УДК 519.61

РАНЖИРОВАНИЕ ДОКУМЕНТОВ ПО ЗАПРОСУ НА ОСНОВЕ ЛОГА ДЕЙСТВИЙ ПОЛЬЗОВАТЕЛЕЙ ПОИСКОВОЙ СИСТЕМЫ

М. С. Агеев[1]

Предложен алгоритм улучшения качества ранжирования поисковой системы на основе предсказания релевантности документов запросу. Для предсказания релевантности используются методы машинного обучения и извлечения информации из логов. Высокая эффективность алгоритма продемонстрирована на реальных, полномасштабных данных поисковой системы. Алгоритм распараллеливается по технологии MapReduce, что позволяет обрабатывать логи и производить машинное обучение на кластерной архитектуре. Разработанная методика формирования факторов ранжирования может применяться для различных задач извлечения знаний из логов. Работа выполнена при финансовой поддержке РФФИ (проект 12–07–31225-мол_а).

Ключевые слова: поисковые системы, машинное обучение, анализ логов.

1. Введение. Поиск документов по запросу пользователя является ключевой технологией, обеспечивающей доступность информации в больших информационных массивах. Для больших коллекций документов и для поиска в сети Интернет наиболее важной характеристикой поисковой системы является качество ранжирования результатов запроса: ключевые, наиболее важные документы должны быть расположены в начале списка результатов.

Современные поисковые системы разбивают задачу построения функции ранжирования на две подзадачи [3, 7]. Первой задачей является разработка факторов ранжирования — множества функций, сопоставляющих паре запрос–документ действительное число, описывающее некоторую характеристику пары, связанную со степенью релевантности документа запросу. Факторами ранжирования могут быть, например, соответствие текста документа запросу на основе совместной частоты встречаемости слов по формуле tf*idf [31], длина запроса, частота цитируемости сайта по формуле PageRank [7]. Второй задачей является формирование функции ранжирования, которая на основе факторов ранжирования вычисляет степень соответствия документа запросу. Для решения второй задачи часто применяются методы машинного обучения ранжированию [30], оптимизирующие функцию качества ранжирования, вычисленную на основе созданной экспертами коллекции оценок релевантности документов запросу.

Одним из наиболее интенсивно развивающихся направлений исследований методов поиска документов по запросу является улучшение качества поиска на основе информации о поведении пользователей поисковой системы [4]. Данная информация накапливается в логах поисковых систем и содержит данные обо всех выполненных пользователями запросах, переходах по ссылкам и посещенных документах. Для идентификации пользователей сети Интернет используются анонимные идентификаторы (cookie), привязанные к браузеру. На основе информации из логов формируются поведенческие факторы ранжирования, которые используются совместно с другими факторами для построения функции ранжирования. Использование информации из логов позволяет улучшать качество поиска, так как логи предоставляют уникальную, не зависимую от других факторов информацию о степени привлекательности документов для пользователей.

Очередным шагом к улучшению методов ранжирования документов на основе логов действий пользователей стал конкурс по разработке алгоритмов предсказания релевантности "Интернет-математика 2011" (`http://imat-relpred.yandex.ru`), проведенный компанией Яндекс в октябре–декабре 2011 г. Впервые исследователям была предоставлена большая (340 миллионов кликов), представительная коллекция логов действий пользователей интернет-поисковой системы [32], плюс большая (71 930 пар документ–запрос) коллекция экспертных оценок релевантности документов для подмножества запросов из логов. Задачей участников конкурса было разработать алгоритм, улучшающий ранжирование поисковой системы при помощи информации из логов. В конкурсе приняли участие 84 команды из разных стран мира, плюс 17 участников из компании Яндекс (вне конкурса).

[1] Научно-исследовательский вычислительный центр, Московский государственный университет им. М. В. Ломоносова, Ленинские горы, д. 1, стр. 4, 119992, Москва; ст. науч. сотр., e-mail: mageev@yandex.ru

Главным результатом данного исследования является новый алгоритм ранжирования документов по запросу на основе лога кликов. Особенностью нашего подхода является формирование эффективных факторов ранжирования с простым физическим смыслом — каждый из разработанных нами факторов основан на явно сформулированной гипотезе о связи действий пользователя и релевантности документа. Алгоритм показал высокое качество ранжирования. При этом, в отличие от других алгоритмов, занявших высокие места и опубликованных в [2, 12, 22], предлагаемый алгоритм основан на относительно небольшом числе факторов ранжирования (24 фактора) и представляется простым для реализации.

Настоящая статья организована следующим образом: в разделе 2 приведен обзор современного состояния проблемы, описаны методы анализа логов и методы машинного обучения ранжированию; в разделе 3 дана постановка задачи конкурса и методы оценки качества ранжирования документов по запросу; в разделе 4 приведено детальное описание алгоритма: перечень используемых факторов, методов машинного обучения, методика разработки алгоритма и методы распараллеливания вычислений; в разделе 5 описаны эксперименты по сравнению нескольких версий алгоритма и результаты конкурса.

2. Современное состояние проблемы. Задача улучшения ранжирования документов на основе данных, содержащихся в логах, разбивается на две части: выделение факторов ранжирования из логов и формирование функции ранжирования методами машинного обучения ранжированию.

2.1. Методы машинного обучения ранжированию. Методы машинного обучения ранжированию начали развиваться с появлением в рамках инициативы TREC (`http://trec.nist.gov`) доступных коллекций экспертных оценок релевантности документов запросам. Так, во втором конкурсе алгоритмов поиска документов по запросу TREC-2 (1992 г.) лучшие результаты показала система [19], основанная на подборе коэффициентов логистической регрессии на основе оценок TREC-1.

Однако сам термин Learning to Rank (машинное обучение ранжированию) появился значительно позже. В 2005 г. прошла первая конференция по методам машинного обучения ранжированию NIPS-2005 Workshop on Learning to Rank (`http://drona.csa.iisc.ernet.in/ shivani/Events/Ranking-NIPS-05/index.html`). Взрывной рост публикаций по этой теме приходится на 2005–2010 г.г., когда проблема машинного обучения ранжированию была сформулирована как важный частный случай задачи машинного обучения. В последние годы основной фокус исследований направлен на изучение свойств алгоритмов машинного обучения ранжированию на реальных полномасштабных данных.

Формально, пусть задана функция $\overline{x}(q,d) \in \mathbb{R}^n$, которая для пары документ–запрос вычисляет вектор факторов ранжирования. Пусть для каждого запроса $\{q_i\}$, $i = 1,\ldots,N$, даны экспертные оценки релевантности найденных документов $\big\{(q_i, d_{ij}, y_{ij})\big\}$, $y_{ij} \in (0,1)$. Функция ранжирования $f\big(\overline{x}(q,d), w\big) \in \mathbb{R}$ с вектором параметров w определяет степень релевантности документа запросу; документы упорядочиваются по убыванию функции ранжирования.

Функция качества поиска для заданного вектора параметров w вычисляет среднюю оценку качества ранжирования по всем запросам: $L(w) = \dfrac{1}{N}\sum\limits_i L\Big(\Big\{f\big(\overline{x}(q_i,d_{ij}),w\big),\ y_{ij}\Big\}\Big)$. Функция качества поиска зависит только от порядка следования документов в результатах запроса. Общепринятыми [1, 29] считаются такие функции качества поиска, как точность на уровне первых N документов P@N — количество релевантных документов среди первых N; взвешенная с учетом позиций точность DCG@N, NDCG@N, метрики ERR и AUC [29].

