



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение эвм и информационные технологии»

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ  
НА ТЕМУ:**

***«Применение машинного обучения в  
поисковых системах»***

Студент      ИУ7-51Б

\_\_\_\_\_      Волков Г.В.

Руководитель

\_\_\_\_\_      Шаповалова М.С.

2022 г.

**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)  
(МГТУ им. Н.Э. Баумана)**

**УТВЕРЖДАЮ**

**Заведующий кафедрой ИУ-7  
(Индекс)**

**И. В. Рудаков  
(И.О.Фамилия)  
«16» сентября 2022 г.**

## **З А Д А Н И Е на выполнение научно-исследовательской работы**

по теме

**«Применение машинного обучения в поисковых системах»**

Студент группы ИУ7-51Б

**Волков Георгий Валерьевич**

Направленность НИР

**учебная**

Источник тематики

**НИР кафедры**

График выполнения НИР: 25% к 6 нед., 50% к 9 нед., 75% к 12 нед., 100% к 15 нед.

### **Техническое задание**

*Провести обзор существующих методов оптимизации поиска информации с использованием машинного обучения. Провести анализ предметной области применения машинного обучения в поисковых системах, сформулировать критерии сравнения рассмотренных методов. Классифицировать существующие способы оптимизации поиска информации с применением машинного обучения.*

### **Оформление научно-исследовательской работы:**

Расчетно-пояснительная записка на **12-20** листах формата А4.

Перечень графического (илюстративного) материала (чертежи, плакаты, слайды и т. п.)

Презентация на **6-10** слайдах.

Дата выдачи задания «16» сентября 2022 г.

**Руководитель НИР**

\_\_\_\_\_  
(Подпись, дата)

**Шаповалова М.С.**

(И.О.Фамилия)

**Студент**

\_\_\_\_\_  
(Подпись, дата)

**Волков Г.В.**

(И.О.Фамилия)

# РЕФЕРАТ

Расчетно-пояснительная записка 33 с., 1 рис., 5 табл., 23 источн., 1 прил.  
МАШИННОЕ ОБУЧЕНИЕ, ОБУЧЕНИЕ РАНЖИРОВАНИЮ,  
НЕЙРОННЫЕ СЕТИ, LINEAR REGRESSION, RANKING SVM, LAMBDA  
RANK, LISTNET

Объектом исследования являются методы машинного обучения, применяемые в поисковых системах.

Цель работы — рассмотрение алгоритмов машинного обучения, в частности, их применение в поисковых системах.

В процессе работы были изучены существующие методы обучения ранжированию и проведён их сравнительный анализ.

# СОДЕРЖАНИЕ

<b>РЕФЕРАТ</b>	<b>3</b>
<b>ВВЕДЕНИЕ</b>	<b>5</b>
<b>1 Анализ предметной области</b>	<b>7</b>
1.1 Основные понятия . . . . .	7
1.2 Применение машинного обучения в поисковых системах . . . . .	8
<b>2 Исследование существующих алгоритмов обучения ранжированию</b>	<b>10</b>
2.1 Классификация алгоритмов . . . . .	10
2.2 Алгоритмы обучения ранжированию . . . . .	11
2.2.1 Алгоритм Linear Regression . . . . .	12
2.2.2 Алгоритм Ranking SVM . . . . .	14
2.2.3 Алгоритм LambdaRank . . . . .	16
2.2.4 Алгоритм ListNet . . . . .	17
<b>3 Сравнение алгоритмов обучения ранжированию</b>	<b>20</b>
3.1 Описание данных . . . . .	20
3.2 Критерии сравнения . . . . .	21
3.3 Сравнение алгоритмов . . . . .	26
<b>ЗАКЛЮЧЕНИЕ</b>	<b>29</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	<b>30</b>
<b>ПРИЛОЖЕНИЕ А Презентация научно-исследовательской работы</b>	<b>33</b>

# ВВЕДЕНИЕ

Машинное обучение — термин, обозначающий набор методов и инструментов, которые помогают компьютерам самостоятельно обучаться и адаптироваться. Алгоритмы машинного обучения позволяют решать задачи, не будучи явно запрограммированными на выполнение желаемого действия. Обучение заключается в обработке специально подобранные входные данные. Технологии связанные с машинным обучением стремительно развиваются и используются людьми во многих сферах жизни, например:

- распознавание изображений и речи;
- прогнозирование автомобильного трафика;
- рекомендательные системы;
- автопилот;
- медицина;
- виртуальные помощники;
- поисковые системы.

