Санкт-Петербургский государственный университет
Математико-механический факультет
Кафедра системного программирования

Алгоритм построения оценок весов интентов для многозначных запросов

Курсовая работа студента 445-ой группы
Григорьева Артёма Валерьевича

Научный руководитель ……. к. ф.-м. н. Грауэр Л.В.

Санкт-Петербург
2012

Оглавление

Введение

Проблема поиска в сети Интернет ввиду постоянного роста объёмов общедоступной информации с каждым годом становится всё актуальнее. Основная цель поисковых систем, таких как Яндекс или Google, – отвечать на запросы пользователей релевантными документами.

Для оценки качества поиска используются различные метрики. Метрика в данном смысле – это функция от запроса и списка документов. К примеру, pfound [1] вычисляется на основании оценок релевантности документов по данному запросу. Если из формулировки запроса ясно, какие документы считать релевантными, а какие нет, то выставить оценки по некоторой шкале – несложная задача для человека. Несколько сложнее дело обстоит с классом многозначных запросов. Запрос относится к данному классу, если по нему существует несколько возможных интентов – намерений пользователя. В таблице 1 приведены примеры таких запросов и возможных интентов для них.

Запрос	**Некоторые возможные интенты**
ягуар	марка автомобиля, животное, алкогольный напиток, фильм
титаник	история корабля, факты о фильме, скачать фильм, где посмотреть в 3D
танки	боевая техника, игра «Танки Онлайн», игра «World of Tanks»

Табл. 1. Примеры интентов для многозначных запросов

Значение метрики качества для многозначного запроса вычисляется по формуле:

$$metrics = \sum_{i=1}^{k+1} w_i \cdot metrics(O_i)$$

как взвешенное среднее значение метрики по каждому интенту O_i в отдельности с весами w_i – вероятностями того, что пользователь имел данный интент, задавая данный многозначный запрос [4].

В роли оценки для этих весов выступает доля сессий, в которых пользователь предполагал конкретное значение запроса, по отношению ко всем сессиям, содержащим данный запрос. Здесь и далее под сессией подразумевается последовательность действий пользователя по отношению к поисковой системе (запросы и клики на документы), заключённая в некотором временном отрезке. Задача разграничения отдельных сессий является предметом иных исследований и в данной работе не рассматривается. Для простоты будем считать, что пользователь начал новую сессию после паузы в 10 минут.

Вручную расчёт оценок весов интентов для конкретного запроса производится по следующему принципу. Из поисковых логов случайным образом выбирается некоторое небольшое количество сессий, содержащих данный запрос, которые затем вместе с предполагаемым набором интентов отправляются на разметку асессорам. Задачей последних является понять по действиям пользователя из данной сессии, к какому из интентов её можно отнести. Итоговые оценки получаются уже описанным выше способом: как доли сессий с данным интентом среди общего количества сессий в выборке.

Малый размер выборки и ручное составление набора интентов могут привести к неполноте и неточности полученных оценок. Эти серьёзные недостатки связаны прежде всего с ограниченными возможностями асессорской службы.

Так возникла тема данной работы. Требуется придумать и реализовать алгоритм, рассчитывающий оценки весов интентов для многозначных запросов. Для проверки качества его работы будут использоваться уже полученные оценки. Постараемся добиться не меньшей полноты полученных наборов интентов, а также незначительных различий в оценках при фиксированных наборах.

Обзор предыдущих работ

Первая работа, о которой хотелось бы упомянуть, стала во многом основополагающей для разработанного алгоритма. Речь о статье E. Sadikov и др. [2] о кластеризации уточняющих запросов по интентам пользователей. В ней рассматривается задача генерации релевантных подсказок по уточнению исходного запроса к поисковой системе, строится марковская цепь, содержащая в качестве состояний запросы и документы, производится случайное блуждание по ней, после чего запросы кластеризуются с использованием предельных вероятностей для документов.