Задачей метода машинного обучения ранжированию является выбор класса функций $f(\overline{x}, w)$ и подбор оптимальных параметров w с целью максимизации функции качества ранжирования $L(w)$.

С точки зрения теории оптимизации, целевая функция $L(w)$ является разрывной многоэкстремальной многомерной функцией. Поэтому для решения задачи максимизации $L(w)$ предложено множество приближенных методов [30], которые учитывают структуру функции $L(w)$ и ищут приближенное решение одним из следующих способов:

a) pointwise approach [19, 23, 33]: решение задачи регрессии — поиска функции $f\big(\overline{x}, w\big)$, которая предсказывает экспертную оценку релевантности $\min\limits_w \sum\limits_{ij}\Big(f\big(\overline{x}(q_i,d_{ij}),w\big) - y_{ij}\Big)^2$ на обучающей выборке $\big\{(q_i, d_{ij}, y_{ij})\big\}$, $y_{ij} \in (0,1)$;

b) pairwise approach [13, 27, 35]: решение задачи классификации пар документов $(d_{ij}, d_{i,k})$ на два класса: документы, которые нужно ранжировать в порядке (j,k), и документы, которые нужно ранжировать в порядке (k,j);

c) listwise approach [14, 34, 37]: максимизация $L(w)$ путем замены близкой функцией $L(w) \approx \widetilde{L}(w)$, обладающей свойством гладкости и/или выпуклости.

Несмотря на наличие большого количества методов и подходов к задаче машинного обучения ранжированию, выбор лучшего алгоритма машинного обучения является сложной задачей. Качество работы алгоритма зависит от набора факторов, качества обучающей коллекции и ограничений на скорость работы алгоритма и масштабируемость. Многие публикации с описанием новых алгоритмов не позволяют воспроизвести результаты и оценить качество работы алгоритма на новых данных [10]. Поэтому выбор алгоритма из числа опубликованных или разработка нового алгоритма требует высокой квалификации и интуиции. В 2009 г. компания Яндекс провела конкурс методов машинного обучения ранжированию "Интернет-математика 2009" (`http://imat2009.yandex.ru`), в котором участникам предлагалось разработать алгоритм машинного обучения ранжированию, максимизирующий функцию качества ранжирования DCG на данных, включающих более 200 факторов ранжирования. Затем в 2010 г. аналогичный конкурс провела компания Yahoo! [18]. Публикации по результатам этих конкурсов [15, 18, 23, 33] указывают на то, что для задачи машинного обучения ранжированию наилучшее качество дают pointwise методы регрессии, основанные на бустинге деревьев решений. В рамках данного исследования мы используем результаты и метод машинного обучения ранжированию [33], представленный одним из лидеров конкурса.

2.2. Ранжирование на основе логов поисковой системы. Основная сложность использования логов поисковой системы для ранжирования документов состоит в том, что наличие или отсутствие клика на документ в SERP (Search Engine Result Page — стандартное сокращение термина "страница результатов запроса") не всегда связано с релевантностью/нерелевантностью документа в связи со следующими особенностями задачи [25].

1. Многозначность поисковых запросов: один и тот же текст запроса может быть связан с различными информационными потребностями пользователя. Соответственно, для разных пользователей один и тот же документ может иметь разную степень релевантности.

2. Изменение информационной потребности пользователя со временем. Зависимость кликов от ранее просмотренных пользователем документов.

3. Несоответствие сниппета документу: пользователь решает, кликать или нет на документ на основе просмотре сниппета документа в SERP. Информации из сниппета может быть недостаточно для правильной оценки релевантности документа.

4. Зависимость от позиции в SERP: документы, расположенные в начале SERP кликают чаще, вне зависимости от релевантности документа.

В одной из первых статей по использованию логов поисковой системы для улучшения ранжирования [27] (2002 г.) реализована следующая идея: пусть по некоторому запросу документ d_1 находится выше в списке результатов, чем документ d_2, но пользователь системы выбрал документ d_2, пропустив d_1. Тогда можно предположить, что, вероятно, d_2 более релевантен, чем d_1, и улучшить качество ранжирования системы, поменяв местами d_1 и d_2.

В часто цитируемой работе [9] (2006 г.) предложен метод улучшения ранжирования на основе логов, ставший стандартным подходом. Из логов выделяется множество факторов ранжирования, таких как доля кликов на документ среди показов, среднее время просмотра документа, доля кликов на предыдущий/следующий документ в SERP, и т.п. На основе этих факторов метод машинного обучения ранжированию RankNet [13] строит функцию ранжирования, по которой упорядочиваются документы.

В работах [8, 17, 20, 21] действия пользователя описаны в виде кликовой модели (click model) — вероятностной графической модели, параметры которой (вероятности переходов между состояниями) определяются на основе обучающей выборки. В статье [6] приведен обзор вероятностных моделей действий пользователей.

В рамках данной работы мы реализовали кликовую модель [8], предложенную нами в 2011 г., и разработали ряд новых факторов ранжирования. Кликовая модель используется как один из множества факторов ранжирования, подаваемых на вход алгоритма машинного обучения.

По итогам конкурса "Интернет-математика 2011" опубликованы работы некоторых участников конкурса [2, 12, 22, 25]. В работе лидера конкурса [12] используется сочетание простых факторов, пяти вариантов кликовых моделей и методов коллаборативной фильтрации (группирования похожих запросов и документов методами понижения размерности). В работе [22] используются 34 фактора ранжирования и необычная комбинация двух алгоритмов машинного обучения. В работе [2] используются 234 фактора ранжирования и алгоритм машинного обучения Random Forests.

В отличие от других работ участников конкурса, мы при разработке алгоритма сделали упор на формирование эффективных факторов с простым физическим смыслом — каждый из разработанных нами факторов основан на явно сформулированной гипотезе о связи действий пользователя и релевантности документа. В результате, предложенный нами алгоритм основан на небольшом количестве факторов и

при этом показывает высокое качество предсказания релевантности.

Анализ логов действий пользователей также применяется для других задач информационного поиска, таких как предсказание удовлетворенности пользователя [8], предсказание готовности к совершению покупки [24], оценка удобства интерфейса сайта [11] и выработка рекомендаций для переформулировки запроса [16]. Решение каждой из этих задач основано на схожих методах выделения полезной информации из логов, поэтому разработанный в рамках данного исследования алгоритм, набор факторов и используемая методика формирования факторов будут полезны при решении других задач, связанных с анализом поведения пользователя по логам действий.

3. Постановка задачи. На вход алгоритма подается набор данных, состоящий из лога действий пользователя поисковой системы, обучающего множества оценок релевантности пар документ–запрос и тестового множества запросов. В конкурсе "Интернет-математика 2011" данные были представлены в следующем формате (описание дается в соответствии со страницей описания конкурса `http://imat-relpred.yandex.ru/datasets`).

1. Файл лога действий пользователя Clicklog.txt — это набор строк, в котором каждая строка представляет одно из возможных пользовательских действий: запрос или клик.

```
Запрос:  SessionID TimePassed TypeOfAction QueryID RegionID ListOfURLs
Клик:    SessionID TimePassed TypeOfAction URLID
```

Принятые обозначения: SessionID — уникальный идентификатор пользовательской сессии; TimePassed — время, прошедшее с начала текущей сессии, в условных временны́х единицах. Количество миллисекунд в одной временно́й единице не разглашается; TypeOfAction — тип пользовательского действия. Это может быть либо запрос (Q), либо клик (C); QueryID — уникальный идентификатор запроса; RegionID — уникальный идентификатор страны, из которой задан запрос (может принимать четыре значения); URLID — уникальный идентификатор документа; ListOfURLs — список документов, упорядоченный слева направо, как на странице выдачи Яндекса (сверху вниз).

