Санкт-Петербургский государственный университет
Кафедра компьютерных технологий и систем

Мартынов Родион Сергеевич

Магистерская диссертация

Оптимизация поисковой выдачи с использованием алгоритмов онлайн-обучения

Направление 02.04.02
Фундаментальные информатика и информационные технологии
Магистерская программа:
Автоматизация научных исследований

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

Санкт-Петербург
2018

Содержание

Введение

Онлайн-обучение ранжированию - одна из наиболее развивающихся областей информационного поиска. В первую очередь это связано с обширным использованием различных поисковых систем, таких как Bing, Yahoo, Google и Yandex, которые служат для получения ранжированных наборов документов по некоторому семантическому запросу. Пользователь, ищущий информацию, отправляет запрос в систему, получает отранжированный список результатов поиска и переходит по наиболее подходящим ему ссылкам, исходя из заголовка и краткого содержания документа.

Ранжирующая функция (или функции) - неотъемлемая часть любой поисковой системы, с помощью которой определяется порядок выдачи документов по каждому конкретному запросу. До недавнего времени, большинство таких функций создавалось вручную экспертами в конкретной области поиска. Однако, такой подход, помимо его очевидной дороговизны и смещенности (качество ранжирования целиком и полностью определялось навыками и знаниями экспертной группы), стал неприменим в контексте огромного количества данных, которые на данный момент содержатся в сети Интернет. Объем информации, который необходимо обрабатывать, растет вместе с потребностями пользователей в как можно более удобном и быстром способе осуществлять поиск. Поэтому задача ранжирования поисковой выдачи стала одной из центральных в сфере информационного поиска.

Исходно к данной задаче применялись классические методы машинного обучения, в частности, обучения с учителем, т.е. алгоритмы, которые обучаются на некоторой выборке данных, содержащей сведения об оптимальном ранжировании и релевантности или не-релевантности документа запросу. Подобные выборки составляются экспертными группами, которые вручную размечают пары запрос-документ, что очень дорого и затратно по времени. К тому же, во многих областях, связанных с персонализированным или специализированным поиском получение необходимого объема размеченных данных практически невозможно - оценка релевантности может терять актуальность раньше, чем поисковые

системы получают шанс ее использовать.

Использование классической постановки задачи обучения затруднено также тем, что число признаков, используемых современными поисковыми системами при описании документа, может превышать несколько тысяч, что может быть проблемой для многих алгоритмов. Поэтому начали активно развиваться методы, способные обучаться ранжированию без использования размеченных данных и способные обрабатывать объекты большой размерности - методы *онлайн-обучения ранжированию.*

Вместе с такой постановкой проблемы возникает другая - неприменимость классических метрик качества из области машинного обучения с учителем, равно как и целевых функций, минимизируемых в задаче оптимизации. Поэтому алгоритмы онлайн-обучения ранжированию активно используют обратную связь, получаемую от пользователей поисковых систем в виде кликов и переходов по ссылкам. Однако необходимо принимать во внимание, что получаемая обратная связь зачастую является сильно зашумленной и смещенной относительно реальной релевантности документов запросу. Кроме того, она может зависеть от того, как именно представляются результаты поиска (фактор смещения из-за презентации). Поэтому методы обучения ранжированию должны учитывать подобные эффекты в процессе обучения.

Кроме того, большинство работ, относящихся к этой сфере, фокусируются на решении непосредственной задачи оптимизации ранжирования, без учета «удобства использования». Это затрудняет использование таких алгоритмов на практике, так как несмотря на теоретическую оптимальность полученной модели (или даже практическую, но оцененную на специальных наборах данных, без взаимодействия с пользователями), она в ходе поиска оптимума может предъявлять пользователю плохо отранжированные списки.

Данная работа представляет собой анализ существующих методов онлайн-обучения ранжированию и рассматривает способы их оптимизации с целью улучшения качества поисковой выдачи для абстрактной поисковой системы. Как итог, предлагается новый алгоритм оптимизации ранжирования, основанный на эволюционном алгоритме.

Постановка задачи

Формальная постановка задачи может быть представлена следующим образом: пусть существует набор документов D и набор запросов Q. Пусть D_q есть множество документов, выданное по запросу q, $q \in Q$, $D_q \subset D$. Рассматриваемое пространство объектов обозначим X. Его элементами являются пары запрос-документ $x = (q, d)$, $q \in Q$, $d \in D_q$. Пусть для каждой такой пары существует оценка релевантности документа запросу: $(q, d) \to r$, в общем случае нам неизвестная и для каждого запроса q задан *верный порядок* вида:

$$(q, d) \prec (q, d') \Leftrightarrow r(q, d) < r(q, d') \tag{1}$$

Требуется построить функцию ранжирования $F : X \to \mathbb{R}$, $F = F(q, d, \omega)$, аппроксимирующую порядок (1). Здесь ω - набор настраиваемых параметров функции. В эту задачу также включается задача построения алгоритма по подбору параметров ω исходя из обратной связи с пользователем в виде кликов c, т.е. построение такого $A(c, q, d, \omega) \to F(q, d, \omega)$.

Как видно из определения A, такой алгоритм не может использовать в качестве своих входных значений истинные оценки релевантности r, заменяя их исключительно использованием кликов от пользователей. Это влечет за собой следующую проблему: несмотря на то, что используемый алгоритм A может иметь хорошую асимптотическую оценку, т.е. производить функцию $F(q, d, \omega)$, наилучшим образом аппроксимирующую верный порядок документов для запросов, его время сходимости может быть слишком велико, что приведет к тому, что пользователь будет взаимодействовать с плохо отранжированной выдачей. Это может привести к тому, что пользователь разочаруется в поисковой системе и может перестать ей пользоваться. Для того, чтобы учесть данный фактор, оценка построенного алгоритма A производится в двух режимах:

- офлайн — оценка текущего наилучшего решения $F(q, d, \omega)$;
- онлайн — оценка ранжированного списка, непосредственно предъявляемого пользователю;

Для получения офлайн-оценки качества в данной работе используется показатель *normalized discounted cumulative gain* ($nDCG$) [1]:

$$nDCG@k = \sum_{i=1}^{k} \frac{2^{rel(r[i])-1}}{\log_2(i+1)} iDCG^{-1}$$

при длине оцениваемого списка $k = 10$. Данная метрика вычисляет прирост релевантности $rel(r[i])$ в зависимости от позиции i документа в ранжированном списке r, который затем нормируется максимальным возможным для данного запроса и множества документов приростом $iDCG$.

При оценке качества в онлайн-режиме используется *накопленный* $nDCG$ за все время взаимодействия пользователей с поисковой системой

$$nDCG_{online} = \sum_{t=1}^{T} \gamma^{t-1} nDCG_t$$

Здесь T - общее число запросов, $nDCG_t$ - метрика $nDCG$ списка, предъявленного пользователю на шаге t, γ - штраф.

Подробнее используемый подход описан в разделе 2. Оценка алгоритма A будет производится с помощью этих двух метрик с приоритетом у онлайн-$nDCG$.

Данная работа состоит из трех основных частей:

1. подробное описание основного современного подхода к онлайн-обучению ранжированию;
2. описание предлагаемого алгоритма эволюционного обучения ранжированию;
3. описание численного эксперимента на открытых данных;

Первая глава посвящена рассмотрению моделей, основанных на постановке задачи многорукого бандита и описывает существующие подходы к смешиванию ранжированных списков, рассматривая их с позиции релевантных для информационного поиска свойств. Вторая глава предлагает теоретическое описание предлагаемого подхода к оптимизации ранжирования основанный на эволюционных алгоритмах и затем, в последней части, приводятся экспериментальные результаты на открытых данных, включая анализ метрик как для онлайн, так и для офлайн-режимов работы.

Обзор литературы

Обучение ранжированию - одна из классических задач науки о данных и, в частности, машинного обучения. Основное развитие она получил в области информационного поиска применительно к поисковым системам и ранжированию документов по релевантности.

Работой, открывшей доступ для применения методов машинного обучения к этой задаче, по праву может считаться [2], в которой была сформулирована вероятностная модель ранжирования в информационном поиске. В ней предлагается ранжировать документы исходя не из абсолютной релевантности документа, которая может меняться со временем и в общем случае неизвестна, а используя вероятность того, что данный документ релевантен конкретному запросу. В частности, обозначим документ как x и пусть в нашем распоряжении имеется некоторое его представление в векторной форме: $x = (x_1, \ldots, x_n)$. Рассмотрим для такого документа полную группу событий (w_0, w_1), где w_0 - документ релевантен запросу и w_1 - документ не релевантен. Тогда ранг документа можно вывести из соотношения:

$$\frac{P(w_0/x)}{P(w_1/x)} \tag{2}$$

Этот результат важен в первую очередь тем, что позволяет использовать теоретическую базу из теории вероятностей и байесовской статистики для вычисления указанных условных вероятностей либо использовать методы из машинного обучения для их качественной аппроксимации.

Традиционно, задача поиска ранга в виде 2 решалась с использованием офлайн-алгоритмов. Одной из первых и наиболее важных работ в данной области можно считать [3]. В ней N. Fuhr определил семейство полиномиальных функций, из которого при наличии достаточного количества данных с известными оценками релевантности возможно было вывести ранжирующую функцию с помощью метода наименьших квадратов как решение задачи полиномиальной регрессии, обобщив модель 2 на не-бинарные оценки релевантности.

Подход к задаче, который был использован в [3], затем использовался

в множестве алгоритмов офлайн-обучения ранжированию. Наиболее влиятельными моделями стали: Ranking SVM - переложение классического метода опорных векторов для решения задачи ранжирования, RankBoost и LambdaMart - адаптация метода градиентного бустинга для оптимизации ранжирования. Разница между двумя алгоритмами в методе сравнения списков: RankBoost использует попарное сравнение документов на каждой позиции, в том время как LambdaRank, его более поздняя модификация, способна дополнительно оценивать качество ранжированного списка целиком (*listwise* подход).

Отдельно следует отметить работы [4] и [5], в которых рассматривается адаптация эволюционных алгоритмов к задаче обучения ранжированию. Авторы отмечают особо естественную применимость подобного подхода к задаче обучения ранжировнаию из-за специфики оптимизационных задач: большинство целевых функций и метрик (таких как $nDCG$ или число кликов) являются недифференцируемыми и применение классических методов теории оптимизации, основанных на явном вычислении градиентов, невозможно. А эволюционные алгоритмы являются очень сильным методом численной оптимизации, не использующими градиенты, что обуславливает их широкую распространенность в задачах, например, *обучения с подкреплением*.

Вместе с развитием офлайн-алгоритмов обучения ранжированию стали становиться очевидными их недостатки, а именно:

- необходимость большого количества данных с известными оценками релевантности
- сложность адаптации для конкретного пользователя
- смещенность в сторону данных, имеющихся в наличии (алгоритм, обученный офлайн, напрямую зависит от них)

В связи с этим стали активно развиваться подходы к онлайн-обучению ранжированию и извлечению информации из кликов.