Актуальность этой темы объясняется широким распространением поисковых систем. Множество людей ежедневно совершают большое число запросов в поисках нужной информации, Google обрабатывает 3.5 миллиарда запросов в день или почти 40 тысяч запросов каждую секунду [1]. Для ускорения и улучшения качества поисковой выдачи применяются методы машинного обучения, а именно обучение ранжированию. Современные поисковые системы используют машинное обучение, например, Матрикснет в Яндексе, JAX в Google.

Целью данной научно-исследовательской работы — изучить алгоритмы машинного обучения, применяемые в поисковых системах.

Для достижения поставленной в работе цели предстоит решить следующие задачи:

- изучить основные понятия алгоритмов обучения ранжированию;

- описать и классифицировать существующие алгоритмы;
- произвести сравнительный анализ рассмотренных алгоритмов.

# 1 Анализ предметной области

## 1.1 Основные понятия

Машинное обучение — раздел информатики, посвященный созданию алгоритмов, опирающихся на набор данных о каком-либо явлении. Данные получают из естественной среды, создают вручную или генерируют другим алгоритмом. Методы машинного обучения формируют статистическую модель на основе специально подобранных обучающих данных, которую потом используют для решения практических задач [2].

Выделяется три основных способа обучения: с учителем, без учителя и с подкреплением. Способы и их применения проиллюстрированы на рисунке 1.1.

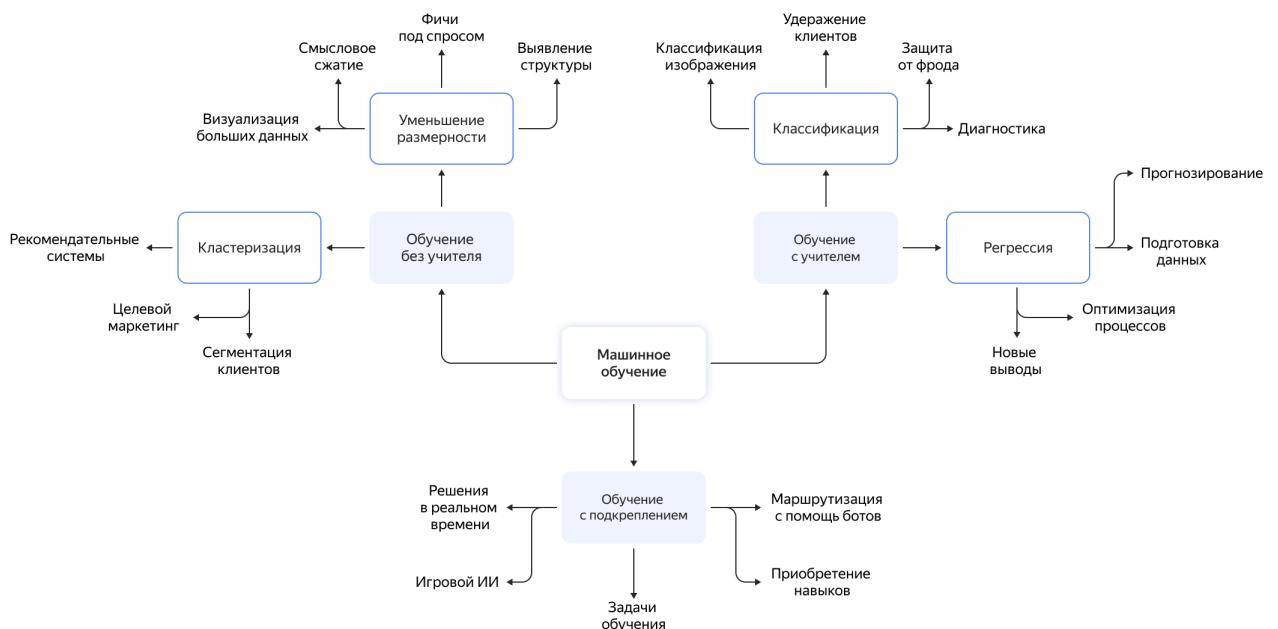


Рисунок 1.1 – Способы обучения

В обучении с учителем набор данных организован как коллекция размеченных образцов вида «известный вход — известный выход», называемый обучающей выборкой. Входные данные являются векторами признаков, описывающими некоторые характеристики образца. Выходом же является предсказание модели, которое может быть элементом конечного множества, вещественным числом или более сложной структурой, например, значение метки, если используется регрессия. Существует некоторая зависимость между

входными и выходными данными, которую в результате обучения должна отражать модель. После обучения она способная выдать достаточно точный результат для любых входных данных. Данный подход в основном используется для решения задач регрессии и классификации [2].