Пример фрагмента файла лога:

```
10989856    0       Q   10364965   2   671723   21839763   3840421   180513
10989856    103     C   21839763
10989856    955     Q   1009161    2   197515   197539     11        179526   5859272
10989856    960     C   197515
```

2. Файл обучающего множества оценок релевантности пар документ–запрос Trainq.txt состоит из строк, каждая из которых состоит из четырех столбцов: "QueryID RegionID URLID RelevanceLabel", где QueryID — уникальный идентификатор запроса; RegionID — уникальный идентификатор страны, из которой задан запрос; URLID — уникальный идентификатор документа; RelevanceLabel — бинарная оценка релевантности (0 или 1).

Пример фрагмента файла оценок:

```
1209161     2     5839294     1
1209161     2     1912415     1
1209161     2     1621201     1
1209161     2     1111        0
```

3. Файл тестового множества запросов Testq.txt состоит из строк, каждая строка состоит из двух столбцов: "QueryID RegionID". Пары (QueryID, RegionID) взяты из лога действий пользователя Clicklog.txt и не пересекаются с обучающим множеством запросов, заданных в Trainq.txt.

Результатом алгоритма является список документов для каждого из тестовых запросов, упорядоченный по убыванию оценки релевантности пары документ–запрос. Оценка качества алгоритма производится по метрике AUC (Area Under Curve) [29], которая представляет собой вероятность того, что данный алгоритм для случайно выбранного запроса и пары, состоящей из релевантного и нерелевантного документа, поставит релевантный документ выше в списке результатов запроса. Формально, пусть $Q = \{q_j\}$ — множество запросов; n_1^q и n_0^q — количество релевантных и нерелевантных документов в ответе системы на запрос q; $S_0^q = \sum_{i=1}^{n_1^q} r_i^q$ — сумма позиций релевантных документов в ответе системы на запрос q, r_i^q — позиция i-го релевантного документа.

Тогда мера качества ответа системы для отдельного запроса q и в среднем по всем запросам определяется как $\mathrm{AUC}(q) = \dfrac{S_0^q - n_0^q(n_0^q+1)/2}{n_0^q n_1^q}$, $\mathrm{AUC} = \dfrac{1}{|Q|}\sum\limits_{q\in Q}\mathrm{AUC}(q)$.

Можно заметить, что так как данные лога содержат ранжированный список документов для каждого запроса, то можно естественным способом определить базовый уровень качества ранжирования: простейший алгоритм ранжирует документы в том же порядке, что и система, логи которой используются для обучения. В рамках данной работы мы отталкивались от базового алгоритма и улучшили качество ранжирования по сравнению с базовой системой.

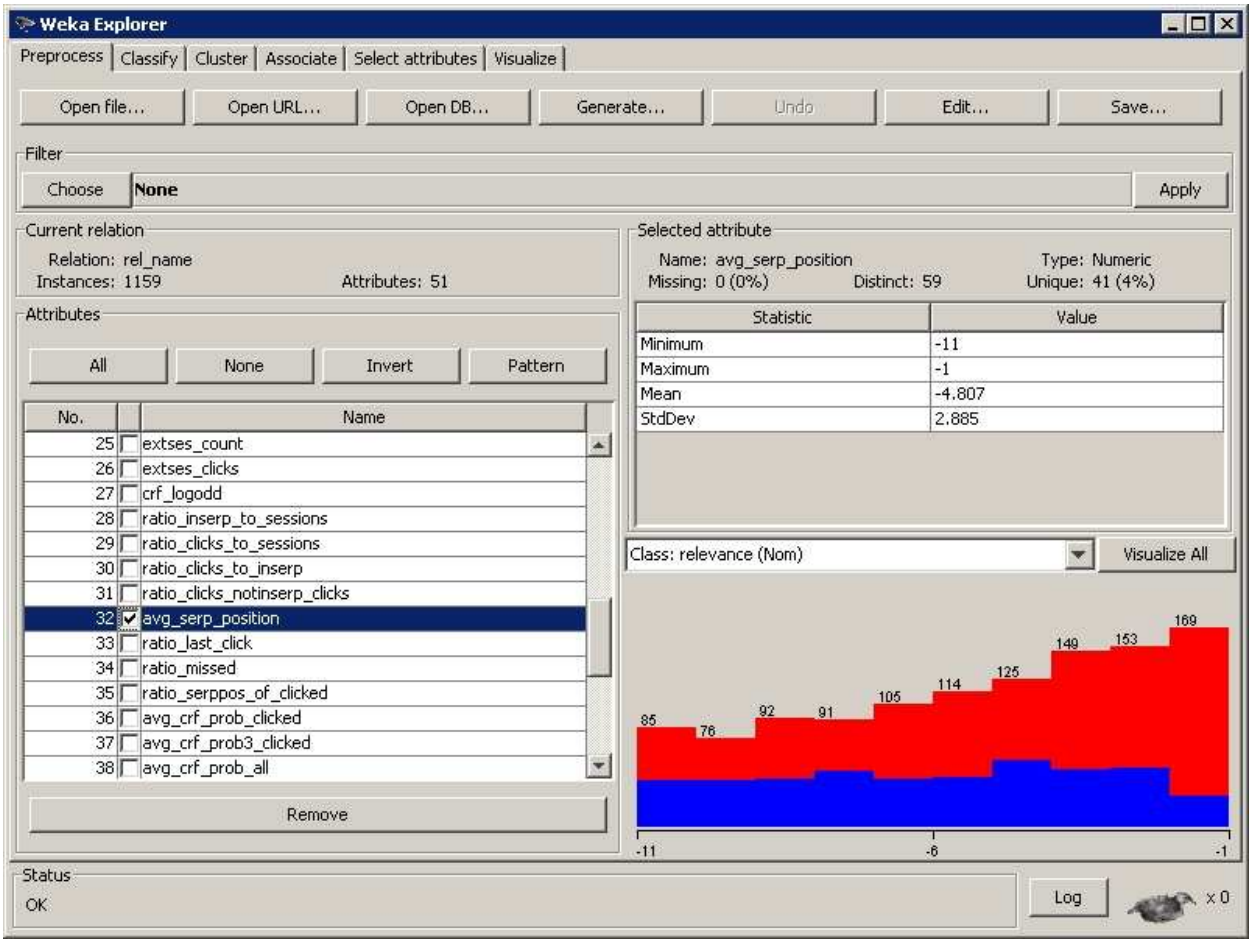

Рис. 1. Проверка качества факторов при помощи визуализации средствами Weka

4. Описание алгоритма. Мы используем стандартный [9] подход к задаче ранжирования документов на основе логов — задача разбивается на два этапа.

1. Входные логи преобразуются в пространство факторов $\overline{x}(q,d) : Q \times D \to \mathbb{R}^n$, каждый фактор сопоставляет паре документ–запрос число — значение фактора.

2. Для построения функции регрессии $f\big(\overline{x}(q,d), w\big) : \mathbb{R}^n \to \mathbb{R}$ применяется метод машинного обучения. Входными данными для машинного обучения является обучающая выборка оцененных пар документ–запрос, преобразованная в пространство факторов: $\Big\{\big(\overline{x}(q_i,d_{ij}), y_{ij}\big)\Big\}$. Функция регрессии f сопоставляет паре документ–запрос оценку релевантности документа запросу.