Одним из первых алгоритмов в этой сфере стала онлайн-версия метода опорных векторов, предложенная Joahims [6], обучающаяся на информации о кликах пользователей (неявная обратная связь). В его статье были описаны проблемы, связанные с обучением через клики, такие как слабая коррелированность кликов на документ с его

реальной релевантностью данному запросу, неструктурированность и случайный характер кликов. Кроме того, им был предложен подход для оценки качества алгоритмов онлайн-обучения ранжированию, который заключается в следующем:

1. Для существующего набора данных, содержащего разметку пар запрос-документ по релевантности, строится функция, значение которой алгоритм должен оптимизировать (она не обязательно должна совпадать с *целевой* функцией алгоритма обучения). В работе Joahims за такую функцию был принят коэффициент ранговой корреляции Кендалла.

2. Для выбранной модели проводится *офлайн* эксперимент, соответствующий решению задачи обучения ранжированию с учителем. Такой шаг необходим для проверки того, что построенная модель действительно оптимизирует выбранную на предыдущем шаге функцию.

3. Если предыдущий шаг показал разумность выбора такой модели, проводится *онлайн* эксперимент, в котором вместо существующей разметки используются собранные каким либо образом (например, через поисковые логи) клики пользователей и оценка качества проводится по ним.

Данный подход с тех пор использовался в подавляющем большинстве исследований в области онлайн-обучения ранжированию.

Также в своей работе Joahims предложил метод для сравнения двух алгоритмов онлайн-ранжирования, который позднее был назван методом сбалансированного смешивания. Его суть заключается в том, что для каждого алгоритма получается ранжированный список документов l_1 и l_2 соответственно, которые затем смешиваются попарно (более подробное объяснение метода будет приведено далее). Данный метод показал куда большую чувствительность к оценкам пользователей чем классический подход A / B тестирования.

Эта идея была развита в работе [7], где был предложен метод смешивания ранжированных списков похожий на алгоритм набора команд в некоторых спортивных состязаниях. Было экспериментально показано, что подобное смешивание результатов работы двух моделей приводит к

получению гораздо менее смещенной обратной связи от пользователя в виде кликов за счет уменьшения влияния, основанного на представлении результатов, и перехода от абсолютных метрик измерения релевантности к относительным.

Следующим значимым в контексте онлайн-обучения ранжированию работой была статья Joahims и Yue [8], в которой был сформулирован подход к онлайн-обучению ранжированию как решение задачи оптимизации методом *многорукого бандита*. Он использует идею смешивания результатов работы двух алгоритмов ранжирования, но уже не для оценивания качества, а для непосредственно процесса обучения.

Метод оперирует над пространством параметров ранжирующей функции Ω, которое, в свою очередь, задает семейство ранжирующих функций $F(\omega_i)$, $\omega_i \in \Omega$. Для каждой пары функций по результатам кликов после смешивания определяется «победитель» и осуществляется шаг градиентного спуска в пространстве параметров Ω в направлении вектора параметров «победившего» алгоритма. Подобный переход позволил напрямую использовать получаемые от пользователей клики в оптимизации на каждом шаге. Также в работе были сформулированы все необходимые ограничения на выбор семейства ранжирующих функций и базовые предположения о характере данных (как документов, так и кликов). Это был крайне значимый результат, что подтверждается тем фактом, что большинство последующих работ использовало эту формулировку как основу или же модифицировало предложенный в данной работе метод.

После этого большинство работ в области онлайн-обучения ранжированию можно отнести к двум областям: улучшение сравнения обучающихся алгоритмов и улучшение самого процесса обучения.

В первой области свое развитие исследования Yue и Joahims получило в работах K. Hoffman, M. de Rijke и A. Schuth. На замену попарному сравнению документов из ранжированных списков, которое использовала первоначальная постановка задачи многорукого бандита, был представлен метод сравнения ранжированных списков целиком [9] с учетом общего ранжирования. Такой подход показал более высокую чувствительность к кликам пользователей.

На его основе был представлен метод *вероятностного смешивания*

[10], для которого была доказана несмещенность в том смысле, что для случайного распределения кликов метод не предпочитает одно ранжирование другому. Однако, за счет построения всех возможных вариантов размещения документов из двух ранжированных списков в списке, предъявляемом пользователю, вероятностное смешивание могло получить финальное ранжирование очень непохожее ни на один из смешиваемых вариантов, что ухудшало качество списка, непосредственно выдаваемого пользователю.

Эта проблема была подробно рассмотрена в [11], где был предложен алгоритм смешивания, основанный на решении задачи оптимизации при заданных ограничениях. В роли ограничений использовались допустимые ранжирования при условиях несмещенности (т. е. на первых позициях в финальном смешанном списке присутствует i документов из первого ранжирования и j документов из второго ранжирования при условии, что $|i - j| \leq 1$).

Следующим важным шагом в оценке ранжирований стало использование более чем двух «участников». Данный подход получил название *многоуровневого смешивания* (multileaving) и на данный момент является основным подходом к оценке ранжирований в онлайн-режиме. Первой работой, в которой на практике применялся такой подход, является [12]. В ней авторы предлагают методы многоуровненного смешивания, следующие принципам командного и оптимизированного смешивания. Также приводятся экспериментальные результаты, показывающие значительное увеличение чувствительности к кликам пользователя по сравнению со смешиванием только двух списков. Одной из последних работ стала [13], в которой к парадигме многоуровневого смешивания приводился метод вероятностного смешивания, однако, несмотря на теоретическую несмещенность, эксперименты не показали явного преимущества над командным многоуровневым смешиванием.

Подход многоуровневого смешивания затем позволил обобщить алгоритм обучения многорукого бандита с сравнения на каждом шаге двух алгоритмов на сравнение n алгоритмов. Подобный подход ускоряет сходимость и позволяет в некоторых случаях обойти проблему попадания в «проблемные» для градиентного спуска участки, за счет выбора сразу многих объектов в различных направлениях в пространстве

параметров Ω. В частности, в работе [14] показаны экспериментальные результаты обучения модели многорукого бандита с использованием многоуровневого командного смешивания и приведено сравнение с такой же моделью, использующей простое командное смешивание (т.е. только одного кандидата). Результаты показали серьезное преимущество в смысле скорости сходимости как офлайн так и онлайн, что было затем закреплено в работе [15], результаты которой считаются на данный момент лучшими в области.

Подобный набор экспериментальных работ в данной области дает серьезную основу для оценки новых алгоритмов как обучения ранжированию, так и его оценки. Однако, все приведенные выше работы делали основной целью оптимизацию офлайн-обучения, т.е. получение наилучшего возможного набора весов для основы выбора, учитывая онлайн-качество лишь в смысле скорости сходимости. Возможный компромисс между этими показателями стал объектом внимания совсем недавно. В частности, в работе [16] рассматривается отношение скорости обучения (онлайн) и качества полученного результата (офлайн). Авторы также предлагают новый алгоритм обучения ранжированию, основанный на метриках схожести документов. Данный алгоритм обеспечивает быструю сходимость, однако полученный оптимум показывает существенно меньшее офлайн-качество.

Также важными направлениями в последних работах, связанных с онлайн-ранжированием, является моделирование пользователя. Построение моделей поведения пользователя (кликовых моделей), учет дополнительных деталей, относящихся к пользователю и к поисковой системе [17], [18], [19].

В целом, подобная активность показывает актуальность данной темы в современном информационном поиске и дает почву для дальнейших исследований в этом направлении.

Глава 1. Существующие подходы

1.1 Алгоритм «многорукого бандита»

Перед тем, как рассматривать непосредственно алгоритмы онлайн-обучения ранжированию, необходимо описать вид функции ранжирования $F(q, d, \omega) = r(q, d)$, где $r(q, d)$ - ранг документа d в выдаче по запросу q. Здесь и далее рассматривается некоторое представление документов d в векторном пространстве D_q: $d = (d_1^q, d_2^q \dots d_n^q)$. Таким образом, каждый документ задан вектором своих признаков $d_i^q, i = \overline{1, n}$ относительно запроса q. Верхний индекс q у каждого признака означает то, что в общем случае такие признаки рассчитываются в зависимости от запроса. Например, таким признаком является значение функции $BM25$ [20]:

$$BM25(q, d) = \sum_{i=1}^{n} IDF(q_i) \frac{f(q_i, d)(k_1 + 1)}{f(q_i, d) + k_1(1 - b + b\frac{|d|}{avgdl})}$$

В этой формуле $q = (q_1 \dots q_n)$, q_i - слова запроса, $avgdl$ - среднее число слов в документе, k_1 и b - гиперпараметры. $IDF(q_i)$ есть обратная документная частота слова q_i:

$$IDF(q_i) = log\frac{N}{n(q_i)}$$

где N - общее число документов в индексе, а $n(q_i)$ - число документов, в которых присутствует q_i.

Функция $F(q, d, \omega)$ далее везде будет полагаться линейной, то есть

$$F(q, d, \omega) = \sum_{i=1}^{n} \omega_i d_i^q$$

Это в первую очередь связано со скоростью обучения линейных моделей [16], а также с качественным интерпретированием результатов обучения: величина веса ω_i при признаке d_i можно оценивать как величину вклада соответствующего признака в релевантность документа.

Основным алгоритмом, решающим задачу обучения ранжированию (т.е. задачу поиска оптимального набора параметров ω) в онлайн-режиме является модель многорукого бандита основанная на градиентном спуске. Модели многоруких бандитов являются классическими в сфере обучения

с подкреплением, где они известны как качественные базисные подходы с контролем над соотношением «исследования» и «эксплуатации». Алгоритм многорукого бандита в постановке [8] можно увидеть в алгоритме 1.