Обучение без учителя — класс задач машинного обучения, в которых известны только описания множества объектов, подаваемых на вход модели, и требуется обнаружить внутренние взаимосвязи между этими объектами. При обучении на вход модели подаётся вектор признаков, а целью алгоритма обучения без учителя является — создание модели, которая принимает входные параметры и преобразует их в значение, которое можно использовать для решения практической задачи. Данный подход используются для задач кластеризации, уменьшения размерности и выявления аномалий [2].

Обучение с подкреплением — это раздел машинного обучения, в котором предполагается, что машина «живет» в каком-то виртуальном окружении и способна воспринимать его как вектор характеристик. Вектор содержит данных о «мире» необходимые для обучения модели. Компьютер взаимодействует с этим окружением, параллельно обучаясь, и получает вознаграждение или штраф за выполнение действий, которые помогают правильно корректировать веса модели, то есть обучается методом проб и ошибок. Цель алгоритма обучения с подкреплением — выучить линию поведения, которая приводит к максимальному ожидаемому вознаграждению [2].

## **1.2 Применение машинного обучения в поисковых системах**

Работа поисковой системы заключается в том, чтобы по запросу пользователя найти документы, наилучшим образом подходящие его запросу. При этом основными критериями качества результатов информационного поиска являются полнота, точность и оперативность. Сайты в поисковой выдаче сортируются в соответствии с факторами, используемыми поисковой машиной, то есть проходят процедуру ранжирования [3].

Существует множество методов подбора формулы для ранжирования, но один из самых популярных — на основе машинного обучения, а именно —

обучение ранжированию с учителем. Целью этих методов является подбор ранжирующей модели по обучающей выборке, которая способна наилучшим образом приблизить и обобщить способ ранжирования на новые данные. Для получения набора примеров используются асессоры, которые оценивают степень релевантности документа запросу. Данные для обучения характеризуются большим объёмом (100 – 500 тысяч примеров) и большой размерностью (100 – 1000 признаков) [3].

## 2 Исследование существующих алгоритмов обучения ранжированию

### 2.1 Классификация алгоритмов

Существующие алгоритмы обучения ранжированию делятся на три группы [3]: поточечный, попарный и списочный.

Поточечный подход — частный случай задачи регрессии или классификации. На практике поточечный подход дает не очень качественный результат, так как каждый документ ранжируется независимо от других. В качестве примеров для обучения используют пары «признаки — значение релевантности». Точечный подход направлен на обучение модели, предсказывающей значение релевантности или порядковый номер документа. Окончательное ранжирование достигается путем простой сортировки списка результатов по оценке документа, выданной обученной моделью [4].

При попарном подходе обучение ранжированию сводится к построению бинарного классификатора, которому на вход поступают два документа, соответствующих одному и тому же запросу, и требуется определить, какой из них будет более релевантным. Недостатком является невозможность учета сразу всех документов запроса. Целью обучения таких алгоритмов является минимизация ошибок в классификации пар документов, а не в их ранжировании. Процесс обучения требует больших вычислительных затрат, поскольку количество пар документов очень велико. Количество сгенерированных пар документов в значительной степени варьируется от запроса к запросу, что приводит к обучению модели, ориентированной на запросы с большим количеством пар, то есть обучается неравномерно. В итоге результаты ранжирования для запросов похожих на запросы из обучающие выборки, которые имеют меньшее количеством пар документов, будут значительно хуже. При ранжировании алгоритмы получают пару документов на вход и определяет их порядок относительно друг друга [4].

Списочный подход заключается в построении модели, на вход которой поступают сразу все документы, соответствующие запросу, а на выходе получается отранжированная перестановка этих документов. Подгонка парамет-

ров модели осуществляется для прямой максимизации одной из метрик ранжирования. Но это затруднительно, так как метрики ранжирования обычно не непрерывны и недифференцируемы относительно параметров ранжирующей модели, поэтому прибегают к максимизации неких их приближений или нижних оценок. С точки зрения поставленной задачи ранжирования, списочные методы решают её исходную постановку. Они обладают одними из лучших показателей метрик, оценивающих ранжирование, на данный момент, но сложны в разработке и подборе обучающих наборов данных. При списочном подходе алгоритмы непосредственно просматривают весь список документов и подбирают для него оптимальный порядок [4].