Для каждого запроса множество найденных документов упорядочивается по степени убывания предсказанной оценки релевантности $f\big(\overline{x}(q,d), w\big)$. Для достижения максимального качества ранжирования важным является каждый из этапов. Далее мы опишем 24 фактора ранжирования и два метода машинного обучения, которые использовались в данном исследовании. Факторы формировались по следующей методике. Каждый фактор отражает некоторое предположение о том, как данные логов могут отражать релевантность документа запросу. Желательно, чтобы значение фактора имело ясный физический смысл — это упрощает анализ влияния факторов на качество предсказания.

Желательно, чтобы предполагаемая зависимость релевантности от значения фактора имела монотонный характер, т.е. чем выше значение фактора, тем более была бы вероятна релевантность документа. Монотонные факторы упрощают задачу машинного обучения и таким образом способствуют повышению качества предсказания. Для проверки монотонности факторов использовался инструментарий WEKA Explorer [36] (`http://www.cs.waikato.ac.nz/ml/weka/`). На рис. 1 представлен график, на котором по оси абсцисс отложены значения фактора avg_serp_position, а по оси ординат — количество релевантных и нерелевантных пар документ–запрос. Из графика видно, что, во-первых, распределение доли релевантных документов почти монотонно возрастает с увеличением значения фактора (это хорошо!). Во-вторых, при avg_serp_position $= -11$ монотонность нарушается — когда документ находится за пределами страницы результатов, то он в среднем более релевантен, чем когда он находится в конце страницы результатов. Из этого следует, что имеет смысл выделить случай "документ находится за пределами страницы результатов" в отдельный фактор. На основании этого наблюдения построен фактор "ratio_clicks_notinserp_clicks".

4.1. Предобработка данных. В логе действий пользователей часто встречаются повторяющиеся действия: пользователь повторно задает один и тот же запрос или кликает на один и тот же документ, причем одинаковые клики идут последовательно в одной сессии. Мы предполагаем, что это вызвано задержками в получении ответа сети и неумелыми действиями пользователя, поэтому подряд идущие одинаковые клики нужно удалять из лога.

После удаления повторяющихся одинаковых кликов размер лога в байтах уменьшился на 4%, качество ранжирования по метрике AUC повысилось на 0.15%. Мы не использовали информацию о распределении запросов по регионам. Каждая пара запрос–регион рассматривалась как отдельный запрос. Идентификатором запроса являлась пара (QueryID, RegionID).

4.2. Группа факторов, основанных на числе кликов. Будем называть сессией последовательность действий пользователя в рамках ограниченного интервала времени. Сессия может состоять из одной или нескольких запросных сессий. Каждая запросная сессия состоит из одного запроса и последующих кликов на документы. Для каждого запроса определяем следующие значения:

— query_sessions — количество запросных сессий для данного запроса;

— query_abandoned_count — количество запросных сессий, в которых пользователь не кликнул ни одного документа в результатах запроса;

— query_sum_lowest_click_pos — сумма позиций самого нижнего клика по всем запросным сессиям данного запроса;

— query_sessions_with_one_click — количество запросных сессий, в которых был всего один клик в SERP (Search Engine Result Page — стандартное сокращение термина "страница результатов запроса").

Для пары документ–запрос вычисляем следующие значения:

— sessions_in_serp_count — количество запросных сессий, в которых данный документ был в SERP;

— sessions_seen — количество запросных сессий, в которых пользователь либо кликнул на данный документ, либо кликнул на документ, находящийся ниже в результатах запроса;

— click_count — количество кликов на данный документ по всем запросным сессиям данного запроса;

— click_inserp_count — количество кликов на данный документ при условии, что документ был в SERP;

— count_last_click — количество кликов на данный документ при условии, что документ был кликнут последним в запросной сессии;

— sessions_missed — количество запросных сессий, в которых пользователь не кликнул на данный документ, но при этом кликнул на документ, находящийся ниже в результатах запроса;

— sum_serp_position — сумма позиций документа в SERP по всем запросным сессиям для данного запроса;

— sum_serp_position_of_clicked — сумма позиций документа в SERP по всем запросным сессиям для данного запроса при условии, что документ был кликнут.

Опишем факторы, основанные на простых отношениях числа сессий и кликов. Для каждого фактора указано название, формула, описание фактора и предположение о связи фактора с релевантностью, на основании которого был предложен данный фактор. Число $\varepsilon = 10^{-12}$ — добавка к знаменателю для учета граничных случаев, когда числитель или знаменатель равны нулю.

Фактор 1: $\text{ratio\_inserp\_to\_sessions} = \dfrac{\text{sessions\_in\_serp\_count}}{\text{query\_sessions} + \varepsilon}$ — доля запросов, в которых данный документ попал на SERP (10 первых документов на странице результатов). Предположение: релевантные документы чаще попадают в выдачу.

Фактор 2: $\text{ratio\_clicks\_to\_sessions} = \dfrac{\text{click\_count}}{\text{query\_sessions} + \varepsilon}$ — отношение количества кликов на документ к количеству сессий, в которых был задан запрос. Предположение: пользователи чаще кликают на релевантные документы.

Фактор 3: $\text{ratio\_clicks\_to\_inserp} = \dfrac{\text{click\_inserp\_count}}{\text{sessions\_in\_serp\_count} + \varepsilon}$ — отношение количества кликов на документ к количеству сессий, в которых данный документ попал на SERP. Предположение: пользователи чаще кликают на релевантные документы.

Фактор 4: $\text{ratio\_clicks\_notinserp\_clicks} = \dfrac{\text{click\_count} - \text{click\_inserp\_count}}{\text{click\_count} + \varepsilon}$ — доля кликов на документ не с SERP. Предположение: если пользователь кликает на документ со второй страницы выдачи или по ссылкам, то это указывает на релевантность документа.

Фактор 5: $\text{avg\_serp\_position} = -\dfrac{\text{sum\_serp\_position} + 11\varepsilon}{\text{sessions\_in\_serp\_count} + \varepsilon}$ — средняя позиция документа в выдаче. В случае, когда документ не встречался в SERP, средняя позиция равна 11. Предположение: наверху SERP — более релевантные документы.

Фактор 6: $\text{ratio\_last\_click} = \dfrac{\text{count\_last\_click}}{\text{click\_count} + \varepsilon}$ — доля сессий, в которых данный документ был последним кликнутым документом. Предположение: пользователь останавливает свой поиск на релевантном документе.

Фактор 7: $\text{ratio\_missed} = -\dfrac{\text{sessions\_missed} + 1}{\text{sessions\_seen} + 1}$ — доля сессий, в которых пользователь не кликал на данный документ, но кликнул на документ, находящийся ниже в SERP. Предположение: пользователь пропускает нерелевантные документы.

Фактор 8: $\text{ratio\_serppos\_of\_clicked} = \dfrac{\text{sum\_serp\_position\_of\_clicked} + 11\varepsilon}{\text{click\_count} + \varepsilon}$ — средняя позиция документа в выдаче при условии, что пользователь кликнул на документ. В случае, когда нет кликов, средняя позиция равна 11. Предположение: наверху SERP более релевантные документы.