В продолжение этой работы F. Radlinski и другие предлагают при помощи описанной модели выделять поисковые интенты для запроса [5], в дальнейшем изложении будет указано, почему их метод работает далеко не для всех классов запросов (см. метод частотных переформулировок, стр. 7).

Проблему кластеризации запросов к поисковой системе решают в своей работе Ji-Rong Wen и др. [3]. Несмотря на то, что они работают со специфическими данными (поиск по FAQ), из их работы можно почерпнуть сведения о различных используемых в предметной области функциях сходства[1].

Исследовать сходство запросов, проецируя граф запросов, предлагают I. Bordino и др. [6]. В их модели не рассматриваются документы, описываются различные весовые функции для рёбер, а кластеризация производится благодаря нахождению подграфов.

Более подробно о моделях графа запросов и графа документов рассказывается в работах N. Craswell и M. Szummer [7] и P. Boldi и др. [8] соответственно.

[1] Similarity function.

Алгоритм

Общая схема алгоритма

Итоговый алгоритм работает по следующей схеме. На вход поступает многозначный запрос q, затем с использованием различной пользовательской статистики определяется набор связанных с исходным запросов $\{q_i\}$. Данный набор кластеризуется, каждый кластер C_j соответствует интенту пользователя (определение этого значения по набору запросов в кластере остаётся за аналитиком). После кластеризации определение искомых оценок весов происходит аналогичным с ручной оценкой образом: выбирается набор случайных сессий, содержащих исходный запрос q, каждая сессия может с некоторыми весами относиться к различным кластерам. Среднее таких весов по всем сессиям из выборки и есть искомые оценки, получаемые вместе с самими кластерами на выходе.

Таким образом весь процесс можно разделить на три этапа:

1) Определение набора связанных запросов;
2) Кластеризация связанных запросов;
3) Распределение сессий по кластерам.

Далее опишем каждый этап в отдельности.

Эталонное множество запросов

Во введении были определены два основных требования к результатам работы алгоритма: полнота и точность. В качестве эталонного множества были выбраны 49 многозначных запросов, по которым есть наборы интентов, размеченные асессорами сессии и оценки для весов интентов. На всех шагах построения алгоритма производились расчёты по данному множеству запросов и выбиралось то из альтернативных решений, которые выигрывало у остальных в сравнении с асессорами. Этим обуславливается размер эталонного множества: все сравнения производились вручную.

Источник статистики

Источником статистики являются очищенные от роботов поисковые логи «Яндекса» за период в 84 дня с 26 декабря 2011 года по 18 марта 2012 года. Помимо самих логов в работе использовалась агрегированная позапросно статистика по кликам, обратным кликам и переформулировкам.

Запросы нормализуются приведением к нижнему регистру, удалением непечатных символов, заменой подряд идущих пробельных символов на один, удалением концевых

пробелов.

URL-адреса документов нормализуются приведением к нижнему регистру и к главным зеркалам доменов, удалением префикса *«www.»*, хеша, концевых слешей и вопросительных знаков.

Определение набора связанных запросов

Построение набора связанных запросов влияет на качество результата по двум параметрам. Во-первых, в набор должно входить хотя бы по одному запросу для каждого интента из некоторого идеального набора для данного запроса. Иначе не будет достигнута желаемая полнота. Во-вторых, несколько забегая вперёд, скажем, что при распределении сессий по кластерам происходит матчинг[2] по тексту. Таким образом нужно, чтобы в набор входили разнообразные формулировки одного и того же запроса. Это влияет на количество сессий, которые будут распределены по кластерам, и следовательно точность полученных оценок весов. Например, если в набор входит [андеграунд играть онлайн][3], то желательно добавить туда такие запросы, как [онлайн игра андеграунд], [играть андеграунд онлайн] и т.п..

В ходе экспериментов были опробованы различные варианты получения набора связанных запросов. Перечислим их далее в том же порядке, в котором производились эксперименты.