Алгоритм 1 Многорукий бандит, основанный на градиентном спуске [8]

Input: α, δ, w_0^0

1: **for** $t = 1 \dots \infty$ **do**
2: $q_t = receive\_query(t)$ *// получение запроса от пользователя*
3: $u_t^1 = sample\_unit\_vector()$ *// вектор направления*
4: $\omega_t^1 = \omega_t^0 + \delta u_t^1$ *// решение-кандидат*
5: Compare ω_t^1 and ω_t^0 :
6: **if** ω_t^1 win **then**
7: $\omega_{t+1}^0 = \omega_t^0 + \alpha u_t^1$
8: **else**
9: $\omega_{t+1}^0 = \omega_t^0$

Его основная идея заключается в том, чтобы найти оптимальный набор параметров $\omega \in \Omega$, сведя задачу поиска к последовательности сравнений между парой ω_t^0 и ω_t^1. На каждом шаге t рассматривается текущий лучший результат ω_t^0 и кандидат ω_t^1, который выбирается шагом в случайном направлении от ω_t^0. Данный алгоритм зависит от двух основных гиперпараметров: δ и α, которые могут быть интерпретированы как величина шага «исследования» и «эксплуатации» соответственно. При использовании данного алгоритма используются следующие допущения:

- Пространство параметров $\Omega = \{\omega_i\}_{i=\overline{1,\infty}}$:
  - является компактным;
  - является выпуклым;
  - содержит нулевой элемент;
  - содержится в n-мерной сфере;
- Целевая функция $v : \Omega \to \mathbf{R}$ отражает «качество» каждой точки $\omega \in \Omega$, однако ее значения не доступны для прямого анализа. Следующие положения предполагаются выполненными по отношению к v:
  - v является хотя бы один раз дифференцируемой;
  - v является строго вогнутой;

– v - подчиняется условию Липшица, т.е.

$$|v(w_1) - v(w_2)| \leq L_v \|w_1 - w_2\|$$

Подобные условия гарантируют наличие единственного максимума функции v в Ω и позволяют воспользоваться для нахождения этого максимума методом *стохастического градиентного спуска*. Однако, поскольку истинные значения функции v и ее градиента неизвестны, используется их аппроксимация.

Данный алгоритм является главной оптимизационной моделью в онлайн-обучении ранжированию благодаря своей интерпретируемости, простоте и при этом устойчивым результатам. Основные его адаптации и улучшения происходили в части сравнения существующего решения и решения-кандидата (алгоритм 1, строка 5). Это связано в первую очередь с тем, что подобные методы сравнения давали также почву для развития способов оценки качества ранжирования исходя из кликов. Подробнее они будут рассмотрены в следующем параграфе.

1.2 Методы сравнения ранжирований

Методы сравнения ранжирований - неотъемлемая часть любого поискового движка, который использует обучение ранжированию. Основная отличительная черта таких методов заключается в том, что они основаны на взаимодействии пользователя с системой и используют для оценки данные, полученные в результате такого взаимодействия. Вне зависимости от используемого алгоритма обучения, оценка качества в онлайн-режиме необходима для того, чтобы принять решение об относительном качестве используемых ранжирующих алгоритмов (в случае алгоритма 1, двух: текущего решения и решения-кандидата).

Первыми подобными методами были оценки экспертами вручную и статистические подходы к исследованию пользователей. Однако, как было сказано ранее и подробно рассмотрено в [21], оценки экспертами содержат неустранимое смещение, связанное с тем, что оценки релевантности исходят от экспертов, а не от пользователей. Исследование пользователей, в свою очередь, не имеет необходимой обобщающей силы, так как число исследуемых зачастую недостаточно для основательных выводов о

поведении общей массы. С другой стороны, онлайн-оценивание и онлайн-эксперименты могут быть использованы для выделения закономерностей поведения пользователя при его взаимодействии с поисковой системой и, как следствие, давать надежную оценку тому или иному алгоритму ранжирования [6]. На сегодняшний день активно используются два основных метода сравнения в онлайн-режиме: A/B тестирование и смешивание ранжированных списков.

A/B тестирование - наиболее часто используемый метод сравнения алгоритмов ранжирования (и в целом моделей машинного обучения, взаимодействующих с пользователями). A/B тестирование позволяет сравнить два ранжирования, показывая первое группе пользователей A и второе группе пользователей B, причем эти группы не пересекаются. Затем по каждой группе собираются различные метрики (например, *показатель кликабельности* [22]), после чего используется набор статистических методов для определения победителя (t-тест, ранговые критерии и т.д.). Подробнее это описано в работе [23].

Главным преимуществом метода является простота подсчета метрик - они вычисляются независимо для каждой группы и за счет этого позволяют получить несмещенную оценку качества для сравниваемых моделей. Однако, для получения такой оценки требуется большое количество наблюдений, что может ухудшать восприятие пользователем поисковой системы, окажись он в «худшей» группе. Кроме того, результаты также быстро устаревают, заставляя проводить все новые A/B тесты для новых моделей, что приводит к тому, что система никогда не выходит из состояния тестирования. Для решения этих проблем были созданы методы смешивания ранжированных списков.

Эта группа методов берет начало в работе [24], в которой был предложен метод уменьшения количества взаимодействий с пользователем, требуемых для оценки, за счет того, что документы, отранжированные с помощью обеих сравниваемых моделей, смешивались в один список и в таком виде предъявлялись пользователю. Затем, после получения обратной связи от пользователя в виде кликов (или их отсутствия), сравнение моделей проводилось в зависимости от этой обратной связи. Подобный метод значительно уменьшает разброс в возможных оценках для каждой из сравниваемых моделей и за счет этого является куда более

чувствительным, чем A/B тестирование.

Конкретный метод смешивания всегда содержит два алгоритма: алгоритм непосредственно смешивания двух ранжированных списков и алгоритм получения оценки исходя из кликов. Наиболее влиятельными среди них стали:

Сбалансированное смешивание [24]. Метод смешивания списков: на первом шаге случайно выбирается один из двух ранжированных списков и первый документ из него добавляется в результирующий список. Затем первый документ, которого еще нет в результирующем списке, добавляется из второго. Так продолжается до тех пор, пока результирующий список не окажется полностью заполнен. Сравнение проводится по числу кликов, которое получил каждый изначальный список. Несмотря на простоту, данный метод стал одним из наиболее часто применимых в оценке ранжирования. Однако, затем было показано, что сбалансированное смешивание может выдавать смещенную оценку в том смысле, что если сравниваются два очень похожих ранжирования, то сравнение всегда будет в пользу одного, вне зависимости от кликов пользователей.

Командное смешивание [7]. Смещение, присутствующее в сбалансированном смешивании, было устранено в данном методе. Метод смешивания списков аналогичен методу выбора команд в некоторых спортивных состязаниях. Подробный процесс смешивания представлен в алгоритме 2.

На выходе алгоритм выдает результирующий список L и полученные «команды» для ранжированных списков $TeamA$ и $TeamB$. Чтобы после получения кликов от пользователя сравнить исходные ранжирования A и B, обозначим клики на документы списка $L = (l_1, l_2, \dots)$ как $c_1, c_2, \dots$. Тогда оценка качества h для каждого из ранжирований может быть получена как:

$$h_a = |\{c_j : l_j \in TeamA\}|$$
$$h_b = |\{c_j : l_j \in TeamB\}|$$

Тогда если $h_a > h_b$, то A выигрывает сравнение, если $h_b > h_a$, то B выигрывает сравнение, если же $h_a = h_b$, то ранжирования считаются одинаковыми *в контексте данного запроса*. Данный метод является наиболее часто применимым в настоящее время благодаря своей

Алгоритм 2 Командное смешивание (TDI) [7]

Input: Ранжированные списки $A = (a_1, a_2, \dots)$ и $B = (b_1, b_2, \dots)$

Init: $L \leftarrow ()$; $TeamA \leftarrow \varnothing$; $TeamB \leftarrow \varnothing$; $i \leftarrow 1$
while $A[i] = B[i]$ **do**
    $L \leftarrow L + A[i]$
    $i \leftarrow i + 1$
while $(\exists i : A[i] \notin L) \wedge (\exists j : B[j] \notin L)$ **do**
    if $(|TeamA| < |TeamB|)$
$\vee((|TeamA| = |TeamB|) \wedge (RandBit() = 1)$ **then**
        $k \leftarrow \min_i\{i : A[i] \notin L\}$
        $L \leftarrow L + A[k]$
        $TeamA \leftarrow TeamA \cup A[k]$
    else
        $k \leftarrow \min_i\{i : B[i] \notin L\}$
        $L \leftarrow L + B[k]$
        $TeamB \leftarrow TeamB \cup \{B[k]\}$

простоте и качеству получаемой оценки.

Вероятностное смешивание [10]. Метод, являющийся вероятностным обобщением командного смешивания. Вместо использования конкретных команд, алгоритм предполагает оценку на основе всех возможных сочетаний команд .которые могли появиться для данных ранжированных списков. Сравнение выигрывает ранжирование, которое имеет наибольшую вероятность получить большее число кликов. Основное преимущество данного метода заключается в том, что он предоставляет возможность оценивать ранжирования не только по документам, которые попали в результирующий список L.

Подробное сравнение данных алгоритмов смешивания представлено в нескольких работах: в [25] предлагается общая схема сравнения различных методов смешивания с учетом несмещенности оценок, соответствия релевантности реальных документов и способности сохранять эти свойства на маленьких объемах данных, в то время как [26] рассматривает методы смешивания с позиции лучшей приспособленности к персонализации и/или поисковым движкам общего назначения.

На основе этих методов позднее были предложены методы

многоуровнего смешивания, которые позволяют сравнение более чем двух ранжирований на каждой итерации. Первым таким методом стало *командное многоуровневое смешивание* (TDM) [14]. Данный метод будет расссмотрен здесь подробнее, так как дальнейшая работа во многом основана на нем.

Алгоритм 3 Многоуровневое командное смешивание (TDM) [14]

Input: Множество ранжированных списков R,

1: длина результирующего списка k
2: $L \leftarrow ()$
3: $\forall R_x \in R : T_x \leftarrow \emptyset$
4: **while** $|L| < k$ **do**
5: выбрать R_x таким образом, чтобы минимизировать $|T_x|$
6: $p \leftarrow 0$
7: **while** $R_x[p] \in L \wedge p < k - 1$ **do**
8: $p \leftarrow p + 1$
9: **if** $R_x[p] \notin L$ **then**
10: $L \leftarrow L + R_x[p]$
11: $T_x \leftarrow T_x \cup R_x[p]$
return L, T

Формальное описание метода можно увидеть в алгоритме 3. Он является обощением алгоритма 2 на любое число ранжирований для сравнения. Важно отметить, что в результирующий список длиной k попадает максимум k различных ранжирований. Это означает, что если число сравниваемых ранжирований $|R| > k$, то это влечет за собой тот факт, что не все сравниваемые ранжирования на каждой итерации будут использованы для предъявления результирующего списка пользователю.

Оценка лучшего алгоритма производится тем же способом, что и в простом командном смешивании: качество ранжирования определяется числом кликов по документам, которые он представил в результирующий список.

Вторым существенным методом стал алгоритм *многоуровневого вероятностного смешивания* [13]. В его основе лежит та же идея, что и в вероятностном смешивании, за исключением того, что он может оценивать ранжирования, которые не попали в результрующий лист, за счет оценки

вероятности получения клика на документ из этого ранжирования. На данный момент с помощью этого метода получен лучший результат в области онлайн-обучения ранжированию на общепринятом тестовом наборе данных [16], который будет использоваться как качественный ориентир в главе с экспериментальными результатами.