Фактор 9: $\text{ratio\_missed\_aband\_top2} = -\dfrac{\text{aband\_in\_top2}}{\text{query\_abandoned\_count} + \varepsilon}$ — доля сессий, в которых документ был в числе первых двух на SERP, но пользователь не кликнул ни одного документа в результатах запроса. Предположение: пользователь всегда просматривает первые две ссылки; если нет клика — значит, они не релевантны.

Фактор 10: $\text{ratio\_missed\_aband\_top3} = -\dfrac{\text{aband\_in\_top3}}{\text{query\_abandoned\_count} + \varepsilon}$ — доля сессий, в которых документ был в числе первых трех на SERP, но пользователь не кликнул ни одного документа в результатах запроса. Предположение: пользователь всегда просматривает первые три ссылки; если нет клика — значит, они не релевантны.

Фактор 11: $\text{ratio\_missed\_aband\_top5} = -\dfrac{\text{aband\_in\_top5}}{\text{query\_abandoned\_count} + \varepsilon}$ — доля сессий, в которых документ был в числе первых пяти на SERP, но пользователь не кликнул ни одного документа в результатах запроса. Предположение: пользователь всегда просматривает первые пять ссылок; если нет клика — значит, они не релевантны.

Фактор 12: $\text{ratio\_missed\_aband\_top10} = -\dfrac{\text{aband\_in\_top10}}{\text{query\_abandoned\_count} + \varepsilon}$ — доля сессий, в которых документ был на SERP, но пользователь не кликнул ни одного документа в результатах запроса. Предположение: пользователь всегда просматривает все ссылки на SERP; если нет клика — значит, они не релевантны.

4.3. Группа факторов CRF. Описанные выше факторы не учитывают временны́е интервалы между действиями пользователя и слабо учитывают порядок действий. Для моделирования последовательности действий воспользуемся широко известной моделью условно-случайных полей (Conditional Random Fields, CRF) [8, 28]. Модель CRF сопоставляет каждому клику пользователя случайную переменную — релевантность документа, на который кликнул пользователь. Модель основана на предположении, что релевантность клика y_i на шаге i сессии s зависит от свойств (факторов) x_i данного клика и от релевантности соседних кликов, но не зависит от свойств кликов, находящихся на расстоянии больше единицы: $P\big(y_i|s\big) = P\big(y_i|x_i, y_{i-1}, y_{i+1}, \lambda\big)$.

На рис. 2 показан пример вероятностной графической модели CRF для сессии, состоящей из двух запросов и трех кликов. Метки y_i являются скрытыми случайными переменными, факторы x_i — наблю-

даемыми переменными, вычисленными на основе данных лога. Параметры вероятностного распределения λ определяются EM-алгоритмом на этапе обучения CRF и используются для предсказания вероятности релевантности кликов для новых данных. На этапе обучения метка релевантности y_i принимает значения из множества: “релевантен”, “нерелевантен”, “не оценен”, “запрос”. Мы использовали свободно доступную реализацию CRF с открытым кодом Mallet (`http://mallet.cs.umass.edu`) и модифицировали класс SimpleTagger следующим образом:

— оптимизирована обработка больших файлов, уменьшены требования к объему оперативной памяти (исходная реализация Mallet потребовала бы для обработки данных конкурса ≈ 100 Gb памяти на этапе предсказания);

— для каждого клика рассчитывается вероятность метки (релевантности клика) $P\big(\mathrm{rel_d} = 1|\mathrm{s}, \lambda\big)$ для наблюдаемой сессии s и наиболее вероятная метка (исходная реализация Mallet выдает только наиболее вероятную метку);

— для каждой сессии рассчитывается наиболее правдоподобная последовательность меток.

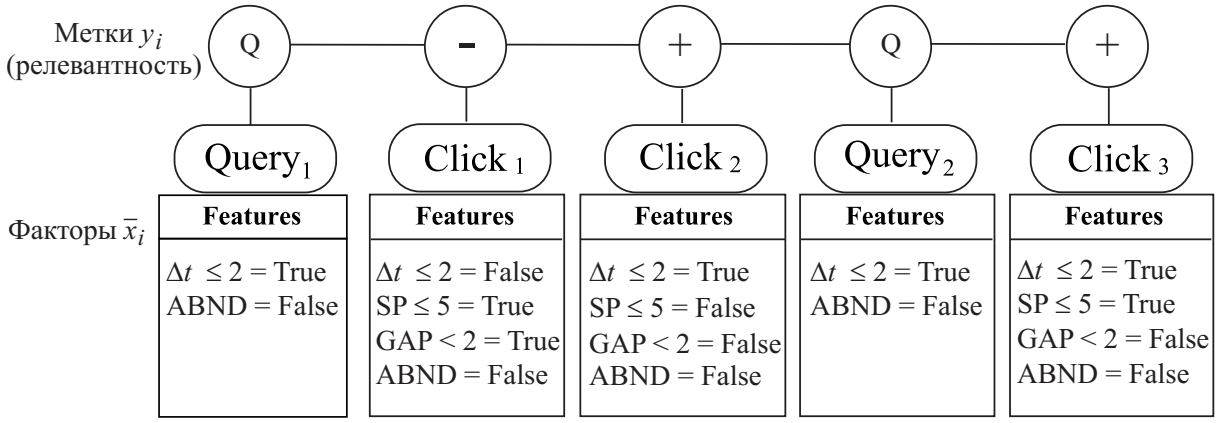

Рис. 2. Модель CRF для предсказания релевантности документов на основе последовательности действий

На вход CRF подаются следующие факторы:

1) type_of_action — тип клика: запрос или клик на документ;

2) $\Delta t \leqslant t_i$ — промежуток времени после предыдущего действия, булевские переменные для разных порогов t_i. В данных конкурса время представлено в условных единицах. Для того чтобы представить время в виде факторов, принимающих дискретные значения, мы построили гистограмму распределения времени первого клика после запроса и на основе гистограммы выбрали пороги: $t = 2,\ 5,\ 8,\ 20,\ 60,\ 200$;

3) ABND (query abandoned) — есть ли клики на документы после данного запроса;

4) $\mathrm{QCC} = 1$, $\mathrm{QCC} \leqslant 3$ — количество кликов на документы после данного запроса (query click count), два фактора для порогов 1 и 3;

5) browse_serp_pos $\leqslant 2$, browse_serp_pos $\leqslant 5$ — позиция документа на SERP, два фактора для порогов 2 и 5;

6) gap_from_previous_click_position $\leqslant 2$ — расстояние между позицией документа на SERP и позицией документа из предыдущего клика;

7) last_click_in_query — является ли данный клик последним в запросной сессии.

В результате применения CRF для каждого клика на документ определяется вероятность метки “релевантен”, “нерелевантен” и “не оценен” (вероятность метки “запрос” всегда равна нулю для кликов на документы). Обозначим эти вероятности $P\big(\mathrm{rel_d} = 1|\mathrm{s}\big)$, $P\big(\mathrm{rel_d} = 0|\mathrm{s}\big)$ и $P\big(\mathrm{rel_d} = \mathrm{NA}|\mathrm{s}\big)$ соответственно.

На основании данных CRF определим факторы для пары запрос–документ.

Фактор 13: $\mathrm{crf\_logodd} = \ln \prod\limits_{\mathrm{s}} \dfrac{\mathrm{P}\big(\mathrm{rel_d} = 1|\mathrm{s}\big)}{\mathrm{P}\big(\mathrm{rel_d} = 0|\mathrm{s}\big)}$ — логарифм отношения правдоподобия для оценки вероятности релевантности документа d, вычисленной CRF по всем сессиям s.