Частотные переформулировки. Определим понятие: запрос B является *переформулировкой* запроса A, если он встречается хотя бы в одной сессии после запроса A.

Зачастую пользователь переформулирует свой запрос, если он не удовлетворён поисковыми результатами. Предлагается в качестве исходного множества брать некоторое количество самых частотных переформулировок.

Общие клики. Минусом первого способа является необходимость иметь хорошую статистику о переформулировках запроса. В ходе экспериментов было замечено, что по тем запросам, где в поисковой выдаче присутствуют документы по большинству предполагаемых интентов, переформулировки содержат много «мусора». Действительно, пользователю незачем переформулировать запрос, если нужный документ уже найден.

Таким образом встал вопрос об интерпретации сделанных пользователем кликов. Нужно было каким-то образом понять, о чём тот или иной документ. Такую информацию можно извлечь из обратных кликов – статистики, где по URL-адресу документа хранится список запросов, по которым были клики по данному документу как по поисковому

[2] Для краткости назовём процесс отнесения действия пользователя к кластеру «матчингом» (от англ. match)
[3] Для обозначения текста поискового запроса внутри статьи будут использоваться квадратные скобки.

результату. Под способом получения связанных запросов «Общие клики» подразумевается следующее: для исходного запроса q выбираются несколько первых документов среди отсортированных по некоторому параметру (количеству кликов или CTR), для каждого из этих документов выбираются самые частотные запросы из статистики по обратным кликам и добавляются в множество связанных.

Смешанный способ. В смешанном способе результаты от двух предыдущих смешиваются в один список, который во избежание «шума» сортируется по относительной частотности. Для переформулировок это $\frac{ref(q \to q')}{\max ref(q \to .)}$, для общих кликов это $\frac{clicks(q,d)}{\max\limits_{d} clicks(q,d)} \cdot \frac{clicks(q',d)}{\max\limits_{q} clicks(q,d)}$. Из этого списка выбираются несколько (в экспериментах 20) первых запросов.

Расширенный способ. Как уже было упомянуто выше, значительную роль при обработке сессий играет текстовое совпадение запросов. В ходе экспериментов было замечено, что оно играет немаловажную роль и что зачастую из-за этого сессия остаётся нераспознанной (см. подробнее про обработку сессий далее на стр. 12). В расширенном способе к уже набранным запросам из предыдущего метода добавляются различные их варианты по следующему алгоритму: просматриваются случайные сессии по исходному запросу (те же, что обрабатываются на третьем этапе), для каждого запроса из них производятся попытки определить, похож ли он на какой-нибудь из уже имеющихся. Для этого оба запроса разбиваются на отдельные слова, затем проверяется, содержит ли запрос из сессий все слова из имеющегося. При чём совпадение проверяется не точное, а при условии, что нормированное расстояние Левенштейна не превосходит порогового значения (в данном случае 0.1).

При таком способе к запросу [снегурочка островский] будут добавлены: [краткое содержание снегурочки], [снегурочка островский смотреть онлайн] и т.п.

В таблице 2 приведены данные о сравнении трёх запусков на эталонном множестве запросов:

	Переформулировки	Смешанный	Расширенный
Количество запросов с полным набором интентов	28	31	32
Количество запросов с набором не более чем без одного интента	39	44	46
Средний процент обработанных сессий среди всех запросов	51%	52%	56%
Средний процент обработанных сессий среди запросов с полным набором интентов	59%	54%	58%

Табл. 2. Сравнение трёх вариантов получения набора связанных запросов

По данным таблицы можно сделать вывод, что расширенный способ получения набора связанных запросов выигрывает у остальных по таким параметрам, как полнота получаемых наборов интентов и количество обработанных на третьем этапе сессий.

Кластеризация связанных запросов

После того, как множество связанных запросов сформировано, следующим этапом работы алгоритма является разбиение этого множества на кластеры, соответствующие пользовательским интентам. В общем случае задача кластеризации сводится к двум этапам: сопоставлению каждому элементу из кластеризуемого множества вектора в некотором многомерном пространстве, а затем кластеризации этих векторов.