Для того, чтобы включить методы многоуровневого смешивания в классический алгоритм многорукого бандита, необходимо произвести некоторые изменения. В частности, поскольку при использовании многоуровневого смешивания несколько ранжирований могут получить одинаковое число кликов и при этом среди них не будет текущего лучшего решения, необходимо адаптировать шаг обновления (алгоритм 1, строка 5). Примем следующие обозначения:

- ω^t - текущее (на шаге t) лучшее решение;
- α - величина градиентного шага;
- b - множество выигравших сравнение на текущем шаге решений;
- u - вектора-направления, соответствующие каждому решению из b.

В такой формулировке были предложены следующие методы [14]:

- Победитель получает все (winner takes all, WTA). В данном подходе из множества b случайным образом выбирается один кандидат b^j и осуществляется обновление в его направлении:

$$w^{t+1} = w^t + \alpha u^j$$

  Его основной недостаток заключается в том, что все выигравшие кандидаты отбрасываются, кроме одного, выбранного случайным образом. Преимущество состоит в том, что обновление весов идет *строго* в сторону одного из участвующих в сравнении решений.

- Победитель «в среднем» (mean winner, MW). В данном методе обновление осуществляется в сторону усредненного по всем выигравшим сравнение кандидатам направлению:

$$w^{t+1} = w^t + \alpha \frac{1}{|u|} \sum_{b^j \in b} u^j$$

  В отличие от WTA алгоритма, здесь для обновления используются все лучшие ранжирования, однако шаг идет в направлении среднего

решения, несмотря на то, что оно не сравнивалось с текущим лучшим решением, и поэтому финальное решение может быть неоптимальным.

В целом, можно заметить, что все данные алгоритмы рассчитаны на оптимизацию в локальной области текущего решения. Такой подход, несмотря на его состоятельность, опирается на существенные предположения о характере оптимизируемой функции, в частности, на непрерывность и дифференцируемость [8]. Однако, в обучении ранжированию функции, отражающие качество ранжированного списка, почти всегда не являются гладкими и не непрерывны [27]. Это ведет к тому, что пользователю на этапе сравнения могут предъявляться неоптимальные ранжирования, несмотря на то, что текущее лучшее решение уже показывает хороший результат. Более того, из-за характера онлайн-обучения, этот процесс никогда не прекращается - даже несмотря на то, что алгоритм оптимизации сошелся в некоторую область, алгоритмы, основанные на методе многорукого бандита, продолжают предъявлять смешанные списки пользователю, что ухудшает общее качество поиска в системе.

В связи с этим стали проводится исследования в области оптимизации ранжирования с помощью методов, не требующих точных вычислений градиента или его аппроксимации [4], [28]. Однако, подобные подходы анализировались в офлайн режиме, где используемая метрика качества может быть вычислена напрямую в процессе обучения. В данной работе предлагается подход к онлайн-оптимизации ранжирования, основанный на *эволюционных алгоритмах* и делающий упор на улучшение предъявляемого пользователю ранжированного списка на каждом шаге.

Глава 2. Эволюционный алгоритм оптимизации

2.1 Мотивация и предпосылки

Как говорилось в предыдущей главе, существующие подходы к онлайн-обучению ранжированию основываются на сильных предположениях о характере оптимизируемой функции релевантности. В частности, ограничения, накладываемые на целевую функцию $v(\omega)$, выполнения которых требуют существующие алгоритмы, подразумевают ее непрерывность в пространстве параметров Ω, что не является верным для большинства метрик качества ранжирования. В связи с этим имеет смысл использовать алгоритмы численной оптимизации, не основанные на использовании градиента целевой функции. Кроме того, алгоритм оптимизации должен применяться в онлайн-режиме и основываться исключительно на обратной связи, полученной от пользователей, без возможности вычислять целевую функцию непосредственно. Также необходимо учитывать, что клики, предоставляемые пользователями, плохо коррелируют с абсолютным качеством ранжирования и их предпочтительно использовать как *относительные* оценки.

Помимо этого, во внимание принимается важный нюанс методов, основанных на алгоритме многорукого бандита: такие методы основываются на улучшении *текущего лучшего решения*. Исходя из предположения о непрерывности целевой функции, подобный подход использует семплирование в локальной окрестности текущего решения для получения векторов-кандидатов. Если же предположение о непрерывности не выполняется, то в ходе семплирования могут получиться вектора, не дающие близкого по качеству ранжирования к текущему решению, которые в ходе смешивания списков могут испортить поисковую выдачу.

Для решения указанных выше проблем предлагается метод эволюционной оптимизации, основанный на генетическом алгоритме. Данный вид оптимизационных алгоритмов осуществляет поиск оптимального решения, подражая процессу эволюции в живой природе [29]. Изначально формируется «популяция» возможных решений, после чего новые решения формируются путем модификации лучших существующих

особей популяции. Процесс продолжается на протяжении многих итераций, в ходе которых происходит улучшение не только одного решения, но и всей популяции в целом за счет применения генетических операторов. Подобный подход хорошо зарекомендовал себя в задачах, где явное решение не существует, или не может быть получено в ходе эксперимента.

Также, в отличие от классических алгоритмов оптимизации, целью генетического алгоритма является поиск достаточно хорошего решения в терминах текущей оценки, а не оптимизация (поиск лучшего возможного решения). Это непосредственно связано с максимизацией качества в онлайн-режиме [30].

2.2 Генетический алгоритм и генетические операторы

Перед тем как перейти непосредственно к предлагаемому подходу, следует рассмотреть подробно процесс оптимизации с помощью генетического алгоритма, возможные параметры, гиперпараметры и предположения, из которых он исходит. Общая терминология, которая будет использоваться в этой главе:

- Особь или индивид — одно решение-кандидат. В постановке задачи онлайн-обучения ранжированию и, конкретно, целевой функции $F(q, d, \omega)$ особью будет считаться n-мерный вектор из пространства Ω.
- Популяция P — множество решений-индивидов фиксированного размера.
- Генетический оператор — оператор, действующий отображающий некоторое подмножество популяции в множество из пространства Ω. Основные виды:
  - Оператор скрещивания: оператор вида $C(\omega_1, \omega_2) \rightarrow \Omega$. Применяется в качестве аналога скрещивания двух особей во время естественной эволюции.
  - Оператор мутации: $M(\omega) \rightarrow \Omega$. Применяется для индивидуального преобразования особи и используется для создания разнообразия особей внутри популяции. Использование данного оператора сильно увеличивает степень «исследования» алгоритма.

– Оператор селекции: $S(\omega_1 \dots \omega_m) \to \Omega$. Данный оператор служит для отбора лучших особей внутри популяции. Обычно применяется к части популяции размера $m \leq n$, выбирая лучшую особь внутри этой части.

- Функция выживаемости $f(\omega)$ — функция оценки качества конкретного решения-кандидата.

Итак, формально генетический алгоритм можно сформулировать следующим образом:

1. Сгенерировать изначальную популяцию.
2. Вычислить функцию выживаемости (целевую функцию) для каждого индивида в популяции.
3. Выбрать с помощью какого-либо оператора *селекции* лучших особей.
4. Применить оператор *скрещивания* к части популяции.
5. Применить оператор мутации к части популяции.
6. Если не выполняется критерий остановки, перейти к шагу два.

Покажем, как каждый из этих шагов может быть применен к задаче онлайн-обучения ранжированию.

Шаг 1. Инициализация. В методах, основанных на алгоритме многорукого бандита, изначальное решение ω_0^0 инициализировалось нулевым вектором [8], [10], [31]. Однако, решение-кандидат ω_0^1 на первом шаге получалось при помощи семплирования n-мерного шара единичного радиуса с центром в ω_0^0, что почти всегда приводило к замене на первом шаге оптимизации, поскольку нулевой вектор не дает никакого осмысленного ранжирования. Также и изначальное поколение в генетическом алгоритме, состоящее из нулевых векторов, не имеет особого смысла. Поэтому, для инициализации генетического алгоритма данная работа использует семплирование n-мерного шара единичного радиуса с центром в 0. Стоит также отметить, что использование подобной инициализации вместо полностью случайной дает эффект *регуляризации* решения, т.е. ограничения области поиска с целью сделать процесс оптимизации более устойчивым.

Шаг 2. Вычисление целевой функции. При онлайн-обучении ранжированию, целевая функция не может быть вычислена напрямую. Однако, использование методов сравнения ранжирований, описанных в параграфе 1.2, позволяет оценить качество конкретно взятой особи относительно других особей в популяции. При этом, учитывая то, что генетические алгоритмы в среднем работают лучше при больших размерах популяции, следует использовать один из методов многоуровнего смешивания. Наиболее применимыми и зарекомендовавшими себя методами из этой области являются метод многоуровневого командного смешивания и метод многоуровневого вероятностного смешивания.

Разница между ними относительно их применения в генетическом алгоритме будет проявляться в нескольких аспектах.

При использовании многоуровневого командного смешивания, значение функции выживаемости каждой особи есть число кликов, которое получили документы из этого ранжирования. Это означает, что ненулевая выживаемость возможна максимум у k-особей, где k — длина результирующего списка.

Кроме того, функция выживаемости в таком случае является целочисленной. Это влечет за собой тот факт, что зачастую в популяции будет присутствовать много особей с одинаковой выживаемостью. Однако, данный факт непосредственно связан с улучшением онлайн-качества следующим образом: идеальное качество в онлайн-режиме достигается в случае если пользователь получил в результирующем (смешанном) списке все релевантные документы.

Если исходить из предположения о том, что пользователь кликает только на релевантные документы (подробнее этот вопрос будет рассмотрен в параграфе 3.2), то тот факт, что у нескольких особей на каком-то шаге оказалась одинаковая выживаемость, значит, что общее качество смешанного списка выше, чем если бы в популяции был один явный лидер с максимальной выживаемостью.

При использовании вероятностного смешивания выживаемостью каждой отдельной особи будет являться вероятность получения клика на текущей итерации. Это значит, что вне зависимости от размеров популяции, каждая особь может иметь ненулевую выживаемость, даже если не участвовала в составлении результирующего списка. Однако, при

увеличении размера популяции средняя выживаемость будет падать, что может негативно сказаться на онлайн-качестве.

Также важной особенностью является то, что значение выживаемости напрямую привязано к текущей итерации (из-за того, что оно вычисляется по кликам на результирующий список, полученный на этой итерации). Следовательно, на каждой итерации, вне зависимости от используемого метода смешивания, необходимо обнулять значение выживаемости у каждой особи внутри популяции.

Шаг 3. Селекция. Данный шаг отвечает за «эксплутационную» часть алгоритма оптимизации, т.е. за то, как сильно текущее решение сдвигается в направлении лучшего полученного результата. Селекция есть процесс отбора особей из популяции, которые будут отобраны для создания популяции на следующем шаге. При этом одно из требований заключается в том, что популяция должна сохранять свой размер, поэтому выбранный метод селекции повторяется столько раз, сколько необходимо для получения такого же числа особей (т.е. результат селекции может включать в себя решения-дубликаты). Существует обширное количество методов (операторов S) осуществления селекции при использовании генетического алгоритма. В данной работе будут рассмотрены основные подходы и их применимость к задаче онлайн-обучению ранжированию.