Фактор 14: $\mathrm{avg\_crf\_prob\_clicked} = \dfrac{1}{\mathrm{click\_count} + \varepsilon} \sum\limits_{\mathrm{s}} \dfrac{\mathrm{P}\big(\mathrm{rel_d} = 1|\mathrm{s}\big)}{\mathrm{P}\big(\mathrm{rel_d} = 1|\mathrm{s}\big) + \mathrm{P}\big(\mathrm{rel_d} = 0|\mathrm{s}\big)}$ — средняя оценка относительной вероятности релевантности документа d, вычисленной CRF по всем сессиям s, в которых был клик на данный документ. Вероятность релевантности вычисляется относительно вероятности того, что документ был оценен.

$$\text{Фактор 15: avg\_crf\_prob3\_clicked} = \frac{\sum_{\text{s}} \mathrm{P}\big(\mathrm{rel}_{\text{d}} = 1|\text{s}\big)}{\text{click\_count} + \varepsilon}$$ — средняя оценка относительной вероятности релевантности документа d, вычисленной CRF по всем сессиям s, в которых был клик на данный документ.

$$\text{Фактор 16: avg\_crf\_prob\_all} = \frac{1}{\text{query\_sessions} + \varepsilon} \sum_{\text{s}} \frac{\mathrm{P}\big(\mathrm{rel}_{\text{d}} = 1|\text{s}\big)}{\mathrm{P}\big(\mathrm{rel}_{\text{d}} = 1|\text{s}\big) + \mathrm{P}\big(\mathrm{rel}_{\text{d}} = 0|\text{s}\big)}$$ — средняя оценка относительной вероятности релевантности документа d, вычисленной CRF по всем сессиям s. Вероятность релевантности вычисляется относительно вероятности того, что документ был оценен.

$$\text{Фактор 17: avg\_crf\_rel1} = \frac{\text{count\_crf\_rel1}}{\text{click\_count} + \varepsilon}$$ — CRF для каждой сессии вычисляет наиболее правдоподобный набор меток. Фактор avg_crf_rel1 оценивает количество меток “релевантен” в наиболее правдоподобном пути, в среднем по всем сессиям, где был клик на документ.

4.4. Группа факторов, основанных на сессиях. Данная группа факторов не зависит от документа и характеризует разные свойства запроса.

$$\text{Фактор 18: qr\_clicks\_to\_sessions} = \frac{\text{query\_all\_clicks}}{\text{query\_sessions} + \varepsilon}$$ — среднее количество кликов в запросных сессиях данного запроса. Предположение: чем больше кликов, тем более сложный запрос, и тем меньше в результатах релевантных документов.

$$\text{Фактор 19: qr\_abandoned} = \frac{\text{query\_abandoned\_count}}{\text{query\_sessions} + \varepsilon}$$ — доля запросов без кликов среди всех запросов. Предположение: чем больше запросов без кликов, тем более сложный запрос, и тем меньше в результатах релевантных документов.

$$\text{Фактор 20: qr\_lowest\_click\_pos} = \frac{\text{query\_sum\_lowest\_click\_pos} + 11\varepsilon}{\text{query\_sessions} - \text{query\_abandoned\_count} + \varepsilon}$$ — среднее значение позиции самого нижнего клика в запросной сессии. Предположение: чем ниже последний клик, тем более сложный запрос, и тем меньше в результатах релевантных документов.

$$\text{Фактор 21: qr\_one\_click} = -\frac{\text{query\_sessions\_with\_one\_click}}{\text{query\_sessions} - \text{query\_abandoned\_count} + \varepsilon}$$ — доля запросных сессий, в которых был ровно один клик среди всех запросных сессий, в которых были клики. Предположение: запросы, в которых был ровно один клик, — простые, поэтому для таких запросов кликнутые документы более релевантны.

$$\text{Фактор 22: log\_query\_sessions} = \log(\text{query\_sessions} + 1)$$ — логарифм количества запросных сессий для данного запроса. Предположение: популярные запросы простые, и для них SERP содержит много релевантных документов.

Последние два фактора учитывают переформулировки запроса и соседние запросы в одной сессии. Напомним, сессией называется одна или несколько запросных сессий, исполненных пользователем в течение интервала времени.

$$\text{Фактор 23: extses\_ratio\_ses\_to\_extses} = \frac{\text{query\_sessions}}{\text{extses\_count} + \varepsilon}$$ — отношение количества запросных сессий к количеству сессий. Предположение: сложные запросы часто порождают множество переформулировок запросов и, как следствие, на каждый сложный запрос будет большее число сессий.

$$\text{Фактор 24: extses\_ratio\_clicks\_to\_sessions} = \frac{\text{extses\_clicks}}{\text{extses\_count} + \varepsilon}$$ — доля сессий, где был клик на данный документ, среди всех сессий для данного запроса.

4.5. Методы машинного обучения. Входными данными для машинного обучения является обучающая выборка оцененных пар документ–запрос, преобразованная в пространство факторов. Для каждой пары документ–запрос указаны 24 фактора ранжирования и метка релевантности $\{0, 1\}$. На основе обучающей выборки метод машинного обучения строит функцию $f\big(\overline{x}(q, d), w\big) : \mathbb{R}^{24} \to \mathbb{R}$, сопоставляющую набору факторов для пары документ–запрос оценку релевантности документа запросу. Набор найденных документов упорядочивается по степени убывания оценки релевантности. Задача машинного обучения функции релевантности называется задачей машинного обучения ранжированию (Learning to Rank).

Многие публикации [15, 18, 23, 33] указывают на то, что для задачи машинного обучения ранжированию наилучшее качество дают методы, основанные на бустинге деревьев решений. Поэтому в данном исследовании мы использовали два метода бустинга деревьев решений, для которых свободно доступны реализации.

1. AdaBoost [26] — один из методов бустинга, реализованных в пакете Weka [36]. Предварительные эксперименты показали, что алгоритм AdaBoost с подобранными параметрами показывает лучшее качество

по сравнению с другими алгоритмами пакета Weka.

2. Additive Groves — современный вариант алгоритма бустинга деревьев решений. Алгоритм Additive Groves [33] показал высокие результаты в конкурсе Yahoo! Learning to Rank Challenge 2010 — четвертое место в одной из двух дорожек конкурса. Это самое высокое место в конкурсе среди алгоритмов, для которых свободно доступна реализация.

Алгоритм Additive Groves [33] строит решающую функцию в виде суммы N компонентов, называемых Groves (роща). Каждый Grove строится независимо на случайном подмножестве обучающего множества (это называется bagging). Каждый Grove состоит из n деревьев решений, последовательно улучшающих регрессию (этот процесс называется boosting деревьев решений). Каждое дерево решений строится последовательно жадным алгоритмом с учетом ограничения на минимальное количество (долю) примеров обучающего множества, попадающего в лист дерева. Минимальная доля примеров в листе дерева задается параметром α алгоритма. Новизна алгоритма Additive Groves заключается в удачном выборе критериев (α, n), регулирующих степень обобщения обучающей выборки, и динамическом алгоритме подбора оптимальных параметров α и n по обучающей выборке. Свободно доступная реализация алгоритма Additive Groves (`http://aggitivegroves.net`) предусматривает возможность эффективного распараллеливания обучения на любом фреймворке параллельного программирования, поддерживающем MapReduce.