Идея кластеризации запросов, используемая в данной работе, почерпнута из статьи «Clustering Query Refinements by User Intent» [2]. По сравнению с оригинальной была несколько изменена вероятностная модель.

Построение марковской модели. Определим взвешенный ориентированный граф, моделирующий поведение пользователей при работе с поисковой системой. Для каждого запроса из множества связанных $Q = \{q_i, i = 1..N\}$ возьмём первые $L=100$ самых кликабельных[4] документов, объединение всех документов образует множество документов $D = \{d_k, k = 1..M\}$. Вершины в графе будут двух типов запросы $q \in Q$ и документы $d \in D$.

[4] Т.е. первые L среди всех (отсортированных по количеству кликов) документов, которые когда-либо показывались в качестве поисковых результатов по данному запросу.

Два запроса соединены ребром $q_1 \to q_2$, если конечный запрос является переформулировкой начального. Запрос и документ соединены ребром $q \to d$, если по данному документу были клики как результату в поисковой выдаче по данному запросу.

Определим веса рёбер как статистические оценки вероятностей кликов и переформулировок соответственно:

$$w(u \to v) = \begin{cases} \varepsilon \cdot ctr \cdot P_{click} \cdot D_{click}, u \in Q, v \in D \\ (1-\varepsilon) \cdot P_{reformulation} \cdot D_{reformulation}, u, v \in Q \\ 1, u = v \in D \end{cases},$$

где

$$ctr = \begin{cases} \dfrac{clicksCount(q,d)}{showsCount(q,d)}, showsCount(q,d) \neq 0 \\ 0, showsCount(q,d) = 0 \end{cases};$$

$clicksCount(q,d)$ – количество кликов по документу d как результату в поисковой выдаче по запросу q,

$showsCount(q,d)$ – количество показов документа d как результата в поисковой выдаче по запросу q.

$$P_{click} = \frac{clicksCount(q,d)}{\sum_{d' \in D} clicksCount(q,d')};$$

$$\mathrm{P}_{reformulation} = \frac{reformulationsCount(q_1,q_2)}{\sum_{q' \in Q} reformulationsCount(q_1,q')};$$

$reformulationsCount(q,q')$ – количество переформулировок запроса q в запрос q'.

Для обеспечения бóльшего правдоподобия данных статистики вводятся дисконтирующие множители за малое количество данных:

$D_{click} = \dfrac{1}{1-\exp(10-requests)}$, где *requests* – общее количество запросов пользователей.

$D_{reformulation} = \dfrac{1}{1-\exp(10-reformulations)}$, где *reformulations* – общее количество переформулирований данного запроса.

ε – нормирующий множитель, в экспериментах брался равным 0.6.

В граф добавлены петли с весом 1 в вершинах с документами.

Таким образом данный граф можно рассматривать как однородную цепь Маркова с поглощающими состояниями – документами. Данная модель не эмулирует поведение

пользователей полноценно: здесь клик на документ происходит после последовательности запросов и означает завершение поиска, однако по мнению автора отвечает задаче определения поискового интента: грубо говоря, пользователь переформулирует запрос, если не находит среди документов нужный, и кликает на один из представленных, если ему кажется, что он отвечает его нуждам.

Предельные вероятности. Рассмотрим матрицу смежности *P* для построенного выше графа. В ячейке *P[i,j]* стоит значение вероятности перехода из состояния *i* в состояние *j*. Если возвести данную матрицу в степень *n*, то в ячейке $P^n[i,j]$ будет стоять значение вероятности перейти из состояния *i* в состояние *j* не более чем за *n* шагов.

Таким образом, согласно интерпретации построенной модели, после возведения матрицы *P* в достаточно большую степень мы получим в ячейках, соответствующих переходам от запроса к документу, вероятности клика по данному документу после цепочки переформулировок данного запроса.

