$, сопоставляющую слову и документу относительную частоту слова в нём. Перенумеровав слова и используя их в качестве индексов, получим представление документа в векторном пространстве размерности $|W|$:

$$d \mapsto (freq(w_1, d), freq(w_2, d), \ldots, freq(w_m, d)), \quad \text{где } w_i \in W, d \in D \quad (9)$$

Используя (8) и (9) сопоставим запросу вектор в пространстве, индексированном словами, по следующему правилу:

$$P_{word}(q, n)^\top = \left(\sum_{d \in D} p_{qd}^n \cdot freq(w_1, d), \ldots, \sum_{d \in D} p_{qd}^n \cdot freq(w_m, d) \right) \quad (10)$$

В итоге мы имеем два способа получения векторов, соответствующих запросам: $P_{doc}(q, n)$ и $P_{word}(q, n)$, далее в главе 5 будет произведено сравнение использующих эти способы вариантов алгоритма.

3.5. Кластеризация набора запросов

В предыдущем разделе были предложены два способа проецирования запроса в векторное пространство, вне зависимости от выбранного способа перед нами стоит задача кластеризации множества векторов $X = \{x^1, x^2, \ldots, x^N\} \subseteq \mathbb{R}^k$.

В качестве функции сходства[11] использовался косинус угла между векторами, принимаемые им значения лежат в отрезке $[0, 1]$, поскольку по построению (8) и (10) все компоненты векторов положительны:

$$Sim(x, y) = \frac{\langle x, y \rangle}{||x|| \cdot ||y||} = \frac{\sum_{i=1}^{k} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{k} x_i^2} \cdot \sqrt{\sum_{i=1}^{k} y_i^2}} \quad (11)$$

Кластеризация производилась при помощи алгоритма *Complete Linkage*, относящегося к классу агломеративных иерархических алгоритмов. Его

[11] Similarity function

суть заключается в следующем:

1. Изначально предполагается, что каждый вектор принадлежит отдельному кластеру: $C_1 = \{x^1\}, C_2 = \{x^2\}, \ldots, C_N = \{x^N\}$.

2. Для каждой пары кластеров вычисляется их сходство как минимальное значение сходства между их элементами:

$$Sim(C, C') = \min_{x \in C, x' \in C'} Sim(x, x')$$

3. Выбирается пара кластеров с наибольшим значением сходства:

$$s = Sim(C_i, C_j) = \max_{C, C'} Sim(C, C')$$

4. Выбранная пара кластеров объединяется в один: $C_i, C_j \mapsto C_i \cup C_j$.

5. Если s больше некоторого наперёд заданного порогового значения θ, то переходим опять к шагу 2, иначе кластеризация завершается.

3.6. Классификация логических сессий

В результате кластеризации получается несколько кластеров с запросами, каждый из которых представляет определённый интент пользователя: $\{C_1, C_2, \ldots, C_n\}$. Заключительным шагом алгоритма является расчёт оценок весов этих интентов. Здесь в качестве оценки вероятности интента используется доля сессий с данным интентом среди всех сессий с определённым интентом и содержащих данный запрос.

Определение принадлежности сессии интенту производится следующим образом. В нашей модели логическая сессия пользователя представляет собой конечную последовательность действий пользователя: запросов и переходов на документы.

$$S = \{q_1, d_2, q_3, \ldots, q_{n-1}, d_n\}$$

Сопоставим каждому действию пользователя вектор с весами интентов. Запросу, принадлежащему одному из кластеров, соответствует вектор

с одной ненулевой компонентой, 1 на позиции, соответсвующей этому кластеру:

$$w(q) = (0, \ldots, 0, \underset{i}{1}, 0, \ldots, 0), \quad \text{если } q \in C_i \subseteq Q \tag{12}$$

Если принадлежность запроса кластеру (а соответственно и интенту) определяется однозначно, то один и тот же документ может быть связан с несколькими запросами, которые в свою очередь могут принадлежать разным кластерам. В определении (8) использовались p^n_{qd} — вероятности оказаться за n шагов в документе d после запроса q. Определим вектор весов для документа следующим образом:

$$w(d) = \frac{1}{||w'(d)||} \cdot w'(d), \quad \text{где } w'(d)_i = \sum_{q \in C_i} p^n_{qd} \text{ и если } d \in D \tag{13}$$

Теперь, используя (12) и (13), определим весовой вектор сессии S. Сначала отфильтруем все действия, про запросы или документы из которых нам ничего не известно ($q \notin Q \vee d \notin D$), затем усредним веса векторов, полученные для каждого действия в отдельности:

$$w(S) = \frac{1}{|S'|} \cdot \sum_{x \in S'} w(x), \quad \text{где } S' = S \cap (Q \cup D) \text{ и если } S' \neq \emptyset \tag{14}$$

И вычислим искомые веса интентов, усреднив значения весов отдельных сессий (при этом сессии, где нет ни одного известного действия, игнорируются):

$$w(\mathbb{S}) = \frac{1}{|\mathbb{S}'|} \cdot \sum_{S \in \mathbb{S}'} w(S), \quad \text{где } \mathbb{S}' = \{S \in \mathbb{S} \mid S \cap (Q \cup D) \neq \emptyset\} \tag{15}$$

4. Особенности реализации

4.1. Подготовка данных пользовательской статистики

В работе алгоритма используются статистические данные, собираемые на основе пользовательских сессий. Обработанные логи хранятся в таблицах на кластере MapReduce. Каждая запись соответствует одному действию пользователя. Формат записей в общем виде представлен в таблице 1.

Ключ	Подключ	Значение
ID пользователя	Дата и время	Тип действия (запрос, переход, ...), Сервис (веб-поиск, картинки, карты, ...), Параметры (URL, текст запроса, ...), ...

Таблица 1: Формат записи о действии в пользовательских сессиях.