Для каждого из алгоритмов, основанных на деревьях решений, имеется параметр — количество итераций. При увеличении количества итераций качество предсказания монотонно увеличивается вплоть до некоторого предела ценой увеличения времени работы алгоритма. Количество итераций подбиралось экспериментально так, чтобы время работы алгоритма составляло не более нескольких часов.

Для алгоритма AdaBoost было выбрано 10000 итераций бустинга, остальные параметры оставлены по умолчанию. Параметры запуска Weka: “weka.classifiers.meta.AdaBoostM1 -opt “-P 100 -S 1 -I 10000 -W weka.classifiers.trees.DecisionStump”. Для алгоритма Additive Groves были выбраны параметры: количество итераций бэггинга: 2000; выбранные параметры: “-s slow -a 0.05 -n 3”.

4.6. Распараллеливание алгоритма. Каждый из этапов алгоритма — вычисление факторов и машинное обучение — эффективно распараллеливается на кластерной архитектуре по технологии параллельного программирования MapReduce [5]. Алгоритм реализован на языке Python с использованием библиотек Weka [36] и AdditiveGroves [33].

Для распараллеливания мы используем Hadoop (`http://hadoop.apache.org`) версии 0.20 и пакет Dumbo, позволяющий писать MapReduce программы на языке Python.

Алгоритм вычисления факторов состоит из следующих этапов.

1. Преобразование исходных данных лога (файла Clicklog.txt) в пары ключ–значение, где ключом является идентификатор сессии SessionID, а значением — фрагмент данных файла лога, относящийся к данной сессии. На этом же этапе производится предобработка — удаление повторяющихся кликов.

2. Вычисление факторов, основанных на числе кликов и сессиях. Производится за одну итерацию MapReduce:

Map: на вход подается фрагмент лога для одной сессии, на выходе — список пар ключ–значение, где ключ равен QueryID, а значение имеет структуру вида:

query_factors — список значений базовых факторов для запроса (примерами служат query_sessions, query_abandoned_count, query_sum_lowest_click_pos и др.), значения вычислены для данного запроса на фрагменте лога для данной сессии;

query_doc_factors — для каждого документа, найденного по запросу, — список базовых факторов (sessions_in_serp_count, sessions_seen, click_count и т.п.), вычисленных для данной пары запрос–документ на фрагменте лога для данной сессии.

Reduce: для каждого запроса — суммирование значений по каждому из факторов и вычисление факторов, используемых для машинного обучения. Результат — список пар ключ–значение, где ключ — это пара (QueryID, URL), а значение — это список пар (имя фактора, значение фактора), а также метка релевантности, проставленная экспертом.

Алгоритм машинного обучения AdditiveGroves распараллеливается на этапе обучения по итерациям бэггинга. Обучение производится за одну итерацию MapReduce. Пусть требуется построить b независимых Groves на N вычислительных узлах.

Map: на вход подается обучающая выборка. Программа ag_learn осуществляет построение b/N независимых Groves на обучающей выборке. Результатом является набор файлов, созданных программой ag_learn.

Reduce: программа ag_merge соединяет построенные Groves в одну модель.

На этапе предсказания алгоритм AdditiveGroves применяется без распараллеливания.

Эксперименты по обработке данных проводились на одном сервере с 24 ядрами процессора. Время работы программы составило: построение факторов — 6 минут с распараллеливанием на 20 процессов; обучение AdditiveGroves — 11 часов с распараллеливанием на 20 процессов, 99% времени заняла параллельная операция Map; предсказание алгоритмом AdditiveGroves — 1 минута.

5. Вычислительные эксперименты. Сравнение качества алгоритмов проводилось на данных конкурса "Интернет-математика 2011" по метрике AUC [29]. В рамках конкурса были доступны оценки на двух подмножествах данных:

— предварительная оценка алгоритма на подмножестве запросов public — доступна сразу после отправки результатов алгоритма. Каждый участник мог отправить результаты нескольких алгоритмов для получения оценки public;

— окончательная оценка алгоритма на подмножестве запросов test — оценка доступна только по окончании конкурса и только для одного алгоритма для каждого участника.

Таблица 1

Сравнение результатов различных алгоритмов по метрике AUC на коллекции public

Алгоритм	AUC (на коллекции public)
Additive Groves (все факторы)	0.6545
Additive Groves (все факторы, кроме группы CRF и 23-24)	0.6514
AdaBoost (все факторы, кроме группы CRF и 23-24)	0.6451
Baseline (количество кликов и средняя позиция SERP)	0.6358

В табл. 1 приведены результаты оценки на коллекции public для следующих алгоритмов:

1) Baseline — простейший алгоритм ранжирования документов. Документы упорядочиваются по убыванию количества кликов на документ и затем по возрастанию средней позиции документа в SERP;

2) AdaBoost — обучение функции ранжирования алгоритмом AdaBoost. Использовалось подмножество факторов ранжирования, состоящее из всех факторов, кроме группы факторов CRF и факторов 23 и 24 (свойств расширенных сессий);

3) Additive Groves на тех же факторах, что при использовании алгоритма машинного обучения AdaBoost;

4) Additive Groves на всех 24 факторах ранжирования.

Из табл. 1 видно, что алгоритм машинного обучения Additive Groves показывает существенно лучшие результаты, чем AdaBoost на том же множестве факторов.

Таблица 2

Сравнение результатов различных алгоритмов по метрике AUC на коллекции test

Алгоритм	AUC (на коллекции public)	Место (участники конкурса)
Cointegral (алгоритм не опубликован)	0.6673	—
Keinorhasen (команда из университета Гонконга [12])	0.6609	1
dminer (наш алгоритм, Additive Groves — все факторы)	0.6563	6
Медианный результат участников конкурса (команда id372)	0.6242	42

Интересным результатом является сравнение результатов нашего алгоритма с другими участниками конкурса. В табл. 2 представлены результаты участников конкурса на тестовой коллекции запросов. Наш алгоритм занял шестое место среди 84 участников конкурса и показал высокое качество ранжирования — 99.3% от лучшего опубликованного алгоритма по метрике AUC. Первое место среди всех участников (в том числе "вне конкурса") заняла команда Cointegral (Андрей Гулин, Яндекс). Первое место в конкурсе заняла команда Keinorhasen из Гонконга, описание их алгоритма опубликовано в статье [12]. Для сравнения в таблице также приведены результаты медианного (42-е место) участника конкурса.

6. Заключение. Мы представили новый алгоритм ранжирования документов по запросу на основе лога кликов. Особенностью нашего подхода является формирование эффективных факторов ранжиро-

вания с простым физическим смыслом — каждый из разработанных нами факторов основан на явно сформулированной гипотезе о связи действий пользователя и релевантности документа. В экспериментах на реальных данных поведения пользователей интернет-поисковой системы алгоритм показал высокое качество ранжирования. Алгоритм основан на относительно небольшом числе факторов ранжирования (24 фактора) и представляется довольно простым для реализации. Алгоритм позволяет параллельно обрабатывать логи и производить машинное обучение на кластерной архитектуре.

Мы планируем продолжить данное исследование и применить разработанный алгоритм, набор факторов и методику формирования факторов для решения других задач, связанных с анализом поведения пользователя по логам действий.