- Селекция k-лучших — наиболее простой алгоритм селекции. Для формирования следующей популяции выбираются k-особей с максимальными значениями функции выживаемости. Преимуществом метода является его простота и интерпретируемость, однако в контексте онлайн-обучения ранжированию он вносит значительное смещение в процесс оптимизации. Смещение возникает из-за того, что характер оценок, получаемый во время обучения, является относительным а не абсолютным. В связи с этим, несмотря на то, что текущий результат имеет высокое значение выживаемости, на последующих запросах выбранные особи могут показать себя гораздо хуже.

- Пропорциональная селекция — классический метод, предложенный в одной из первых формулировок генетического алгоритма как метода оптимизации [32]. В данном методе вводится математическое ожидание

селекции индивида $E_s[\omega]$, т.е. ожидаемое число раз, которое индивид будет выбран. Оно вычисляется следующим образом (здесь m — размер популяции):

$$E_s[\omega] = \frac{f(\omega)}{\frac{1}{m}\sum_{i=1}^{m} f(\omega_i)}$$

После вычисления $E_s[\omega]$ для каждого индивида в популяции производится семплирование методом «рулетки»: каждой особи ставится в соответствие сектор круглого колеса рулетки, размер которого пропорционален выживаемости индивида. Колесо вращается m раз, и на каждом вращении стрелка рулеточного колеса указывает на некоторые сектор. Индивид, соответствующий этому сектору, отбирается в следующую популяцию.

Показано, что при достаточно большом размере выборки данный метод выбирает каждого индивида ω ожидаемое число раз $E[\omega]$. Однако, при использовании в онлайн-ранжировании следует учитывать, что в большинстве методов «эффективный» размер популяции (т.е. число особей с возможной ненулевой выживаемостью) ограничено размером результирующего списка. Поэтому данный метод селекции следует рассматривать разве что в паре с многоуровневым вероятностным смешиванием.

- Турнирная селекция [33]. Турниром размера k называется подмножество индивидов размером k из текущей популяции. Внутри этого подмножества индивиды упорядочиваются по убыванию значения выживаемости и каждому из них сопоставляется вероятность выигрыша в турнире, прямо пропорциональная значению выживаемости.

  Таким образом, турнирная селекция, будучи инструментом для поощрения «эксплуатации», может также вносить свою долю в «исследование» во время оптимизации, давая шанс быть выбранными индивидам с меньшей выживаемостью. При этом соотношение «исследования» и «эксплуатации» напрямую зависит от размера k турнира.

  Подобный метод селекции является одним из наиболее распространенных за счет простоты вычисления и меньшей

смещенности по сравнению с предыдущими методами.

Также в контексте селекции следует упомянуть прием под названием *элитизм*. Элитизм есть модификация метода селекции, при которой генетический алгоритм сохраняет некоторое количество лучших особей и добавляет их в популяцию на каждой итерации. Подобные индивиды могут уйти из популяции только если они были изменены генетическими операторами скрещивания или мутации. Многие исследователи отмечают, что элитизм значительно улучшает качество работы генетического алгоритма.

Шаг 4. Скрещивание. Скрещивание (C) есть операция получения нового индивида путем совершения преобразования над двумя другими индивидами. Оператору скрещивания в генетическом алгоритме также соответствует гиперпараметр p_c - вероятность того, что две особи из популяции буду скрещены. Таким образом, оператор скрещивания применяется к каждой возможной паре в популяции с вероятностью p_c. Это один из инструментов, поощряющих «исследование» алгоритма путем внесения разнообразия в популяцию. Наиболее используемые виды оператора скрещивания приведены ниже.

- Одноточечное скрещивание. Пусть $p_1 = (p_1^1 \dots p_1^n)$ и $p_2 = (p_2^1 \dots p_2^n)$ - индивиды-«родители». Случайно выберем индекс j, $j \in (0, n)$. Тогда полученные в результате скрещивания индивиды будут выглядеть как $c_1 = (p_1^1 \dots p_1^j, p_2^{j+1}, \dots p_2^n)$ и $c_2 = (p_2^1 \dots p_2^j, p_1^{j+1}, \dots p_1^n)$. Процесс для $n = 5$ и $j = 2$ можно увидеть на рисунке 1.

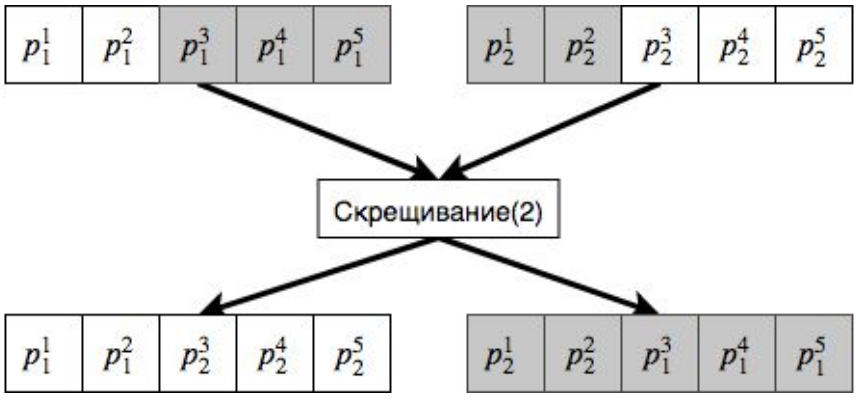

Рис. 1. Оператор одноточечного скрещивания

- Двухточечное скрещивание. Добавим к обозначениям, принятым для одноточечного скрещивания, второй случайно выбранный индекс

l, $l \in (0, n)$, $l \neq j$. Для простоты положим, что $j < l$. Тогда результатом применения оператора двухточечного скрещивания станут индивиды $c_1 = (p_1^1 \dots p_1^j, p_2^{j+1}, \dots p_2^l, p_1^{l+1}, \dots p_1^n)$ и $c_2 = (p_2^1 \dots p_2^j, p_1^{j+1}, \dots p_1^l, p_2^{l+1}, \dots p_2^n)$ (рисунок 2).

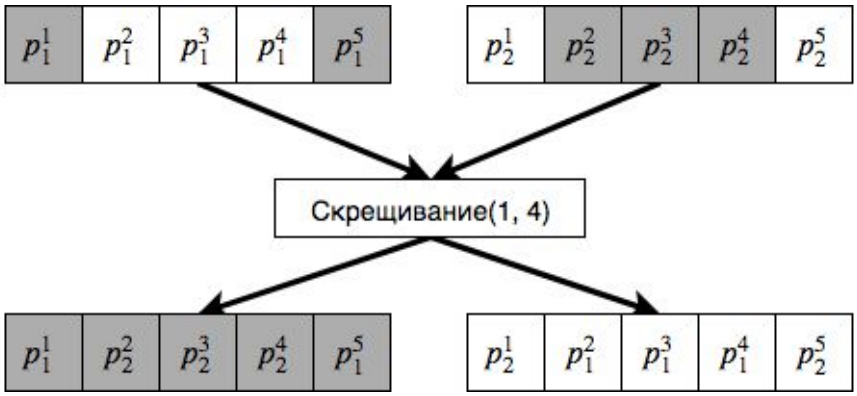

Рис. 2. Оператор двухточечного скрещивания

Оба метода широко применяются при использовании генетических алгоритмов. Двухточечное скрещивание в среднем создает больше вариаций внутри популяции, увеличивая степень «исследования».

Шаг 5. Мутация. Еще один генетический оператор (M), который, как и скрещивание, увеличивает степень «исследования» генетического алгоритма. В общем случае, разницу между ними можно сформулировать следующим образом: скрещивание является инструментом, обеспечивающим вариативность популяции в целом, тогда как мутация отвечает за то, чтобы алгоритм не фокусировался на отдельных компонентах векторов-особей. Как и в случае оператора скрещивания, оператору мутации соответствует гиперпараметр генетического алгоритма p_m - вероятность того, что конкретная особь подвергнется мутации.

Применительно к популяциям, в которых особями являются векторы вещественных чисел, наиболее часто используемыми операторами мутации являются равномерный оператор и оператор Гаусса. Пусть $\omega = (\omega_1 \dots \omega_n)$ — особь, подвергаемая мутации. Выберем случайным образом индекс i, $i \in (0, n]$. Тогда при использовании равномерного оператора появится особь, у которой на месте i-ой компоненты будет стоять значение случайной величины, равномерно распределенной в пределах заранее заданного интервала, а в случае оператора Гаусса — значение нормально

распределенной случайной величины с заранее заданным математическим ожиданием и дисперсией.

Шаг 6. Критерий остановки. Учитывая специфику постановки задачи онлайн-обучения ранжированию, алгоритм не прекращает работу до тех пор, пока существует поисковая система, выполняя одну итерацию при каждом запросе от пользователя.

На основании всего вышесказанного можно сформулировать следущий алгоритм *эволюционной оптимизации ранжирования*, основанной на генетическом алгоритме (дополнительный гиперпараметр k определяет длину смешанного списка, предъявляемого пользователю):

Алгоритм 4 Эволюционная оптимизация ранжирования (GARank)

Input: $P_1 \leftarrow ()$, $M(\omega)$, p_m, $C(\omega_i, \omega_j)$, p_c, $S(\omega_1 \dots \omega_m)$, k

for $i = 1 \dots m$ **do** *// начальная популяция*
  $P_1(i) = sample\_unit\_vector()$
for $t = 1 \dots \infty$ **do**
  $q_t = receive\_query(t)$ *// получение запроса от пользователя*
  $l = multileave(P_t, q_t, k)$ *// смешивание (см. алгоритм 5)*
  $c = infer\_clicks(l)$ *// получение кликов от пользователя*
  for $i = 1 \dots m$ **do** *// оценка решения (см. алгоритм 6)*
    $P_t(i).fitness = infer\_preference(l, P_t(i), c)$
  $O = S(P_t)$ *// селекция лучших*
  $P_{t+1} \leftarrow O$ *// новая популяция*
  for (ω_i, ω_j) in $O \times O$ **do** *// скрещивание*
    if $uniform\_random() < p_c$ and $i \neq j$ **then**
      $P_{t+1}(i), P_{t+1}(j) \leftarrow C(\omega_i, \omega_j)$
  for ω_i in O **do** *// мутация*
    if $uniform\_random() < p_m$ **then**
      $P_{t+1}(i) \leftarrow M(\omega)$
  for $i = 1 \dots m$ **do** *// обнуление выживаемости*
    $P_{t+1}(i).fitness = 0$

Таким образом, показано, что генетический алгоритм может быть применен вместо алгоритма многорукого бандита в задаче онлайн-обучения ранжированию. Учитывая специфику эволюционных

Алгоритм 5 Общая схема смешивания списков

function MULTILEAVE(P, q, k)

P — набор кандидатов

q — запрос

k — длина смешанного списка

$D_q \leftarrow get\_documents(q)$ *// извлечение документов по запросу*

$R \leftarrow ()$ *// все ранжирования, соответствующие P*

for ω_i in P **do**

$r_i = get\_ranking(D_q, \omega_i)$ *// ранжирование документов*

$R \leftarrow R + r_i$

$l \leftarrow ()$ *// финальный смешанный список*

Применение одного из методов многоуровневого смешивания из параграфа 1.2:

$l \leftarrow get\_multileaved(R, k)$

return l

Алгоритм 6 Вычисление выживаемости

function INFER_PREFERENCE(l, P, c)

l — список, предъявленный пользователю

P — оцениваемое решение

c — клики пользователя

Качество f вычисляется в зависимости от выбранного метода смешивания (см. параграф 1.2)

return f

алгоритмов, можно сформулировать следующие гипотезы, требующие эксперментальной проверки:

- Гипотеза 1: использование генетического алгоритма для онлайн-обучения ранжированию должно улучшить онлайн-качество ранжированию по сравнению с методами, основанными на алгоритме многорукого бандита, за счет большей степени «исследования» во время оптимизации и стремления улучшить подмножество решений нежели какое-то одно.
- Гипотеза 2: офлайн-качество генетического алгоритма может быть меньше в рамках выполнения компромисса между «исследованием»

и «эксплуатацией».