## 2.2 Алгоритмы обучения ранжированию

В данном разделе будут рассмотрены несколько популярных алгоритмов из каждой категории: Linear Regression (поточечный), Ranking SVM (по-парный), LambdaRank(по-парный), ListNet(списочный).

Перед описанием алгоритмов введём следующие обозначения:

- $q_i$  —  $i$ -ый поисковый запрос;
- $D_i$  — список всех документов, которые ассоциированы с  $q_i$  запросом;
- $d_{i,j}$  —  $j$ -ый документ из списка  $D_i$ ;
- $\pi_i$  — отсортированный список  $D_i$ ;
- $y_{i,j}$  — метка, показывающая релевантность документа  $d_{i,j}$  запросу  $q_i$ ;
- $y_i$  — вектор меток  $y_{i,j}$ .

При ранжировании документов признаки формируются не только на основании документа или запроса по отдельности, но и как функция от поискового запроса и документа в совокупности. Такие признаки способны хорошо показывать степень релевантности документа запросу, так как лучше отображают связь запроса и документа. Зачастую используются следующие признаки:

- текстовые — описывают количество и место слов встречающихся в документе и запросе. Используются меры, такие как TF-IDF, используемая для оценки важности слов в контексте документа, BM 25 и различные языковые модели;
- ссылочные — описывают количество гиперссылок на документ на других сайтах и сколько внутри самого документа полезных ссылок, удаляющих запросу. Мера PageRank назначает каждому документу некоторое численное значение, измеряющее его «важность» или «авторитетность» среди остальных документов основываясь на информации о гиперссылках на эту страницу. HITS позволяет находить страницы, соответствующие запросу пользователя, на основе информации, заложенной в ссылках, находящихся в документе. Идея основана на предположении, что гиперссылки кодируют значительное количество скрытых сайтов соответствующих запросу пользователя. Рекурсивно просматривая ссылки, находящиеся на странице, алгоритм оценивает их содержание и в соответствии с этим переоценивает страницу, содержащую эти ссылки;
- кликовые — описывают количество кликов пользователей на документ в поисковой выдаче. Учитывается как общее количество кликов, так и по какому-то определённому запросу.

### 2.2.1 Алгоритм Linear Regression

Метод обучения на основе регрессии используется для решения задачи оптимизации метрик DCG [5]. Пусть  $\mathcal{F}$  — функциональное пространство, содержащее функции  $X \times S \rightarrow \mathcal{F}$ , где  $X$  — множество векторов представления документов,  $S$  — множества всех конечных подмножеств  $X$ . Произвольно созданы множества  $S_1, \dots, S_n$ , где  $S_i = \{x_{i,1}, \dots, x_{i,m}\}$ , с соответствующими оценками  $\{y_{i,j}\} = \{y_{i,1}, \dots, y_{i,m}\}$ . Тогда для решения проблемы ранжирования можно использовать простой подход, основанный на регрессии:

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \left[ \sum_{j=1}^m (f(x_{i,j}, S_i) - y_{i,j})^2 \right]. \quad (2.1)$$

Регрессия — задача прогнозирования метки по набору признаков объектов. Метка здесь может принимать любое значение. Алгоритм моделирует зависимость меток от признаков, чтобы определить закономерности изменения меток в зависимости характеристик объекта. При обучении алгоритму на вход подаются примеры с известными метками. Результатом работы является функция, способная прогнозировать значение метки для новых данных.

Для возможности предсказания меток в регрессии гиперплоскость проводится так, чтобы она оказалась как можно ближе к обучающим примерам, которые представляют собой точку в пространстве размерностью равной длине вектора характеристик каждого объекта [2]. Для этого требуется минимизировать целевую функцию (2.1). Выражение  $(f(x_{i,j}, S_i) - y_{i,j})^2$  называют функцией потерь, в данном случае используется квадратичная функция потерь. Она определяет величину штрафа за неправильную классификацию. В этом алгоритме функция стоимости определена как средняя потеря. Минимизировав целевую функцию получим оптимальные параметры модели, для предсказания значения метки.