В дальнейшем в качестве действий рассматриваются только запросы и переходы на документы, для удобства введём следующие обозначения: $Q_i = (u_i, t_i, q_i)$ — запрос пользователя u_i в момент времени t_i и текстом q_i; $C_j = (u_j, t_j, d_j)$ — переход пользователя u_j в момент времени t_j на документ с URL-адресом d_j; тогда сессия отдельного пользователя за сутки представляет собой конечную последовательность из запросов и переходов: $S = (Q_1, C_2, C_3, \ldots, Q_{n-1}, C_n)$, где $\forall i, j\ u_i = u_j,\ t_i \leq t_j$.

В работе использовались сессии поисковой системы «Яндекс», собранные за 84-хдневный период с 17 сентября по 9 декабря 2012 года.

Статистика по переформулировкам рассчитывается при помощи трёх операций `REDUCE`:

1. Первая обрабатывает набор сессий за одни сутки от каждого пользователя в отдельности, разбивая их на логические по интервалу между действиями пользователя в 10 минут. Для каждой пары запросов (Q_i, Q_{i+k}) добавляется переформулировка из запроса Q_i

в запрос Q_{i+k}. Отдельно помечаются прямые переформулировки (когда между Q_i и Q_{i+k} нет других запросов).

2. Вторая операция агрегирует переформулировки с одинаковым начальным и конечным запросом.
3. Третья агрегирует статистику, рассчитанную за каждые сутки из указанного периода.

Помимо основной записывается информация о количестве уникальных пользователей её сделавших и о количестве переходов на документы после переформулировки. Полученная статистика также используется поисковыми аналитиками и лежит в основе нескольких факторов ранжирования.

В том формате, в котором хранятся поисковые сессии, получение **всех сессии, содержащих данный запрос**, занимает длительное время, ведь быстрый поиск по таблице на кластере реализован только по ключам, а текст хранится в значении записи. Поэтому понадобилась утилита, формирующая выборку сессий с данным запросом. На вход поступает файл, содержащий тексты запросов, для которых необходимо построить индекс сессий, на выходе получается множество таблиц, содержащих в названии значение хеш-функции от текста запроса (для удобного поиска), а внутри множество записей с ключом-идентификатором логической сессии, подключом, содержащим время и тип действия, и значением с информацией о действии. Помимо поиска нужных сессий производилось разбиение их по времени на логические (интервал 10 минут), фильтрация только переходов и запросов и только нужной информации об этом действии.

Как уже упоминалось выше в разделе 3.1 в работе алгоритма используется **статистика по запросам с переходами на общие документы**. Исходными данными для расчёта является уже подготовленная таблица со статистикой по переходам за выбранный период в формате, приблизительно описанном в таблице 2. Сбор запросов с общими кликами производится в три операции (одна `MAP` и две `REDUCE`):

1. Первая операция `MAP` порождает таблицу, в которой ключ и под-

Ключ	Подключ	Значение
Текст запроса	URL-адрес документа	Статистика: количество показов количество переходов ...

Таблица 2: Формат записи о статистике переходов на документы.

ключ меняются ролями.

2. Затем при помощи операции `REDUCE` для данного документа-ключа на выход выписываются все пары документов (ключ и подключ) с произведением CTR[12] в качестве весов в значении.
3. Последняя операция `REDUCE` агрегирует статистику по парам запросов, суммируя веса.

В данной работе вес не используется.

4.2. Использование MapReduce для расчётов

Кластера MapReduce используются не только для запуска различных операций в этой парадигме, но также и для хранения данных. Для доступа к ним предоставляется HTTP-интерфейс, позволяющий загрузить определённую таблицу целиком, либо часть её записей, содержащихся в заданном диапазоне ключей. В первых версиях алгоритма использовалось небольшое количество связанных запросов, при этом значительное время тратилось на загрузку статистики с кластера. С увеличением на порядок количества связанных запросов возникла проблема с возведением в степень матрицы переходов на этапе случайного блуждания по марковской цепи. Размерность матрицы на некоторых запросах имеет порядок 10^5, что при использовании кубического алгоритма требует порядка 10^{15} операций и за разумное время невыполнимо на одном сервере даже при наличии нескольких вычислительных ядер.

[12]Click-through rate — показатель кликабельности, отношение количества переходов к количеству показов.

В связи с вышеперечисленными проблемами было принято решение перенести часть расчётов на кластер MapReduce. Для этого на языке C++ с использованием библиотеки Yandex MapReduce (см. раздел 1.5) были созданы две утилиты, вызываемые через создание дочернего процесса из Java-машины.

Первая принимает на вход файл, содержащий связанные запросы, и строит матрицу переходов при помощи трёх операций `REDUCE`: чтение данных о переходах, чтение данных о переформулировках и добавление единичных рёбер в документах. Матрица хранится в формате, описанном в таблице 3.

Ключ	Подключ	Значение
Индекс строки: запрос или документ	Индекс столбца: запрос или документ	Значение в ячейке матрицы

Таблица 3: Формат записи в таблице, содержащей матрицу.

Вторая утилита производит возведение в степень матрицы в описанном формате. Используется быстрое возведение в степень (количество умножений порядка логарифма от степени). Алгоритм перемножения двух матриц как операция `JOIN` описывается в работе [12]. Созданная его реализация требует одну операцию `MAP` и две `REDUCE` и получена из следующей модификации определения:

$$(\mathbf{A} \cdot \mathbf{B})_{ij} = \sum_{k=1}^{n} A_{ik} \cdot B_{kj} = \sum_{k=1}^{n} A^{\top}_{ki} \cdot B_{kj} \tag{16}$$

Первым действием первая матрица транспонируется (выполняется операция `MAP`, которая меняет местами ключ и подключ и помечает записи из первой таблицы), затем в операции `REDUCE` для каждого ключа k считаются всевозможные произведения $A^{\top}_{ki} \cdot B_{kj}$, после чего ещё одна операция `REDUCE` суммирует значения для пар индексов (i, j).