Данные гипотезы будут проверены экспериментальным путем с помощью моделирования взаимодействия пользователя с поисковой системой. Подробное описание эксперимента приведено в следующей главе.

Глава 3. Эксперимент

3.1 Описание данных

Для проверки работы предлагаемого алгоритма использовались открытые наборы данных Letor 3.0 и Letor 4.0 [34]. Это общеизвестные наборы данных, использованные во многих работах по оптимизации ранжирования и улучшении процедуры поиска. Letor 3.0 состоит из двух коллекций пар запрос-документы: «Gov» и «OHSUMED». Коллекция «Gov» отражает решение различных поисковых задач, поставленных на документах в доменной области .gov. Данные задачи были сформулированы на конференции по текстовому поиску (Text Retrieval Conference, TREC) в 2003 и 2004 годах. Три вида таких задач включены в Letor 3.0: *тематический поиск*, *поиск домашней страницы* и *поиск именованной страницы*.

Тематический поиск - это задача по поиску входных точек в какую-либо тематику. Основной упор делается на поиск главных страниц веб сайтов, хорошо отражающих тему поиска, нежели на более узкоспециализированные страницы, так как главные страницы сайтов могут дать более полный обзор всего источника данных.

Поиск домашней страницы направлен на быстрое нахождение главной страницы какого-то конкретного ресурса. Поиск именованной страницы решает задачу отыскания веб страниц, заголовок которых содержит запрос (в отличие от тематического поиска, где запрос предпочтительнее находить в теле страницы).

Вся коллекция документов разделена на части в соответствии с решаемыми задачами и годом, в который были собраны данные. Сводные данные приведены в таблице 1.

Коллекция OHSUMED представляет собой подчасть базы данных MEDLINE, которая содержит различные медицинские публикации. Коллекция состоит из 106 запросов, каждый из которых описывает поиск информации, связанной с медициной (либо на некоторую врачебную тематику, либо информация о симптомах конкретного пациента). Оценки релевантности документов по отношению к запросу были размечены с

Набор данных	Решаемая задача	Число запросов
HP2003	Поиск домашней страницы	150
HP2004	Поиск домашней страницы	75
NP2003	Поиск именованной страницы	150
NP2004	Поиск именованной страницы	75
TD2003	Тематический поиск	50
TD2004	Тематический поиск	75

Таблица 1. Характеристики Letor 3.0

Набор данных	Число запросов	Число документов
MQ2007	1692	69623
MQ2008	784	15211

Таблица 2. Характеристики Letor 4.0

помощью экспертов и могут принимать следующие значения:

- 2 - максимально релевантный запросу документ;
- 1 - частично релевантный запросу документ;
- 0 - не релевантный запросу документ;

Всего в данном наборе представлено 106 запросов и 16140 документов.

Letor 4.0 - обновленный набор данных для обучения ранжированию и оценки качества обучения. Он состоит из двух наборов запросов, взятых из направления Million Query track конференции по текстовому поиску и коллекции Gov2 документов из домена .gov. Данные запросы более не разделены на выполняемые поисковые задачи и представляют собой самые различные потребности пользователей поисковой системы внутри одного домена. Оценки релевнатности по своим обозначениям совпадают с обозначениями из набора OHSUMED. Весь набор поделен на две части, соответствующие направлению TREC 2007 и 2008 года. Численные характеристики находятся в таблице 2.

Внутри каждого набора данных множество запросов и документов представлено следующим образом: каждый запрос определяется уникальным идентификатором *qid*. Одному запросу соответствует коллекция документов, каждый из которых представлен вектором

Набор даных	Число признаков	Число уровней релевантности
HP2003	64	2
HP2004	64	2
TD2003	64	2
TD2004	64	2
NP2003	64	2
NP2004	64	2
MQ2007	46	3
MQ2008	46	3
OHSUMED	45	3

Таблица 3. Описание документов в различных наборах данных

различных признаков. Признаки включают в себя различные метрики релевантности (например, TF.IDF по отношению к запросу, BM25, HITS [35], PageRank [36] и так далее). Кроме того, для каждой пары запрос-документ задана метка релевантности. Эти метки не используются моделями онлайн-обучения ранжированию во время непосредственно процесса обучения, однако позволяют оценить сходимость алгоритма после того, как некоторое разумное число запросов было предоставлено алгоритму для обучения.

Сводные данные по каждому из наборов данных можно найти в таблице 3.

3.2 Моделирование пользователя

Для проведения эксперимента с использованием готовых наборов данных для ранжирования, необходимо рассмотреть вопрос о моделировании поведения пользователя. Основным способом моделирования являются *кликовые модели*. Кликовая модель — вероятностная модель, которая представляет поведение пользователя как серию наблюдаемых и скрытых событий. С её помощью можно оценить следующие вероятности: $P(C_i)$ — вероятность клика на i-ый документ и $P(S_i)$ — вероятность прекратить поиск на i-ом документе.

В данной работе используется каскадная кликовая модель, адаптированная для онлайн-обучения ранжированию [37], в которой

вышеуказанные вероятности моделируются как условные вероятности, зависящие от релевантности документа $R_i \in \overline{0,n}$: $P(C_i) = P(C_i = 1|R_i)$, $P(S_i) = P(S_i = 1|R_i)$.

Поведение пользователей, согласно этой модели, полагается следующим: изначально пользователь начинает просматривать ранжированный список документов, начиная с первого. Для каждого документа он решает, следует ли ознакомиться с ним более подробно, т. е. достоин ли он клика. После клика, пользователь выбирает: либо он нашел необходимую информацию и поиск следует прекратить, либо необходимо рассмотреть большее количество документов. Соответственно, варьируя указанные выше вероятности $P(C_i)$ и $P(S_i)$, можно задавать различные модели поведения пользователя.

В ходе эксперимента использовались три таких модели [19], [26]:

- Идеальная (per): пользователь кликает исключительно на самые релевантные документы и только на них.
- Навигационная (nav): пользователь в основном кликает на релевантные документы, однако ему требуется более обзорная информация по запросу.
- Информационная (inf): пользователь рассматривает некоторое подмножество документов, исходя в своем выборе не из релевентности, а из некоторого другого критерия, нам неизвестного.

Параметры каждой из моделей можно увидеть в таблице 4.

	$P(C_i = 1\|R)$			$P(S_i = 1\|R)$		
R	0	1	2	0	1	2
per	0,0	0,5	1,0	0,0	0,0	0,0
nav	0,05	0,5	0,95	0,2	0,5	0,9
inf	0,4	0,5	0,6	0,1	0,3	0,5

Таблица 4. Параметры кликовых моделей

3.3 Оценка качества

Основной метрикой качества данной работы является NDCG [1]. Данная матрика вычисляется следующим образом:

$$nDCG@k = \sum_{i=1}^{k} \frac{2^{rel(r[i])-1}}{\log_2(i+1)} iDCG^{-1}$$

при длине оцениваемого списка $k = 10$. Данная метрика вычисляет прирост релевантности $rel(r[i])$ в зависимости от позиции i документа в ранжированном списке r, который затем нормируется максимальным возможным для данного запроса и множества документов приростом $iDCG$.

Оценка качества работы проводилась в двух режимах - онлайн и офлайн. Для оценки качества в офлайн-режиме $NDCG@10$ вычисляется на отложенных тестовых запросах (такие по умолчанию присутствуют в каждом разбиении изначального набора данных в Letor 3.0 и Letor 4.0) и затем усредняется по всем тестовым запросам с использованием текущего лучшего решения.

Оценка качества в онлайн-режиме проводится с помощью вычисления накопленного $NDCG@10$ за все время, которое алгоритм продолжал обучение.

$$nDCG_{online} = \sum_{t=1}^{T} \gamma^{t-1} nDCG_t$$

Здесь T - общее число запросов, $nDCG_t$ - метрика $nDCG$ списка, предъявленного пользователю на шаге t, γ - штраф. В данной работе $T = 10000$ и $\gamma = 0.9995$. Значение γ выбрано таким образом, чтобы запросы за пределами 10000 оказывали влияние на оценку менее 1%.

Данная метрика может быть рассмотрена как аналог «награды», используемой в алгоритмах обучения с подкреплением, и призвана моделировать качество сходимости при стремлении числа запросов к бесконечности при оценке на фиксированном числе запросов.

Следует также заметить, что данные метрики были выбраны для того, чтобы иметь возможность полноценно сравнивать предлагаемый алгоритм с уже существующими лучшими решениями в области онлайн-обучения ранжированию. Подобный набор метрик и схема оценки качества

являются классическим при проведении экспериментов в данной области [16], [14], [15].

3.4 Гиперпараметры алгоритмов

В ходе эксперимента проводилось сравнение предлагаемого алгоритма эволюционного обучения ранжированию (алгоритм 4) с текущим лучшим результатом в области - PMGD [16] (probabilistic multileave gradient descent).

Для проведения сравнения алгоритм вычислялся на трех моделях пользователя - идеальной, навигационной и информационной. Каждый набор данных был разбит на 5 частей для скользящего контроля и внутри каждой части были выделены обучающая и тестовая выборка (для вычисления офлайн-качества). Алгоритм обучался на 10000 запросах от пользователей из каждой части. Всего было произведено 125 запусков для каждого набора данных, равномерно распределенные на 5 частей, и все результаты усреднены.