Введём понятие вектора *V(q)* – вектора предельных вероятностей документов для запроса *q*. В нём на *k*-ой позиции стоит значение ячейки предельной матрицы с координатами *i* – номером состояния, соответствующего запросу *q*, и *j* – номером состояния, соответствующего *k*-ому документу из предварительно отсортированного множества *D*:

$$V(q)_k = P^n[i, j]$$

Теперь, когда для каждого запроса получен вектор, ему соответствующий, можно перейти непосредственно к кластеризации запросов.

Кластеризация запросов. Получив для каждого запроса вектор в многомерном пространстве, мы свели задачу кластеризации запросов к задаче кластеризации этих векторов. Существует множество различных функций сходства и алгоритмов кластеризации, вопрос выбора подлежит дальнейшим исследованиям. В данном варианте алгоритма использовалась функция сходства *cosine-similarity* и алгоритм кластеризации *CompleteLink* [2].

$$sim(\overline{v}, \overline{w}) = \frac{\sum v_i w_i}{\sqrt{\sum v_i^2}\sqrt{\sum w_i^2}}$$

Кратко опишем алгоритм кластеризации. Изначально каждый запрос находится в собственном кластере. На каждом шаге перебираем все возможные пары кластеров и объединяем в один ту пару, для которой достигается максимальное значение *completeLink*:

$$completeLink(C_i, C_j) = \min_{v \in C_i, w \in C_j} sim(v, w)$$

Остановка алгоритма может производиться по двум вариантам условия. Во-первых,

можно задать необходимое количество кластеров, однако данный вариант не подходит ввиду того, что заранее это число неизвестно. Во-вторых, можно остановиться, если максимальное значение *completeLink* стало меньше некоторого порогового. Изначально был выбран этот способ со значением порога 0.05. Однако в дальнейшем было замечено, что остаются необъединённые кластеры с одинаковым интентом, в следствие чего порог был уменьшен до 0.01. Более подробное исследование содержимого кластеров показало, что по крайней мере на эталонном множестве запросов дальнейшее снижение порога не приводит к ошибочному объединению кластеров с различными интентами.

Распределение сессий по кластерам

На выходе этапа кластеризации мы имеем множество кластеров C_j, каждому из которых соответствует множество запросов $\{q_{jl}\}$, в него входящих, и множество документов $\{d_k\}$ с весами u_{jk}. Вес документа определяется как сумма предельных вероятностей клика по нему для каждого запроса из кластера: $u_{jk} = \sum_{q \in C_j} V(q)_k$

Напомним, что сессия пользователя в нашем рассмотрении состоит из последовательности действий: запросов и кликов. В ходе распределения сессий по кластерам последовательно обрабатываются сессии из выборки. Для каждого действия пытаемся получить вектор размерности равной количеству кластеров (K). Координаты этого вектора определяются по следующим правилам:

1) Для запроса (из списка связанных) полученный вектор содержит 1 на той позиции, номер которой совпадает с номером кластера, в котором данный запрос содержится, и 0 на всех остальных позициях.
2) Для клика по документу (из рассматриваемого набора) вектор состоит из его весов u_{jk} на j-ой позиции, соответствующей j-ому кластеру.
3) Все остальные действия (клики на документы не принадлежащие рассматриваемому набору и запросы с текстом не принадлежащие списку связанных) не рассматриваются и для них вектор не создаётся.

Для каждой сессии значения векторов, соответствующих действиям внутри неё, усредняются. Таким образом для каждой сессии мы получаем оценки вероятности того, что пользователь её совершивший придерживался интента, соответствующего кластеру. После чего усредняются результаты по всем сессиям. Сессии, в которых не содержалось ни одного действия, соответствующего пунктам 1 или 2, в усреднении не участвуют.

Полученный вектор согласно общей схеме алгоритма, описанной выше, содержит искомые оценки весов интентов.