4.3. Создание тестовых наборов многозначных запросов

Для тестирования и оценки качества алгоритма (см. главу 5) нужны тестовые данные — многозначные запросы. Сбор их производился полуавтоматическим способом:

1. Для получения запросов-кандидатов на многозначность использовалась русскоязычная Википедия, в которой есть специальный раздел «Категория:Многозначные термины» [21], содержащий все статьи с меткой «Шаблон:Неоднозначность». Была написана небольшая программа, которая скачивала все страницы этого раздела и выделяла из их содержимого названия многозначных статей.
2. Затем полученные названия нормализовались заменой знаков препинания на пробелы и приведением букв к нижнему регистру, получались возможные многозначные запросы. Среди них получилось 44 508 уникальных. Для каждого из них выбиралась статистика, сколько раз они были заданы пользователями поисковой системы «Яндекс» за указанный выше период. Выбирались те, у кого это количество лежит в отрезке от 1000 до 10 000, это было сделано, чтобы исключить низкочастотные, для которых может быть недостаточно данных для работы алгоритма, и высокочастотные, где может быть слишком много шума. Получилось 11 650 запросов с такой частотностью.
3. Однако не все полученные запросы являются действительно многозначными в терминах данной работы. Дело в том, что зачастую на Википедии в качестве значений могут выступать узкоспецифические, которые не встречаются в поисковых запросах пользователей «Яндекса». Поэтому для формирования итоговой тестовой выборки, была сделана случайная выборка большего размера, из которой вручную выбирались нужные действительно многозначные.

4.4. Особенности реализации веб-сервиса

Сервис для запуска и анализа расчётов реализован на языке программирования Java с использованием сервера Jetty[13] и библиотеки Spring Framework[14]. Выбор данных технологий обусловлен тем, что именно они используются в проекте, в команде которого состоит автор дипломной работы и который предоставляет подготовленную кодовую базу для создания подобных веб-сервисов.

Конечному пользователю предоставляется несколько динамических веб-страниц:

- Главная страница сервиса позволяет запускать расчёты по отдельным запросам, настраивая их параметры или используя значения по умолчанию. Также на ней показывается список активных расчётов и расчётов в очереди, в том числе и запущенных другими пользователями, ход выполнения расчёта сопровождается пояснениями каждого шага алгоритма. На главной странице также отображается архив всех выполненных расчётов с их результатами.
- Страница «Верификация» предназначена для единовременного запуска расчётов по заранее загруженному набору запросов, возможно указать те же настройки, что и на главной странице, а задачи попадают в общую очередь. Для уже выполненных запусков предусмотрена возможность расчёта различных метрик качества и прочих связанных с ними значений (см. главу 5).
- Отдельный сервис предназначен для ручной разметки связанных запросов по классам, изначально на странице разметки показываются запросы, сгруппированные по кластерам, найденным алгоритмом, пользователь может вносить в данное разбиение любые изменения, предусмотрено отдельное поле, куда попадают запросы, интент которых непонятен пользователю. Распределение запросов по классам производится при помощи курсора. Результаты

[13]http://www.eclipse.org/jetty/
[14]http://www.springsource.org/

разметки используются при расчёте метрик качества кластеризации (см. раздел 5.1).

- Сервис разметки сессий позволяет создавать и размечать выборки сессий по запросу. При разметке есть возможность добавлять обнаруженные новые интенты, а пометка сессии определённым интентом производится нажатием соответствующей цифры на клавиатуре.

Каждая страница реализована в виде класса-сервиса, наследующего базовый для всех сервисов класс. Все вызовы сервисов принимают на вход параметры запроса, а на выходе генерируют XML-документ, после чего на него накладывается XSLT-шаблон[15], в результате получается XHTML-документ, который передаётся веб-браузеру пользователя. Для построения дизайна интерфейсов использовался язык стилей CSS[16], многие интерактивные элементы выполнены на языке JavaScript[17] с использованием библиотеки JQuery[18], для фоновой передачи данных использовалась технология AJAX[19].

[15]http://www.w3.org/TR/xslt20/
[16]http://www.w3.org/Style/CSS/
[17]http://en.wikipedia.org/wiki/JavaScript
[18]http://jquery.com/
[19]http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

5. Эксперименты и оценки

5.1. Качество кластеризации

Кластеризация запросов — один из основных этапов работы алгоритма, влияющий впоследствии на точность расчёта весов. При помощи описанного в разделе 4.4 инструмента были разбиты на классы 30 наборов запросов (в среднем по 165 в каждом). Для каждого такого набора (соответствующего многозначному запросу) имеется множество *кластеров*, полученных в результате работы алгоритма, и множество *классов*, полученных при ручной разметке.

Введём обозначения: SS, SD, DS, DD — количество пар элементов кластеризуемого множества, где оба элемента принадлежат одному/разным кластерам и одному/разным классам, первому условию соответствует первая буква ("S" — одному, "D" — разным), а второму — вторая; например, SD — элементы принадлежат одному кластеру, но разным классам. Были использованы четыре метрики оценки качества чёткой кластеризации [24]: Rand Index (17), Jaccard Index (18), Folkes-Mallows Index (19) и F1-мера (20).

$$Rand = \frac{SS + DD}{SS + SD + DS + DD} \tag{17}$$

$$Jaccard = \frac{SS}{SS + SD + DS} \tag{18}$$

$$FM = \sqrt{\frac{SS}{SS + SD} \cdot \frac{SS}{SS + DS}} \tag{19}$$

Для каждого кластера c_i и класса g_j определим:

- n_{ij} — количество запросов, попавших в c_i и g_j одновременно,
- n_i — количество элементов кластера c_i,
- n_j — количество элементов класса g_j,
- $Precision(i, j) = \frac{n_{ij}}{n_i}$ — точность,
- $Recall(i, j) = \frac{n_{ij}}{n_j}$ — полнота,

- $F1(i,j) = \frac{2 \cdot Precision(i,j) \cdot Recall(i,j)}{Precision(i,j) + Recall(i,j)}$ — F1-мера.

Тогда:

$$F1 = \sum_j \frac{n_j}{N} \max_i F1(i,j) \tag{20}$$

В проводимых экспериментах для построения доверительных интервалов и проверки значимости различий между значениями метрик использовались бутстреп[20] и критерий перестановок[21] [11].