Подбор гиперпараметров для GARank осуществлялся при помощи предварительных экспериментов. Особо внимания заслуживает выбор метода смешивания ранжированых списков. Гипотеза о том, что метод многоуровневого вероятностного смешивания может предоставлять худшее качество в онлайн-режиме из-за того, что стремится выделить «явного лидера» внутри популяции, подтверждается (см. рисунок 3).

Как видно из графиков, метод вероятностного смешивания показал худшие результаты как в онлайн, так и в офлайн-режиме (на рисунке GARank-TDM соответствует применению многоуровневого командного смешивания, а GARank-PM - применению многоуровневого вероятностного смешивания). В соответствии с этим, в качестве метода смешивания в данной работе используется многоуровневое комнандное смешивание. Далее алгоритм онлайн-обучения ранжированию с применением многоуровневого командного смешивания будет обозначен как GARank-TDM.

Подобным образом подбирались значения всех остальных гиперпараметров. В целом, алгоритм показал низкую зависимость от конкретных значений вероятностей мутации и скрещивания p_m и p_c.

Также был использован элитизм для улучшения процесса селекции.

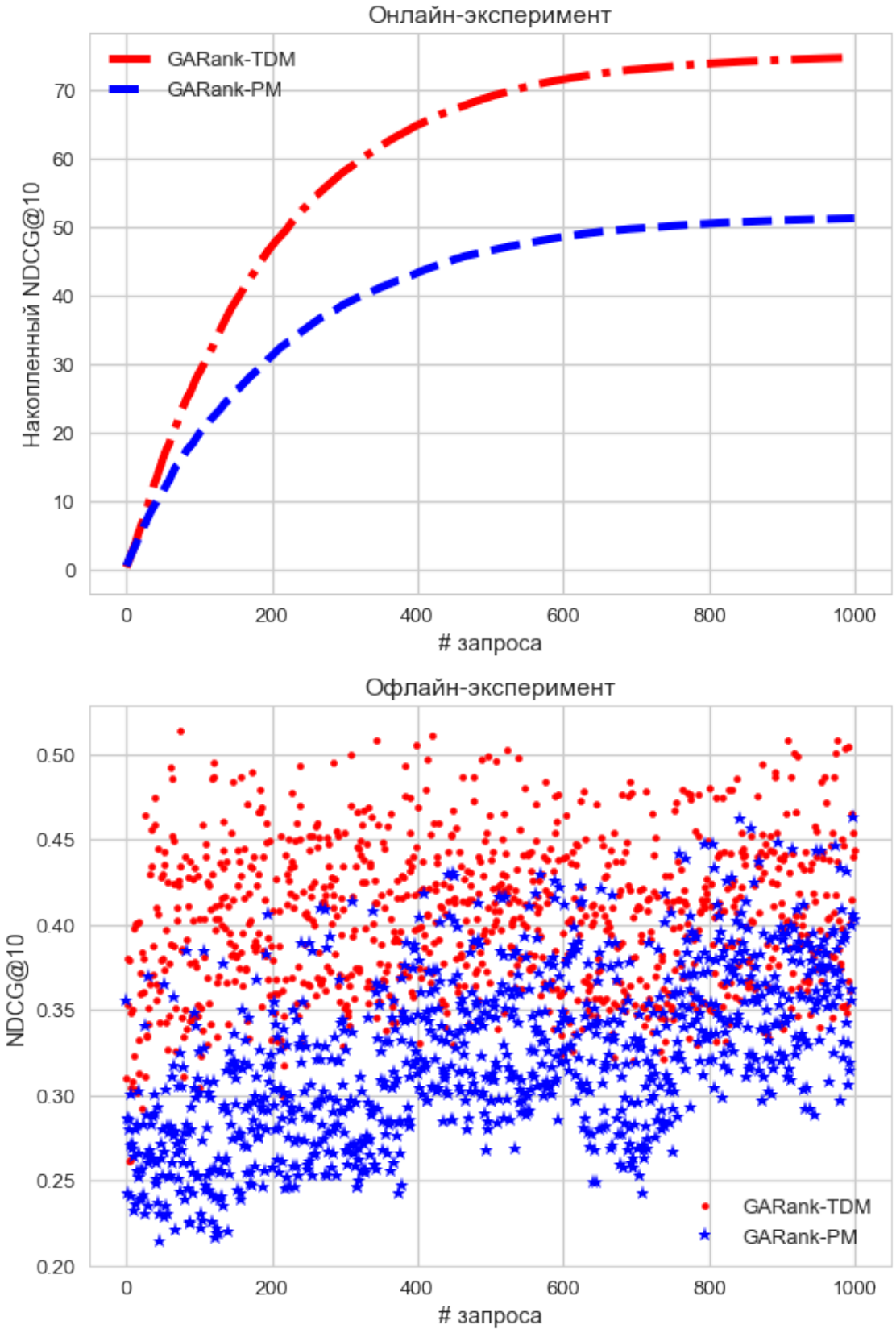

Рис. 3. Сравнение методов смешивания для GARank.
Набор данных OHSUMED, навигационная модель пользователя

Гиперпараметры каждого из алгоритмов указаны в таблицах 5 и 6

для PMGD и GARank-TDM соответственно.

Параметр	Значение
δ - величина шага исследования	1
α - величина шага эксплуатации	0.1
Число кандидатов	19

Таблица 5. Гиперпараметры PMGD

Параметр	Значение
Вероятность скрещивания p_c	0.5
Вероятность мутации p_s	0.3
Размер популяции m	10
Размер турнира при турнирной селекции (k)	3
Размер элитарной группы	1

Таблица 6. Гиперпараметры GARank-TDM

3.5 Результаты

Результаты проведения численных экспериментов приведены в таблицах 7 и 8 для онлайн и офлайн режимов соответственно. Жирным шрифтом выделен лучший результат. Для получения результатов использовалась программная библиотека Lerot [38], а также библиотека эволюционной оптимизации DEAP [39]. Программный код для запуска эксперимента может быть найден по ссылке: https://github.com/Morgolt/lerot-3.

Как видно, основная гипотеза о том, что GARank-TDM должен показывать более оптимальные ранги для пользователя за счет более активного «исследования» во время оптимизации, подтверждается - он значительно превосходит PMGD в онлайн-режиме практически на всех наборах данных. Подобная разница достигается также за счет иного подхода к поиску решения - улучшения набора решений, нежели какого-то одного.

Гипотеза о том, что подобное активное исследование ухудшит офлайн-сходимость также имеет место быть - PMGD опережает GARank-

TDM на всех наборах данных в офлайн-режиме. Это также прямое следствие компромисса между «исследованием» и «эксплуатацией».

Также важно отметить, что GARank-TDM оказался неустойчив к зашумленной обратной связи - его качество, как онлайн так и офлайн, значительно уменьшается при появлении кликов на совершенно нерелевантные документы (информационная модель поведения пользователя). Решение этой проблемы входит в дальнейшее исследование свойств предложенной модели.

Также важным результатом и объектом для дальнейшего исследования является возможная комбинация двух алгоритмов в один мета-алгоритм онлайн-обучения ранжированию. Одним из вариантов подобной комбинации может быть использование метода, основанного на алгоритме многорукого бандита внутри элитной группы популяции. Это возможно, так как и алгоритм многорукого бандита и генетический алгоритм в данной постановке используют одну и ту же оценку качества, основанную на методе смешивания.

Кликовая модель	Набор данных	PMGD-19	GARank-TDM
идеальная	HP2003	764.4	**1342.1**
	HP2004	723.3	**1268.2**
	MQ2007	412.5	**647.8**
	MQ2008	523.2	**839.4**
	NP2003	699.5	**1194.7**
	NP2004	719.9	**1250.1**
	OHSUMED	494.8	**788.4**
	TD2003	312.2	**506.2**
	TD2004	298.9	**476.9**
навигационная	HP2003	701.2	**1159.9**
	HP2004	663.0	**1114.5**
	MQ2007	385.9	**607.8**
	MQ2008	501.5	**789.8**
	NP2003	637.6	**1026.4**
	NP2004	653.2	**1128.2**
	OHSUMED	482.6	**704.0**
	TD2003	272.5	**421.0**
	TD2004	263.3	**389.5**
информационная	HP2003	**650.9**	586.9
	HP2004	**616.1**	475.9
	MQ2007	377.2	**563.5**
	MQ2008	496.3	**724.8**
	NP2003	**603.0**	508.9
	NP2004	**617.8**	530.3
	OHSUMED	474.3	**641.9**
	TD2003	**251.6**	210.1
	TD2004	**245.0**	234.0

Таблица 7. Результаты в онлайн-режиме

Кликовая модель	Набор данных	PMGD-19	GARank-TDM
идеальная	HP2003	**0.782**	0.729
	HP2004	**0.751**	0.675
	MQ2007	**0.406**	0.346
	MQ2008	**0.493**	0.436
	NP2003	**0.719**	0.649
	NP2004	**0.719**	0.655
	OHSUMED	**0.456**	0.413
	TD2003	**0.327**	0.261
	TD2004	**0.333**	0.279
навигационная	HP2003	**0.764**	0.695
	HP2004	**0.740**	0.626
	MQ2007	**0.356**	0.324
	MQ2008	**0.468**	0.415
	NP2003	**0.711**	0.618
	NP2004	**0.717**	0.622
	OHSUMED	**0.439**	0.381
	TD2003	**0.315**	0.225
	TD2004	**0.314**	0.240
информационная	HP2003	**0.759**	0.364
	HP2004	**0.732**	0.299
	MQ2007	**0.340**	0.294
	MQ2008	**0.456**	0.378
	NP2003	**0.704**	0.322
	NP2004	**0.711**	0.320
	OHSUMED	**0.433**	0.340
	TD2003	**0.286**	0.116
	TD2004	**0.299**	0.140

Таблица 8. Результаты в офлайн-режиме

Выводы

В ходе данной работы были достигнуты следующие результаты:

1. Был представлен обзор и подробный анализ современных подходов к онлайн-обучению ранжированию. В частности, был подробно рассмотрен алгоритм многорукого бандита, приведены его сильные и слабые стороны вместе со всеми ограничениями, которые должны быть выполнены для успешного его использования. Затем произведено сравнение подходов к важнейшей составляющей алгоритма - оценке ранжированных списков, приведены доводы в пользу тех или иных подходов и дана ретроспектива исследований в этой области.

2. Как результат рассмотрения достоинств и недостатков существующих подходов, был сформулирован новый подход к оптимизации поисковой выдачи с помощью использования эволюционных алгоритмов, в частности, генетического алгоритма. Приведены теоретические основы данного подхода, возможные его модификации применимо к поставленной задаче и сформулированы гипотезы о предполагаемом качестве предложенного решения.

3. Проведен численный эксперимент на открытых данных с использованием общепринятой в данной области методологии проведения эксперимента. Все исходные данные и полученные результаты приведены полностью, им дана соответствующая аналитическая оценка. Основная задача работы, — улучшение поисковой выдачи в онлайн-режиме, — выполнена, что подтвеждается результатами эксперимента в терминах онлайн-метрики качества — накопленного $NDCG@10$. В итоге сформулированы достоинства и недостатки предложенного подхода, равно как и возможные направления дальнейшего исследования.