Однако метод прямой регрессии не подходит для крупномасштабных задач ранжирования, таких как веб-поиск, для которых требуется ранжировать множество элементов. Хорошим результатом работы такого алгоритма будет выбор страниц с самым высоким рейтингом. Данный метод уделяет равное внимание релевантным и нерелевантным страницам. На самом деле, следует уделять больше внимания страницам с самым высоким рейтингом. Оценка страниц с низкой релевантностью не нуждается в большой точности, до тех пор, пока не происходит новая оценка их релевантности для того, чтобы эти страницы отображались на верхних позициях в поиске. С учётом этих рассуждений стоит изменить формулу (2.1) следующим образом:

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n L(f, S_i, \{y_{i,j}\}), \quad (2.2)$$

где для  $S = \{x_1, \dots, x_m\}$ , с соответствующим  $\{y_j\}$ , имеем функцию потерь:

$$\begin{aligned} L(f, S, \{y_j\}) &= \\ &= \sum_{j=1}^m w(x_j, S)(f(x_j, S) - y_j)^2 + u \max_j w'(x_j, S)(f(x_j, S) - \delta(x_j, S))_+^2, \end{aligned}$$

где  $u$  — неотрицательный параметр. Весовая функция  $w(x_j, S)$  выбрана таким образом, чтобы она фокусировалась только на наиболее важных документах. Во второй части уравнения  $w'(x_j, S)$  выбирается так, чтобы она фокусировалась на страницах, не охваченных  $w(x_j, S)$ . А  $\delta(x_j, S)$  выбирается в качестве нижнего порога. Хотя  $t$  часто очень велико, количество точек, при которых  $w(x_j, S)$  отлично от нуля, зачастую достаточно мало. Более того,  $(f(x_j, S) - \delta(x_j, S))_+$  не равно нулю только тогда, когда  $f(x_j, S) \geq \delta(x_j, S)$ . Следовательно, полностью игнорируются низкоранговые страницы, что делает алгоритм более эффективным даже при большом  $t$ . Формулы (2.1) и (2.2) [5].

### 2.2.2 Алгоритм Ranking SVM

Ключевая идея алгоритма, предложенного Р. Хербрихом [6], заключается в использовании метода опорных векторов для попарного сравнения документов на то, какой из элементов более релевантен запросу.

Метод опорных векторов — алгоритм обучения с учителем, используемый для задач классификации. Суть метода заключается в построении гиперплоскости так, чтобы расстояние от неё до ближайшей точки было максимальным [2]. Это эквивалентно тому, что сумма расстояний до гиперплоскости от двух ближайших к ней точек, лежащих по разные стороны от неё, максимальна. На основе взаимоположения точки, представляющей объект, и гиперплоскости происходит классификация.

При обучении на вход подаются набор пар вида «точка — метка класса», где метка класса это 1 или  $-1$ . Строится гиперплоскость имеющая вид  $w \cdot x - b = 0$ , где  $x$  — входной вектор признаков. Так как ищется оптимальное разделение, особый интерес представляют гиперплоскости, параллельные оптимальной и ближайшие к опорным векторам двух классов. Можно показать, что эти параллельные гиперплоскости могут быть описаны следующими уравнениями:

$$\begin{aligned} w \cdot x - b &= 1, \\ w \cdot x - b &= -1. \end{aligned} \tag{2.3}$$

Ширина полосы легко найти из соображений геометрии. Она равна  $\frac{2}{\|w\|}$ .

Тогда поставленая перед нами задача состоит в минимизации  $\|w\|$ . Чем меньше  $\|w\|$ , тем больше расстояние между этими двумя плоскостями. Большой зазор способствует лучшему обобщению, то есть качеству классификации новых данных предложенной моделью. Чтобы исключить все точки  $x_i$  из полосы необходимо включить следующие ограничения:

$$\begin{aligned} w \cdot x - b &\geq 1, \text{ если } y_i = +1, \\ w \cdot x - b &\leq -1, \text{ если } y_i = -1, \end{aligned} \quad (2.4)$$

где  $y_i$  — метка. Уравнения (2.4) можно записать кратко в виде:

$$y_i(w \cdot x - b) \geq 1. \quad (2.5)$$

В случае линейной неразрешимости, алгоритму позволяет допускать ошибки на обучающей выборке. Для этого вводятся переменные  $\xi_i$  характеризующие величину ошибки на объектах  $x_i$  и коэффициент  $C$  — параметр, позволяющий настраивать отношение между максимизацией ширины разделяющей полосы и минимизацией суммарной ошибки. Тогда задача SVM формулируется как система (2.6). Формулы (2.3)–(2.5) [2].