Тестовый эксперимент. При помощи критерия перестановок на уровне значимости 0.1% для всех значений параметров (за исключением крайнего $\theta = 1$) и для обоих вариантов представления запроса в векторном пространстве (по документам и по словам) была отвергнута гипотеза о том, что полученная в результате работы алгоритма кластеризация не отличается от случайной. Проверка производилась следующим образом: каждый результат кластеризации это набор кластеров определённого размера, затем производилось 9999 перестановок, каждая из которых заменяла истинный результат кластеризации на случайный, но с кластерами таких же размеров.

Запуском будем называть набор расчётов, запущенных с одинаковыми параметрами для всех запросов из размеченного вручную множества. Для каждого запроса можно расчитать значение метрики качества, а усреднив по всем запросам, получить оценку качества конкретного запуска.

Первый эксперимент был посвящён зависимости качества кластеризации от значения порогового параметра θ (см. раздел 3.5) при фиксированном количестве шагов случайного блуждания по марковской цепи (см. раздел 3.5) и использованию векторов, основанных на словах в документах (см. раздел 3.4). Была проведена серия запусков расчётов на тестовых данных с значениями $\theta \in [0, 1]$ с шагом 0.1, получены выборки из 30 значений для каждой метрики и каждого запуска. Согласно технике бутстрепа из них было составлено 1000 выборок с повторениями такого же размера, вычислены средние значения метрик, а

[20]Bootstrap

[21]Permutation test

затем среднее значение метрик после бутстрепа. График зависимостей средних значений метрик от параметра представлен на рис. 6.

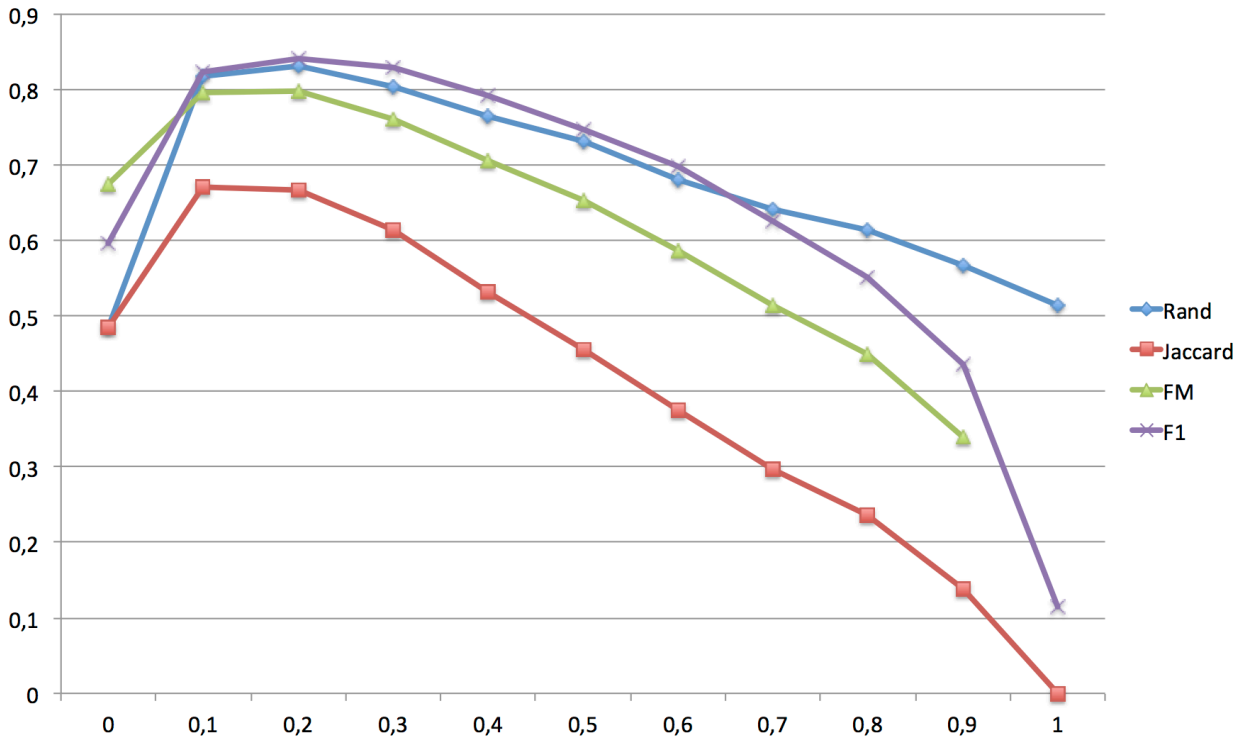
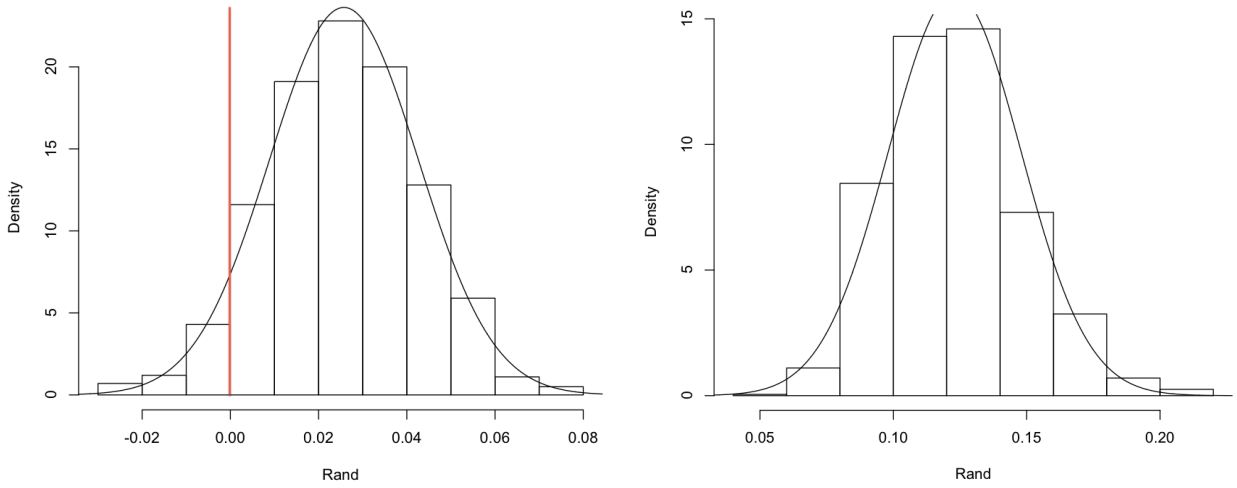


Рис. 6: Зависимость метрик качества от порогового значения θ при кластеризации, $n = 16$, вектора $P_{word}(q)$.