Заключение

В данной работе рассмотрена проблема улучшения поисковой выдачи с помощью использования методов онлайн-обучения ранжированию. Был рассмотрен основной подход к данной проблеме в контексте современных исследований — метод обучения ранжированию с помощью алгоритма многорукого бандита. Также рассмотрены возможные его модификации, направленные на улучшение качества ранжирования при помощи более качественного сравнения ранжированных списков — методы смешивания и многоуровневого смешивания. В результате обзора существующих алгоритмов предложен новый метод, основанный на семействе методов оптимизации — эволюционных алгоритмах.

Предложенный алгоритм оптимизации поисковой выдачи с помощью генетического алгоритма показал свою эффективность в онлайн-режиме. В целом, гипотеза о том, что смещение в сторону большей степени «исследования» во время оптимизации улучшает онлайн-качество ранжирования, требует дальнейшего изучения и обоснования на более чем одном семействе методов, однако уже сейчас можно видеть, что отход от классического подхода дает существенный прирост в онлайн-качестве.

Кроме того, алгоритм GARank-TDM показывает определенную неустойчивость к шуму в обратной связи (клики пользователей на нерелевантные документы), для устранения которой необходима модификация непосредственно процесса оптимизации либо использование инструментов по очистке обратной связи от такого шума. Такими инструментами, например, могут быть кликовые модели, способные моделировать поведение пользователя в онлайн-режиме [18].

Еще одним важным преимуществом предложенного метода GARank является его независимость от предполагаемой функции релевантности. Это означает, что в ходе дальнейших исследований, если возникнет такая необходимость, алгоритм будет свободно обобщаться на более узкие задачи информационного поиска, например, персонализация поисковой выдачи или узкоспециализированный поиск внутри одной тематики. Это обеспечивается отутствием необходимости выполнения каких-либо предположений о свойствах оптимизируемой функции при использовании

генетического алгоритма.

В итоге, в ходе данной работы был полностью описан и обоснован новый подход к онлайн-обучению ранжированию, решающий поставленную задачу об оптимизации непосредственно поисковой выдачи, предъявляемой пользователю. Данный подход проиллюстрирован численным экспериментом, результаты которого проанализированы в рамках современной методологии проведения экспериментов в области обучения ранжированию, а также приведено полное сравнение с текущим лучшим результатом в данной области.

Список литературы

1. Järvelin K., Kekäläinen J. Cumulated Gain-based Evaluation of IR Techniques // ACM Trans. Inf. Syst. 2002. Vol. 20, no. 4. P. 422–446.
2. Blair D. C. Information Retrieval, 2nd ed. // Journal of the American Society for Information Science. 1979. Vol. 30, no. 6. P. 374–375.
3. Fuhr N. Optimum Polynomial Retrieval Functions Based on the Probability Ranking Principle // ACM Trans. Inf. Syst. 1989. Vol. 7, no. 3. P. 183–204.
4. Ibrahim O. A. S., Landa-Silva D. ES-Rank: Evolution Strategy Learning to Rank Approach // Proceedings of the Symposium on Applied Computing. New York, NY, USA : ACM, 2017. P. 944–950. (SAC '17).
5. Semenikhin S. V., Denisova L. A. Learning to rank based on modified genetic algorithm // 2016 Dynamics of Systems, Mechanisms and Machines (Dynamics). 2016. P. 1–5.
6. Joachims T. Optimizing Search Engines Using Clickthrough Data // Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Edmonton, Alberta, Canada : ACM, 2002. P. 133–142. (KDD '02).
7. Radlinski F., Kurup M., Joachims T. How Does Clickthrough Data Reflect Retrieval Quality? // Proceedings of the 17th ACM Conference on Information and Knowledge Management. New York, NY, USA : ACM, 2008. P. 43–52. (CIKM '08).
8. Yue Y., Joachims T. Interactively Optimizing Information Retrieval Systems As a Dueling Bandits Problem // Proceedings of the 26th Annual International Conference on Machine Learning. New York, NY, USA : ACM, 2009. P. 1201–1208. (ICML '09).
9. Hofmann K., Whiteson S., Rijke M. de. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval // Information Retrieval. 2013. Vol. 16, no. 1. P. 63–90.

10. Hofmann K., Whiteson S., Rijke M. de. A Probabilistic Method for Inferring Preferences from Clicks // Proceedings of the 20th ACM International Conference on Information and Knowledge Management. New York, NY, USA : ACM, 2011. P. 249–258. (CIKM '11).

11. Radlinski F., Craswell N. Optimized Interleaving for Online Retrieval Evaluation // Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. New York, NY, USA : ACM, 2013. P. 245–254. (WSDM '13).

12. Schuth A., Sietsma F., Whiteson S., Lefortier D., Rijke M. de. Multileaved Comparisons for Fast Online Evaluation // Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. New York, NY, USA : ACM, 2014. P. 71–80. (CIKM '14).

13. Schuth A., Bruintjes R.-J., Buüttner F., Doorn J. van, Groenland C., Oosterhuis H., Tran C.-N., Veeling B., Velde J. van der, Wechsler R., Woudenberg D., Rijke M. de. Probabilistic Multileave for Online Retrieval Evaluation // Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA : ACM, 2015. P. 955–958. (SIGIR '15).

14. Schuth A., Oosterhuis H., Whiteson S., Rijke M. de. Multileave Gradient Descent for Fast Online Learning to Rank // Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. New York, NY, USA : ACM, 2016. P. 457–466. (WSDM '16).

15. Oosterhuis H., Schuth A., Rijke M. de. Probabilistic Multileave Gradient Descent // Proceedings of ECIR. Springer, 03/20/2016. published.

16. Oosterhuis H., Rijke M. de. Balancing Speed and Quality in Online Learning to Rank for Information Retrieval // Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. New York, NY, USA : ACM, 2017. P. 277–286. (CIKM '17).

17. Borisov A., Markov I., Rijke M. de, Serdyukov P. A Neural Click Model for Web Search // Proceedings of the 25th International Conference on World Wide Web. Republic, Canton of Geneva, Switzerland : International World Wide Web Conferences Steering Committee, 2016. P. 531–541. (WWW '16).

18. Guo F., Li L., Faloutsos C. Tailoring Click Models to User Goals // Proceedings of the 2009 Workshop on Web Search Click Data. New York, NY, USA : ACM, 2009. P. 88–92. (WSCD '09).

19. Chuklin A., Markov I., Rijke M. de. Click Models for Web Search. Morgan & Claypool, 2015.

20. Robertson S., Zaragoza H. The Probabilistic Relevance Framework: BM25 and Beyond // Found. Trends Inf. Retr. 2009. Vol. 3, no. 4. P. 333–389.

21. Kelly D. Methods for Evaluating Interactive Information Retrieval Systems with Users // Found. Trends Inf. Retr. 2009. Vol. 3, 1&#8212, 2. P. 1–224.

22. Schuth A., Hofmann K., Radlinski F. Predicting Search Satisfaction Metrics with Interleaved Comparisons // Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA : ACM, 2015. P. 463–472. (SIGIR '15).

23. Kohavi R., Longbotham R., Sommerfield D., Henne R. M. Controlled Experiments on the Web: Survey and Practical Guide // Data Min. Knowl. Discov. 2009. Vol. 18, no. 1. P. 140–181.

24. Joachims T., Granka L., Pan B., Hembrooke H., Radlinski F., Gay G. Evaluating the Accuracy of Implicit Feedback from Clicks and Query Reformulations in Web Search // ACM Trans. Inf. Syst. 2007. Vol. 25, no. 2.

25. Hofmann K., Whiteson S., Rijke M. D. Fidelity, Soundness, and Efficiency of Interleaved Comparison Methods // ACM Trans. Inf. Syst. 2013. Vol. 31, no. 4. 17:1–17:43.

26. Камалов М. В., Мартынов Р. С. Сравнительный анализ алгоритмов онлайн-обучения ранжированию поисковой выдачи // Процессы управления и устойчивость. 2017. т. 4, № 1. с. 382—388.

27. Chen W., Liu T.-y., Lan Y., Ma Z.-m., Li H. Ranking Measures and Loss Functions in Learning to Rank // Advances in Neural Information Processing Systems 22 / ed. by Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, A. Culotta. Curran Associates, Inc., 2009.

28. Taylor M., Guiver J., Robertson S., Minka T. SoftRank: Optimizing Non-smooth Rank Metrics // Proceedings of the 2008 International Conference on Web Search and Data Mining. New York, NY, USA : ACM, 2008. P. 77–86. (WSDM '08).

29. Thede S. M. An Introduction to Genetic Algorithms // J. Comput. Sci. Coll. 2004. Vol. 20, no. 1. P. 115–123.

30. Mitchell M. An Introduction to Genetic Algorithms. Cambridge, MA, USA : MIT Press, 1998.

31. Zhao T., King I. Constructing Reliable Gradient Exploration for Online Learning to Rank // Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. New York, NY, USA : ACM, 2016. P. 1643–1652. (CIKM '16).

32. Holland J. H. Genetic Algorithms // Scientific American. 1992.

33. Genetic and Evolutionary Computation - GECCO 2003, Genetic and Evolutionary Computation Conference, Chicago, IL, USA, July 12-16, 2003. Proceedings, Part II. Vol. 2724 / ed. by E. Cantú-Paz, J. A. Foster, K. Deb, L. Davis, R. Roy, U. O'Reilly, H. Beyer, R. K. Standish, G. Kendall, S. W. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. A. Dowsland, N. Jonoska, J. F. Miller. Springer, 2003. (Lecture Notes in Computer Science).

34. Introducing LETOR 4.0 Datasets. https://arxiv.org/ftp/arxiv/papers/1306/1306.2597.pdf.

35. Kleinberg J. M. Authoritative Sources in a Hyperlinked Environment // J. ACM. New York, NY, USA, 1999. Sept. Vol. 46, no. 5.

36. Page L., Brin S., Motwani R., Winograd T. The PageRank Citation Ranking: Bringing Order to the Web. 1999. Nov.

37. Hofmann K., Schuth A., Whiteson S., Rijke M. de. Reusing Historical Interaction Data for Faster Online Learning to Rank for IR // Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. New York, NY, USA : ACM, 2013. P. 183–192. (WSDM '13).

38. Schuth A., Hofmann K., Rijkem Maarten de. Lerot: an Online Learning to Rank Framework // Living Labs for Information Retrieval Evaluation workshop at CIKM'13. 2013.

39. Fortin F.-A., De Rainville F.-M., Gardner M.-A., Parizeau M., Gagné C. DEAP: Evolutionary Algorithms Made Easy // Journal of Machine Learning Research. 2012. Vol. 13. P. 2171–2175.