Веб-инструмент

Для удобства проведения экспериментов, а также для дальнейшего использования аналитиками, на языке Java был разработан инструмент с веб-интерфейсом, доступный во внутренней сети компании «Яндекс».

Инструмент включает в себя форму для запуска расчёта оценок весов для запроса с указанием региона, при желании можно указать дополнительные настройки: способ построения набора связанных запросов, вероятностную модель, алгоритм кластеризации и его параметры (см. рис. 1).

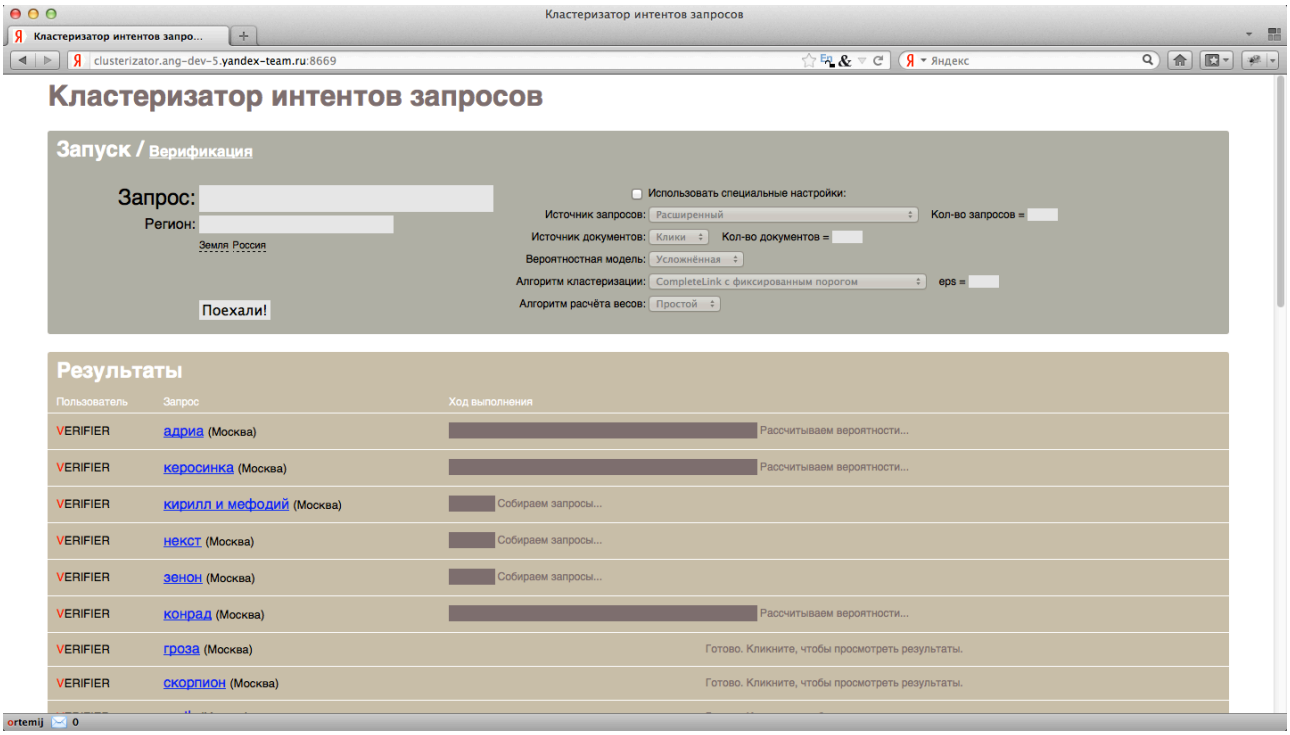

Рис. 1. Интерфейс веб-инструмента «Кластеризатор интентов запросов»

Внутри инструмента производится запуск описанного выше алгоритма расчёта весов, параллельно обрабатывается несколько запросов. Необходимые данные пользовательской статистики и сессии загружаются непосредственно с кластера MapReduce.