Судя по графику, можно сделать предположение о том, что при значении $\theta \in [0.1, 0.3]$ по совокупности метрик качество кластеризации выше, чем при остальных значениях параметра.

Сравнение двух запусков производилось следующим образом. Пусть $\{x_1, x_2, \ldots, x_{30}\}$ — значения какой-то из метрик на запросах при первом запуске, а $\{y_1, y_2, \ldots, y_{30}\}$ — при втором. В качестве нулевой гипотезы выступает предположение о том, что запуски ничем не отличаются: $x = y$ или $d = x - y = 0$. Составляем выборку, состоящую из попарных разностей: $\{d_1 = x_1 - y_1, d_2 = x_2 - y_2, \ldots, d_{30} = x_{30} - y_{30}\}$. Затем с полученной выборкой проведём бутстрепинг, построим гистограмму полученных средних значений (см. рис. 7). Если 0 попадает в промежуток между 0.5% и 99.5% квантилями, то это означает, что нулевую гипотезу нет оснований отклонять на уровне значимости 1% (как, к примеру, при сравнении $\theta = 0.2$ и $\theta = 0.3$, рис. 7а), в противном случае

гипотеза отвергается на уровне значимости 1%, т.е. запуски различаются, при чём если 0 меньше 0.5% квантили, то x лучше y (см. рис. 7б), а если 0 больше 99.5% квантили, то y лучше x.



а) $\theta = 0.2$ и $\theta = 0.3$ б) $\theta = 0.3$ и $\theta = 0.6$

Рис. 7: Сравнение двух запусков при различных значениях θ, $n = 16$.

Так в результате сравнений выяснилось, что по всем метрикам запуски со значениями $\theta \in \{0.1, 0.2, 0.3\}$ между собой не различаются, однако значимо (на уровне 1%) лучше остальных. При $\theta \geq 0.4$ с возрастанием θ качество кластеризации ухудшается.

Второй эксперимент был направлен на изучение вопроса о целесообразности добавления шага получения векторов по словам в документе, не достаточно ли ограничиться предложенным в работе [2] способом, основанном только на документах? Сначала была произведена серия запусков, аналогичная предыдущему эксперименту, только в качестве кластеризуемых векторов использовались вектора на документах. На рисунке 8 изображён график зависимости значений используемых метрик качества от порога θ. При попарном сравнении запусков выяснилось, что запуски со значениями $\theta = 0.0$ и $\theta = 0.1$, а также $\theta = 0.0$ и $\theta = 0.2$ на уровне 1% значимо не различаются. При $\theta \geq 0.1$ запуск с бóльшим значением θ по всем метрикам качества значимо (на уровне 1%) хуже меньшего.

Для сравнения двух подходов были выбраны запуски кластеризации векторов на словах со значением порога $\theta = 0.2$ и на документах со

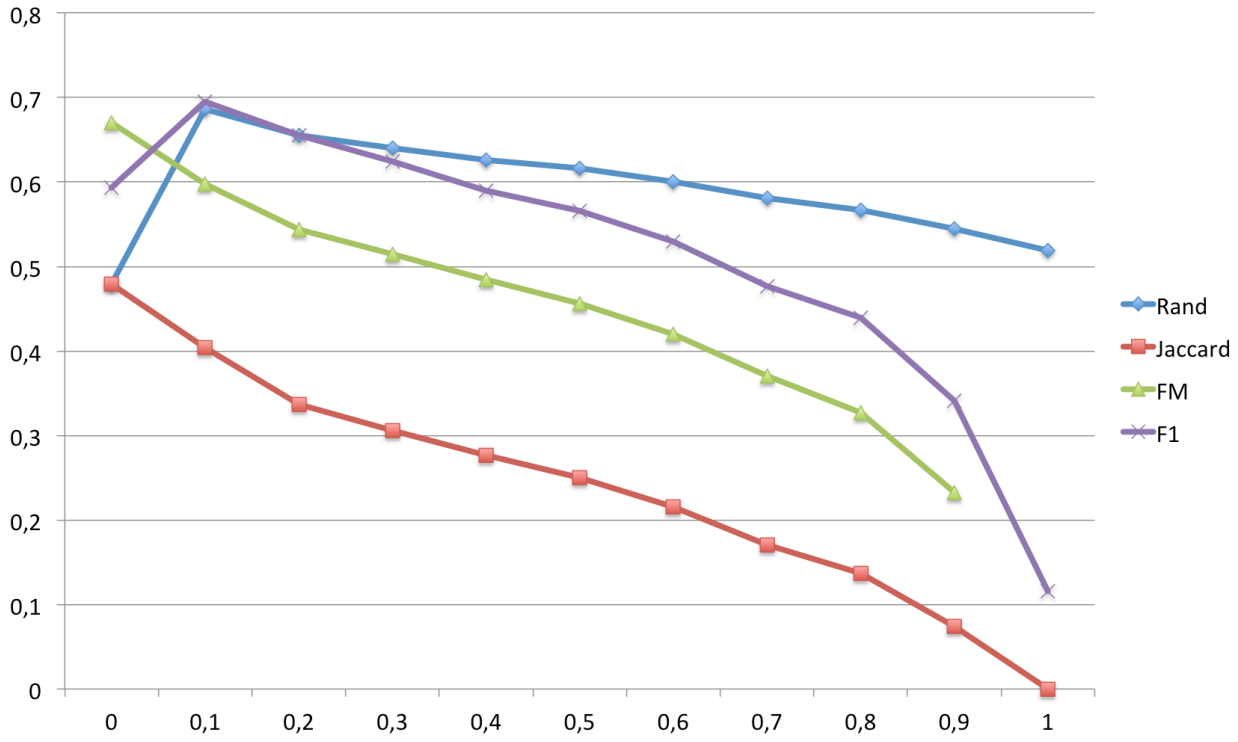


Рис. 8: Зависимость метрик качества от порогового значения θ при кластеризации, $n = 16$, вектора $P_{docs}(q)$.