СПИСОК ЛИТЕРАТУРЫ

1. *Агеев М.С., Кураленок И.Е., Некрестьянов И.С.* Официальные метрики РОМИП 2010 // Тр. Росс. сем. по оценке методов информационного поиска. Казань: Казан. гос. ун-т., 2010. 182–187.
2. *Гуда С.А., Рябов Д.С.* Отчет по конкурсу Relevance Prediction Challenge (http://download.yandex.ru/company/SnD.pdf).
3. *Гулин А., Карпович П., Расковалов Д., Сегалович И.* Оптимизация алгоритмов ранжирования методами машинного обучения // Тр. Росс. сем. по оценке методов информационного поиска. СПб.: НУ ЦСИ, 2009. 163–168.
4. *Марманис Х., Бабенко Д.* Алгоритмы интеллектуального интернета. Передовые методики сбора, анализа и обработки данных. СПб.: Символ-плюс, 2011.
5. *Лэм Ч.* Hadoop в действии. М.: ДМК Пресс, 2012.
6. *Николенко С.И., Фишков А.А.* Обзор моделей поведения пользователей для задачи ранжирования результатов поиска // Тр. Санкт-Петербургского ин-та информатики и автоматизации РАН. Вып. 22. СПб.: СПИИРАН, 2012. 139–175.
7. *Маннинг К., Рагхаван П., Шютце Х.* Введение в информационный поиск. М.: Вильямс, 2011.
8. *Ageev M., Guo Q., Lagun D., Agichtein E.* Find it if you can: a game for modeling different types of web search success using interaction data // Proc. of the 34th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR). New York: ACM, 2011. 345–354.
9. *Agichtein E., Brill E., Dumais S.* Improving web search ranking by incorporating user behavior information // Proc. of the 29th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR). New York: ACM, 2006. 19–26.
10. *Armstrong T., Moffat A., Webber W., Zobel J.* Improvements that don't add up: ad-hoc retrieval results since 1998 // Proc. of the 18th Int. ACM Conf. on Information and Knowledge Management (CIKM). New York: ACM, 2009. 601–610.
11. *Atterer R., Wnuk M., Schmidt A.* Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction // Proc. of the 15th Int. Conf. on World Wide Web (WWW). New York: ACM, 2006. 203–212.
12. *Botao H., Liu N., Chen W.* Learning from click model and latent factor model for relevance prediction challenge // Proc. of Int. Conf. on Web Service and Data Mining workshop on Web Search Click Data. New York: ACM, 2012. 12–20.
13. *Burges C.J.C., Shaped T., Renshaw E., Lazier A., Deeds M., Hamilton N., Hullender G.* Learning to rank using gradient descent // Proc. of the 22nd Int. Conf. on Machine learning. New York: ACM, 2005. 89–96.
14. *Burges C.J.C., Ragno R., Le Q.* Learning to rank with nonsmooth cost functions // Proc. of Int. Conf. on Neural Information Processing Systems (NIPS) Cambridge: MIT Press, 2006. 395–402.
15. *Busa-Fekete R., Kegl B., Elteto T., Szarvas G.* Ranking by calibrated AdaBoost // J. of Machine Learning Research — Proc. Track 14. Brookline: Microtome Publishing, 2011. 37–48.
16. *Cao H., Jiang D., Pei J., He Q., Liao Z., Chen E., Li H.* Context-aware query suggestion by mining click-through and session data // Proc. of the 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD). New York: ACM, 2008. 875–883.
17. *Chapelle O., Zhang Y.* A dynamic bayesian network click model for web search ranking // Proc. of the 18th Int. Conf. on World Wide Web (WWW). New York: ACM, 2009. 1–10.
18. *Chapelle O., Zhang Y.* Yahoo! learning to rank challenge overview // J. of Machine Learning Research — Proc. Track 14. Brookline: Microtome Publishing, 2011. 1–24.
19. *Cooper W., Gey F., Dabney D.* Probabilistic retrieval based on staged logistic regression // Proc. of the 15th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR). New York: ACM, 1992. 198–210.
20. *Craswell N., Zoeter O., Taylor M., Ramsey B.* An experimental comparison of click position-bias models // Proc. of the Int. Conf. on Web Search and Web Data Mining (WSDM). New York: ACM, 2008. 87–94.
21. *Dupret G., Piwowarski B.* A user browsing model to predict search engine click data from past observations // Proc. of the 31st Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR). New York: ACM, 2008. 331–338.

22. *Figurnov M., Kirillov A.* Linear combination of random forests for the Relevance Prediction Challenge // Proc. of Int. Conf. on Web Service and Data Mining workshop on Web Search Click Data. New York: ACM, 2012. 71–75.
23. *Gulin A., Kuralenok I., Pavlov D.* Winning the transfer learning track of Yahoo!'s learning to rank challenge with yetirank // J. of Machine Learning Research — Proc. Track 14. Brookline: Microtome Publishing, 2011. 63–76.
24. *Guo Q., Agichtein E.* Ready to buy or just browsing?: detecting web searcher goals from interaction data // Proc. of the 33rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR). New York: ACM, 2010. 130–137.
25. *Guo Q., Lagun D., Savenkov D., Liu Q.* Improving relevance prediction by addressing biases and sparsity in Web search Click Data // Proc. of Int. Conf. on Web Service and Data Mining workshop on Web Search Click Data. New York: ACM, 2012. 71–75.
26. *Freund Y., Schapire R.* Experiments with a new boosting algorithm // Proc. of Int. Conf. on Machine Learning. San Francisco: Morgan Kaufmann, 1996. 148–156.
27. *Joachims T.* Optimizing search engines using clickthrough data // Proc. of ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining. New York: ACM, 2002. 133–142.
28. *Lafferty J., McCallum A., Pereira F.* Conditional random fields: probabilistic models for segmenting and labeling sequence data // Proc. of Int. Conf. on Machine Learning (ICML). New York: ACM, 2001. 282–289.
29. *Ling C.X., Huang J., Zhang H.* AUC: a statistically consistent and more discriminating measure than accuracy // Proc. of Int. Joint Conf. on Artificial Intelligence (IJCAI). **18**. Abington, UK: Lawrence Erlbaum Associates Ltd., 2003. 519–526.
30. *Liu T.-Y.* Learning to rank for information retrieval. Hanover, US: Now Publishers, 2009.
31. *Salton G., Buckley C.* Term-weighting approaches in automatic text retrieval // J. Inf. Processing and Management. 1988 **24**, N 5. 513–523.
32. *Serdyukov P., Craswell N., Dupret G.* WSCD 2012: workshop on web search click data 2012 // Proc. of Int. Conf. on Web Service and Data Mining. New York: ACM, 2012. 771–772.
33. *Sorokina D., Caruana R., Riedewald M.* Additive groves of regression trees // Proc. of European Conf. in Machine Learning. Berlin: Springer, 2007. 323–334.
34. *Taylor M., Guiver J., Robertson S., Minka T.* SoftRank: optimizing non-smooth rank metrics // Proc. of Int. Conf. on Web Search and Web Data Mining (WSDM). New York: ACM, 2008. 77–86.
35. *Tsai M.-F., Liu T.-Y., Qin T., Chen H.-H., Ma W.-Y.* FRank: a ranking method with fidelity loss // Proc. of the 30th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval. (SIGIR) New York: ACM, 2007. 383–390.
36. *Witten I., Frank E., Hall M.* Data mining: practical machine learning tools and Techniques. San Francisco: Morgan Kaufmann, 2011.
37. *Xu J., Liu T.-Y., Lu M., Li H., Ma W.-Y.* Directly optimizing evaluation measures in learning to rank // Proc. of the 31st Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR). New York: ACM, 2008. 107–114.

Поступила в редакцию
19.10.2012