По всем когда-либо запущенным запросам можно посмотреть параметры запуска, результаты, сравнить их со значениями асессоров, если таковые имеются (см. рис. 2).

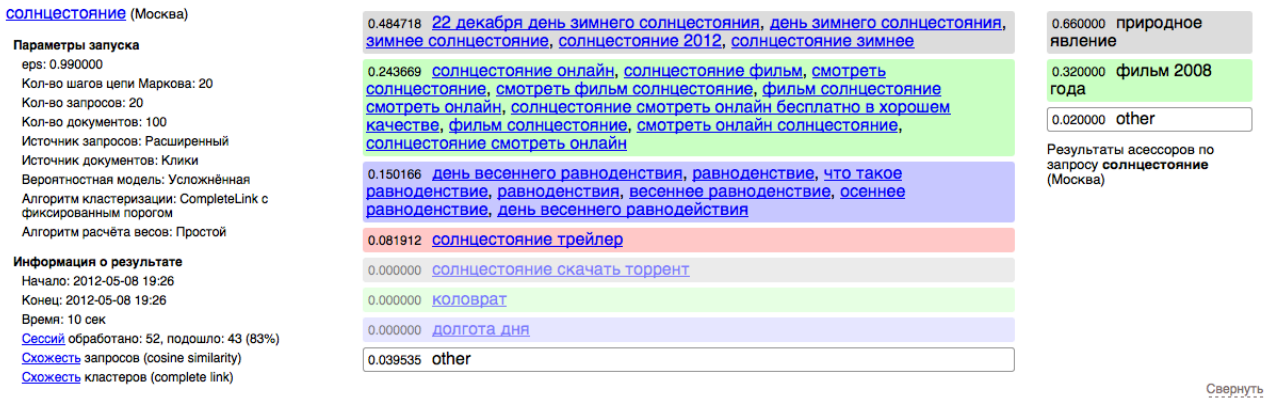

Рис. 2. Пример отображения результатов запуска

Инструмент позволяет посмотреть случайные сессии по которым рассчитывались веса, а также матрицу со значениями функций сходства запросов и полученных кластеров.

Результаты работы алгоритма

Анализ полноты и точности

При постановке задачи упоминалось о двух критериях, по которым будут проверяться результаты работы алгоритма.

Первый критерий – это полнота. Для каждого запроса из эталонного множества вручную производилась проверка, для какого количества интентов из известного набора присутствуют кластеры в наборе, полученном автоматически. В результате для 65% эталонных запросов был получен полный набор интентов, для 94% недоставало максимум одного.

Второй критерий – точность. Оценивалось максимальное различие между посчитанными асессорами весами и весами, полученными в результате работы алгоритма. В случае, если один интент из «ручного» набора был представлен несколькими в «автоматическом» (к примеру, интент «богиня» для запроса [афина] алгоритмом разделяется на более мелкие «богиня», «священное животное Афины» и «другие греческие богини»), значения весов для разделённых наборов суммировались.

Проверка точности вызвала трудности, связанные с тем, что данные о весах, полученные асессорами, устарели по сравнению с тем периодом пользовательской статистики, по которому производились расчёты. В ближайшее время планируется организовать такую проверку.

Однако, если использовать устаревшие данные о весах, максимальное различие, усреднённое по всем запросам с полным набором интентов, составляет 0.17.

Примеры обработанных результатов

В таблице 3 приведены примеры обработанных результатов работы алгоритма по нескольким запросам в сравнении с результатами, полученными после работы асессоров. Обработка заключалась в сопоставлении кластеру интента запроса.