значением порога $\theta = 0.1$ при одинаковом количестве шагов случайного блуждания $n = 16$. Значения метрик качества по выбранным запускам представлены на рис. 9. Сравнение, проведённое бутстрепом для парных наблюдений, показало, что кластеризация векторов на словах значимо (на уровне 1%) лучше кластеризации векторов на документах, т.о. модификация алгоритма оправдана.

Третий эксперимент должен был пролить свет на зависимость качества кластеризации от количества шагов n, выполненных при случайном блуждании по построенной марковской цепи, и на то, есть ли смысл вообще его проводить. Для удобства вычислений проводилось сравнение запусков со следущими значениями n: 1 (т.е. блуждания как такового нет, вектор документов формируется из оценок вероятностей перехода), 2, 4, 8 и 16. Как лучшие по итогам предыдущих экспериментов были зафиксированы параметр порога $\theta = 0.2$ и способ кластеризации векторов на словах.

На рисунке 10 представлены графики зависимости значений мет-

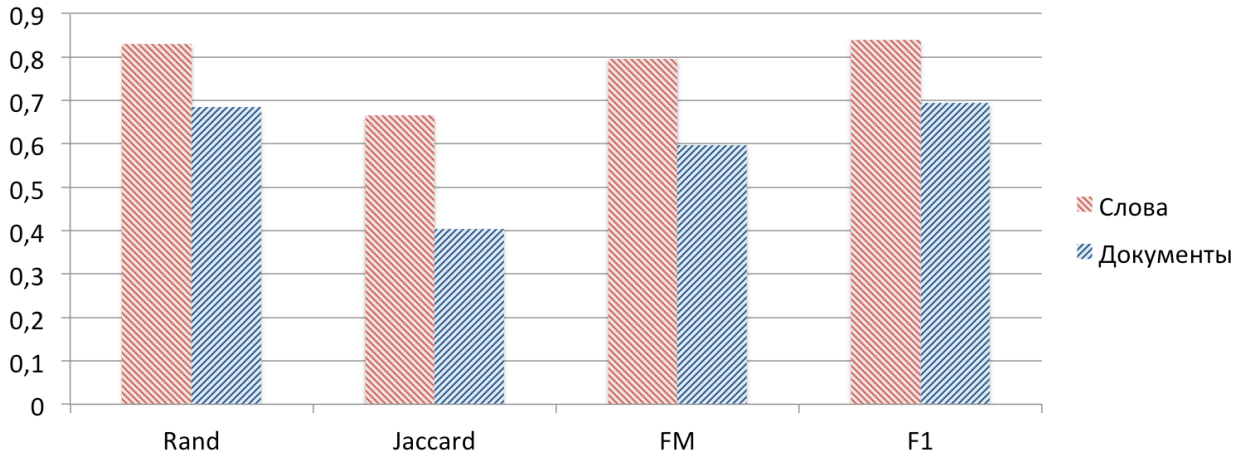


Рис. 9: Сравение значений метрик качества для кластеризации векторов на словах (при $\theta = 0.2$, $n = 16$) и на документах (при $\theta = 0.1$, $n = 16$).

рик качества от выбранных значений количества шагов n. Попарное сравнение бутстрепом даёт следующие результаты: запуски с соседними значениями (к примеру, 1 и 2, 2 и 4, 8 и 16) на уровне значимости 1% не различимы, однако если сравнивать небольшие значения n (1 или 2) с бóльшими (8 или 16), то различия становятся значимыми на этом уровне. В результате можно сделать вывод, что на выбранных тестовых данных случайное блуждание вносит вклад в качество результата при количестве шагов $n \geq 8$.

5.2. Качество расчёта весов

Исходя из того, как рассчитываются итоговые веса (сопоставление каждой сессии вектора с весами отдельных интентов), можно рассматривать этот процесс как нечёткую кластеризацию сессий. В работе [9] предлагается способ рассчитывать Rand Index для нечёткой кластеризации, адаптируем его для нашего случая.

Пусть $C = \{C_1, C_2, \ldots, C_m\}$ — множество кластеров связанных запросов, $I = \{I_1, I_2, \ldots, I_n\}$ — множество интентов, на которые вручную были размечены сессии. $P(S) = (P_1, P_2, \ldots, P_m)$ — вектор весов сессии, $Q(S) = (Q_1, Q_2, \ldots, Q_n)$ — ручная разметка данной сессии (только одна координата Q_i отлична от нуля, та, которая соответствует интенту сессии). В контексте данного разбиения определим функцию сходства

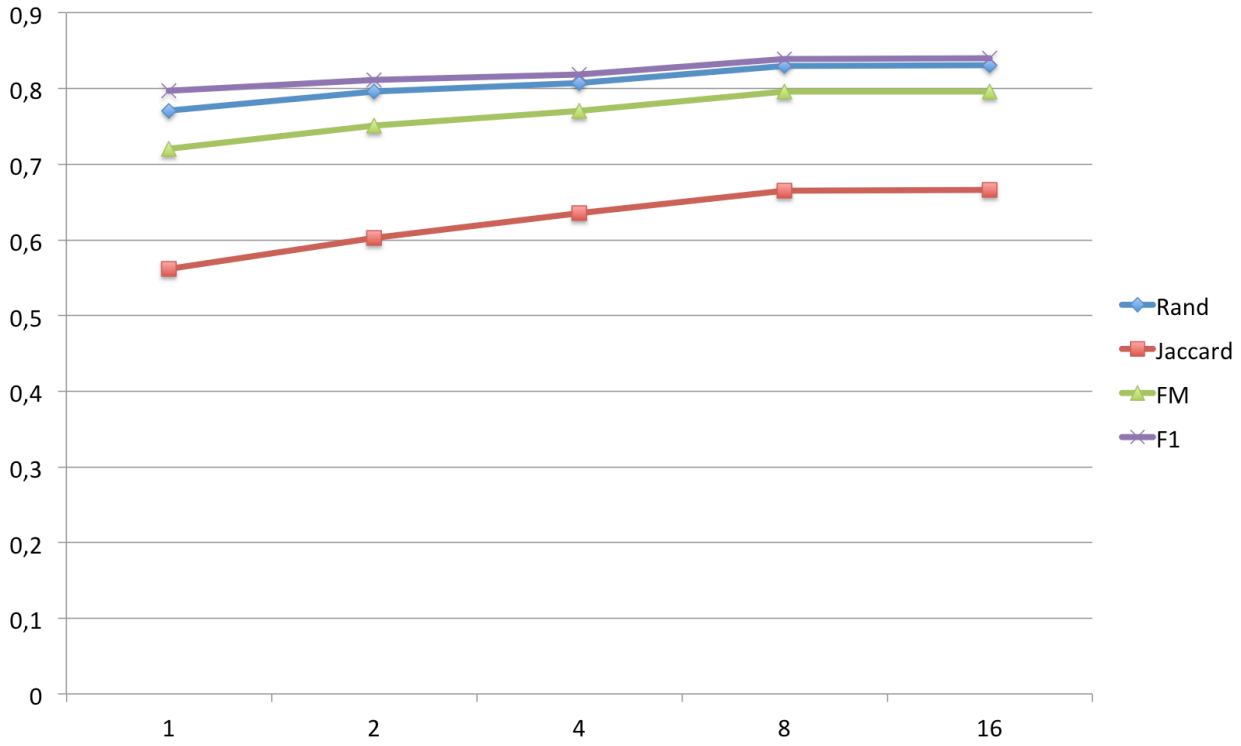


Рис. 10: Зависимость метрик качества от количества шагов случайного блуждания, $\theta = 0.2$, вектора $P_{word}(q)$.

двух сессий:

$$E_P(S, S') = 1 - ||P(S) - P(S')|| \tag{21}$$

Здесь $||.||$ — норма в пространстве $[0, 1]^{dimP}$, значения которой должны принадлежать отрезку $[0, 1]$ — единственное ограничение, которое накладывают на неё авторы статьи. В качестве такой нормы был выбран максимум:

$$||x|| = \max_i |x_i| \tag{22}$$

Пусть $\mathbb{S}$ — некоторая случайная выборка сессий, содержащих исходный запрос; $\mathbb{S}_{marked} \subseteq \mathbb{S}$ — множество сессий, для которых удалось вручную определить интент; $\mathbb{S}_{marked} \subseteq \mathbb{S}$ — множество сессий, для которых алгоритм определил веса интентов. Далее определяются мера несогласия (23), нормированная сумма этих мер (24) по всем сессиям в размеченной вручную выборке $\mathbb{S}_{marked}$ и наконец сам аналог Rand Index для нечёткой кластеризации (25):

$$|E_P(S, S') - E_Q(S, S')| \tag{23}$$

$$d(P,Q) = \frac{\sum\limits_{S,S' \in \mathbb{S}_{marked} \cap \mathbb{S}_{matched}, S \neq S'} |E_P(S,S') - E_Q(S,S')|}{C_n^2} \tag{24}$$

$$Rand_{fuzzy} = 1 - d(P,Q) \tag{25}$$

Помимо $Rand_{fuzzy}$ введём ещё две метрики, оценивающие качество расчёта весов. Для каждой сессии из набора при ручной разметке и при работе алгоритма определяется наличие хоть какого-то интента. Так что расчёт весов можно рассматривать как бинарный классификатор, определяющий наличие интента. Первая метрика его качества — аналог точности — отношение $Precision_{sessions} = \frac{|\mathbb{S}_{marked} \cap \mathbb{S}_{matched}|}{|\mathbb{S}_{matched}|}$, отвечает на вопрос, какая доля сессий из определенных алгоритмом была также определена и вручную. Вторая — аналог полноты — отношение $Recall_{sessions} = \frac{|\mathbb{S}_{marked} \cap \mathbb{S}_{matched}|}{|\mathbb{S}_{marked}|}$, отвечает на вопрос, какая доля сессий из определенных вручную была также определена и алгоритмом. В идеальной ситуации $\mathbb{S}_{marked} = \mathbb{S}_{matched}$ и обе метрики равны 1.

Для каждого из 30 тестовых запросов были случайным образом выбраны и размечены по 100 сессий, бутстреп доверительные интервалы метрик качества приведены в таблице 4, расчёт производился для запуска с кластеризацией векторов на словах, $\theta = 0.2$, $n = 16$.

Метрика	95% доверительный интервал
$Rand_{fuzzy}$	$(0.685, 0.717)$
$Precision_{sessions}$	$(0.864, 0.905)$
$Recall_{sessions}$	$(0.413, 0.466)$

Таблица 4: Значения метрик качества расчёта весов.

Заключение

В результате данной работы:

1. Разработан алгоритм расчёта оценок весов интентов для многозначных запросов. Алгоритм реализован на языке программирования Java с использованием утилит на C++ для обработки данных на кластере MapReduce;
2. Реализована система для запуска и анализа расчётов в виде веб-сервиса;
3. На размеченных вручную 30 многозначных запросах (в среднем по 165 связанных на каждый) и 3000 сессиях проведена оценка качества кластеризации и расчёта весов. Проведён сравнительный анализ запусков алгоритма с различными его параметрами.

Дальнейшая работа

Созданный сервис предоставляет удобный интерфейс для ручной разметки данных, и первое, что стоит сделать в продолжении данной работы, это собрать больше данных для тестирования качества алгоритмов.

Улучшение алгоритма возможно за счёт ряда изменений:

- Лингвистические данные, а именно приведение слов из документов к начальным формам уменьшит размерность пространства векторов и может улучшить качество кластеризации.
- Аккуратная работа с содержимым документов, изменяющимся со временем.
- При обработке сессий на этапе расчёта весов использование как дополнительных факторов последовательность действий, а также временные паузы между ними.
- Автоматическая обработка полученных кластеров с целью получения интента может вообще исключить человека из процесса расчёта весов и наборов интентов.