Запрос **Регион**	**Алгоритм**		**Асессоры**	
	Вес	*Интент*	*Вес*	*Интент*
Солнцестояние (Москва)	0.48	Солнцестояние	0.66	Природное явление
	0.27	Фильм	0.32	Фильм
	0.10	Смотреть фильм онлайн	0.02	*Другое*
	0.09	Равноденствие		
	0.07	*Другое*		
Афина (Москва)	0.43	Богиня Афина	0.69	Богиня
	0.20	Священное животное Афины	0.07	Певица
	0.20	Другие греческие богини	0.07	Салон красоты
	0.10	Певица	0.17	Другое
	0.04	Салон красоты		
	0.03	*Другое*		
Понедельник (Москва)	0.19	Светотехническая компания	0.81	День недели
	0.10	Афоризмы про понедельник	0.17	Светотехническая компания
	0.09	Картинки про понедельник	0.02	*Другое*
	0.04	Виктор Понедельник		
	0.04	Понедельник по-английски		
	0.03	Понедельник начинается в субботу		
	0.53	*Другое*		
Наполеон (Земля)	0.73	Наполеон I Бонапарт	0.85	Наполеон I Бонапарт
	0.14	Торт	0.05	Игра “Napoleon: Total War”
	0.05	Фильм 2002 года	0.03	Торт
	0.03	Игра “Napoleon: Total War”	0.07	*Другое*
	0.01	Фильм 2011 года		
	0.04	*Другое*		
Вертикаль	0.57	Группа компаний	0.58	Группа компаний
	0.07	Понятие	0.08	Центр образования
	0.06	Магазин	0.03	Фильм
	0.05	Программа	0.01	Фестиваль туристических фильмов
	0.04	Роллы в Екатеринбурге	0.30	*Другое*
	0.02	Турфирма		
	0.02	Смотреть фильм онлайн		
	0.07	*Другое*		

Табл. 3. Примеры результатов работы алгоритма

Заключение

Был построен алгоритм, позволяющий получать наборы интентов для многозначных запросов и оценки их весов и показывающий лучшую полноту по сравнению с результатами асессоров на эталонном множестве запросов.

В завершении работы по проверке точности полученные данные будут использоваться для расчёта метрик в службе оценки качества поиска компании «Яндекс», обновления наборов запросов, по которым производится оценка качества поиска.

В ходе работы кроме самого алгоритма:

1) Был создан инструмент на языке Java, имеющий веб-интерфейс, удобный для использования аналитиками.
2) Написана утилита на языке C++ для расчёта статистики по переформулировкам запросов на кластере MapReduce.
3) Написаны скрипты для сбора по поисковым логам сессий пользователей, используемых в расчётах.

В дальнейшем предполагается работа в следующих направлениях:

1) Создание полуавтоматической системы проверки точности и полноты;
2) Фильтрация не относящихся к данному интенту данных;
3) Улучшение матчинга сессий с использованием лингвистических данных;
4) Исследования по определению интента из коротких, малоинформативных сессий;
5) Другие алгоритмы кластеризации и функции сходства.

Список литературы

[1] Оптимизация алгоритмов ранжирования методами машинного обучения, А. Гулин, П. Карпович, Д. Расковалов, И. Сегалович // РОМИП'09.

[2] Clustering Query Refinements by User Intent, E. Sadikov, J. Madhavan, Lu Wang, A. Halevy // WWW'10.

[3] Clustering User Queries of a Search Engine, Ji-Rong Wen, Jian-Yun Nie, Hong-Jiang Zhang // WWW'01.

[4] Diversifying Search Results, R. Agrawal, S. Gollapudi, A. Halverson, S. Ieong // WSDM'09.

[5] Inferring Query Intent from Reformulations and Clicks, F. Radlinski, M. Szummer, N. Craswell // WWW'10.

[6] Query Similarity by Projecting the Query-Flow Graph, I. Bordino, C. Castillo, D. Donato, A. Gionis // SIGIR'10.

[7] Random Walks on the Click Graph, N. Craswell, M. Szummer // SIGIR'07.

[8] The Query-flow Graph: Model and Applications, P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, S. Vigna // SIKM'08.