Пользовательский веб-сервис также может быть улучшен:

- Добавление возможности запускать сбор сессий, содержащих данных запрос, в интерфейс сервиса избавит пользователя от необходимости делать это вручную.
- Интеграция с существующими аналитическими инструментам компании позволит не производить никаких дополнительных действий при создании корзины многозначных запросов.

Список литературы

[1] Beeferman Doug, Berger Adam. Agglomerative clustering of a search engine query log // Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining / ACM. — 2000. — P. 407–416.

[2] Clustering query refinements by user intent / Eldar Sadikov, Jayant Madhavan, Lu Wang, Alon Halevy // Proceedings of the 19th international conference on World wide web / ACM. — 2010. — P. 841–850.

[3] Craswell Nick, Szummer Martin. Random walks on the click graph // Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval / ACM. — 2007. — P. 239–246.

[4] Dean Jeffrey, Ghemawat Sanjay. MapReduce: simplified data processing on large clusters // Communications of the ACM. — 2008. — Vol. 51, no. 1. — P. 107–113.

[5] Diversifying search results / Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, Samuel Ieong // Proceedings of the Second ACM International Conference on Web Search and Data Mining / ACM. — 2009. — P. 5–14.

[6] From Dango to Japanese cakes: Query reformulation models and patterns / Paolo Boldi, Francesco Bonchi, Carlos Castillo, Sebastiano Vigna // Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01 / IEEE Computer Society. — 2009. — P. 183–190.

[7] Generating query substitutions / Rosie Jones, Benjamin Rey, Omid Madani, Wiley Greiner // Proceedings of the 15th international conference on World Wide Web / ACM. — 2006. — P. 387–396.

[8] Huang Jeff, Efthimiadis Efthimis N. Analyzing and evaluating query reformulation strategies in web search logs // Proceedings of the 18th ACM conference on Information and knowledge management / ACM. — 2009. — P. 77–86.

[9] Hüllermeier Eyke, Rifqi Maria. A fuzzy variant of the Rand index for comparing clustering structures // Joint IFSA World Congress and EUSFLAT Conference, Lisbon, Portugal. — 2009. — P. 1294–1298.

[10] Leung KW-T, Ng Wilfred, Lee Dik Lun. Personalized concept-based clustering of search engine queries // Knowledge and Data Engineering, IEEE Transactions on. — 2008. — Vol. 20, no. 11. — P. 1505–1518.

[11] Moore David. Bootstrap Methods and Permutation Tests // The Basic Practice of Statistics. — WH Freeman, 2010.

[12] Myung Jaeseok, Lee Sang-goo. Matrix chain multiplication via multi-way join algorithms in MapReduce // Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication / ACM. — 2012. — P. 53.

[13] Query similarity by projecting the query-flow graph / Ilaria Bordino, Carlos Castillo, Debora Donato, Aristides Gionis // Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval / ACM. — 2010. — P. 515–522.

[14] Query suggestions using query-flow graphs / Paolo Boldi, Francesco Bonchi, Carlos Castillo et al. // Proceedings of the 2009 workshop on Web Search Click Data / ACM. — 2009. — P. 56–63.

[15] Radlinski Filip, Szummer Martin, Craswell Nick. Inferring query intent from reformulations and clicks // Proceedings of the 19th international conference on World wide web / ACM. — 2010. — P. 1171–1172.

[16] Szpektor Idan, Gionis Aristides, Maarek Yoelle. Improving recommendation for long-tail queries via templates // Proceedings

of the 20th international conference on World wide web / ACM. — 2011. — P. 47–56.

[17] Understanding user's query intent with wikipedia / Jian Hu, Gang Wang, Fred Lochovsky et al. // Proceedings of the 18th international conference on World wide web / ACM. — 2009. — P. 471–480.

[18] Ustinovskiy Yury, Mazur Anna, Serdyukov Pavel. Intent-Based Browse Activity Segmentation // Proceedings of the 35th European Conference on Information Retrieval. — 2013. — P. 242–253.

[19] Wang Xuanhui, Zhai ChengXiang. Mining term association patterns from search logs for effective query reformulation // Proceedings of the 17th ACM conference on Information and knowledge management / ACM. — 2008. — P. 479–488.

[20] Wen Ji-Rong, Nie Jian-Yun, Zhang Hong-Jiang. Clustering user queries of a search engine // Proceedings of the 10th international conference on World Wide Web / ACM. — 2001. — P. 162–168.

[21] Wikipedia. Категория:Многозначные термины // Википедия, свободная энциклопедия. — 2013. — URL: http://clck.ru/8cjVN (дата обращения: 26.12.2012).

[22] The query-flow graph: model and applications / Paolo Boldi, Francesco Bonchi, Carlos Castillo et al. // Proceedings of the 17th ACM conference on Information and knowledge management / ACM. — 2008. — P. 609–618.

[23] Игнатьев Александр. Инфраструктура MapReduce // Яндекс.События. — 2012. — URL: http://clck.ru/8cjVR (дата обращения: 24.04.2013).

[24] Сивоголовко Елена. Оценка качества чёткой кластеризации. — 2011. — URL: http://synthesis.ipi.ac.ru/sigmod/seminar/sivogolovko20111124.pdf.

[25] Яндекс. Поиск Яндекса предлагает диалог. — 2012. — URL: http://company.yandex.ru/press_releases/2012/0321/index.xml (дата обращения: 10.05.2013).

[26] Яндекс. Индексирование интернета. — 2013. — URL: http://company.yandex.ru/technologies/searchindex/index.xml (дата обращения: 12.05.2013).

[27] Яндекс на РОМИП'2009. Оптимизация алгоритмов ранжирования методами машинного обучения / Андрей Гулин, Павел Карпович, Денис Расковалов, Илья Сегалович // Российский семинар по Оценке Методов Информационного Поиска. Труды РОМИП. — 2009. — P. 163–168.