Пусть имеется функция  $f(x)$ , которая определяет релевантность документа относительно запроса. Тогда утверждение о том, что документ  $x_i$  релевантнее чем документ  $x_j$  ( $x_i \succ x_j$ ) эквивалентно тому, что  $f(x_i) > f(x_j)$ . Тогда если выбрать функцию  $f(x) = (\omega, x)$  то имеем, что:

$$f(x_i) > f(x_j) \iff (\omega, x_i - x_j) > 0.$$

Теперь рассматривая разность векторов как новые объекты  $\hat{x}_{i,j} = x_i - x_j$ , получаем стандартную постановку SVM алгоритма. Теперь задача ставится следующим образом:

$$\begin{cases} \frac{1}{2}\|\omega\|^2 + C \sum_{i=1}^N \xi_i \rightarrow \min_{\omega, \xi}, \\ y_i(\omega, x_i^1 - x_i^2) \leq 1 - \xi_i, \\ \xi_i \geq 0, \end{cases} \quad (2.6)$$

где  $N$  — количество пар объектов,  $x_i^1, x_i^2$  — первый и второй объект пары

соответственно. При этом  $y_i = 1$ , если  $x_i^1 \succ x_i^2$ , иначе  $y_i = -1$ . Допустимы парами являются документы из одного списка с различной степенью релевантности. Формула (2.6) [6].

### 2.2.3 Алгоритм LambdaRank

В поточечных и попарных методах ранжирования итоговый функционал при обучении обычно не дифференцируется, так как он зависит от порядка элементов. Рассмотрим работу метода для одного поискового запроса. В данном алгоритме, описанном Бургесом [7], не определяется непрерывное приближение функционала  $L$ , вместо этого используется градиент функционала на всем списке документов:

$$\frac{\partial L}{\partial s_i} = -\lambda(s_1, y_1, \dots, s_n, y_n), \quad (2.7)$$

где  $s_i$  — рейтинг документа,  $y_i$  — степень релевантности, а  $n$  — количество документов. Градиент выбранного документа зависит от рейтингов и степени релевантности. Он пересчитывается после каждой генерации списка.

Один из способов повышения эффективность алгоритма — увеличение градиента документов на первых позициях. Для двух документов  $d_i$  и  $d_j$  имеем, если  $d_i \succ d_j$ , то  $|\frac{\partial L}{\partial s_i}| > |\frac{\partial L}{\partial s_j}|$ .

Метод позволяет настраиваться на широкий класс функционалов качества, однако в стандартном варианте  $\lambda$  вычисляется по метрике NDCG:

$$\lambda_i = \frac{\partial L}{\partial s_i} = \frac{1}{G_{max}} \sum_j \left( \frac{1}{1 + \exp(s_j - s_i)} \right) (G(y_j) - G(y_i)) (D(\pi_j) - D(\pi_i)), \quad (2.8)$$

где  $G$ ,  $D$  — функции преобразования релевантности документа в его рейтинг и значимости порядкового номера документа в ранжированном списке.  $\lambda_i$  — показывает насколько надо увеличить рейтинг  $i$ -го документа. Для этого необходимо изменить веса  $\omega$ :

$$\frac{\partial L}{\partial \omega} = \sum_{i=1}^n \frac{\partial s_i}{\partial \omega} \sum_{j \in P_i} \frac{\partial L(s_i, s_j)}{\partial s_i} + \sum_{j=1}^n \frac{\partial s_j}{\partial \omega} \sum_{i \in P_j} \frac{\partial L(s_i, s_j)}{\partial s_j}, \quad (2.9)$$

где  $P_i$ ,  $P_j$  — множество пар документов с индексами  $j$  и  $i$ , для которых существуют пары  $(i, j)$  во множестве пар документов  $P$ , соответственно.

Таким образом, алгоритм LambdaRank заключается в итерационном пересчете весов  $\omega$ :

$$\omega = \omega - \eta \frac{\partial L}{\partial \omega}, \quad (2.10)$$

где  $\eta$  — итерационный шаг. Если запросов несколько, то необходимо расширить множество  $P$ , которое может быть построено как и в методе Ranking SVM. Формулы (2.7)–(2.10) [7].

В данном методе для минимизации целевой функции используется градиентный спуск, суть которого заключается в использовании градиента для поиска локального минимума целевой функции. За счёт частных производных, домноженных на  $-1$ , при каждой итерации её параметры пересчитываются таким образом, что происходит движение в сторону локального минимума [2].

## 2.2.4 Алгоритм ListNet

Впервые алгоритм был описан З. КАО [8]. Цель обучения формализована как минимизация общих потерь в отношении обучающих данных.

$$\sum_{i=1}^m L(y^{(i)}, z^{(i)}), \quad (2.11)$$

где  $L$  — функция потерь по списку,  $y^{(i)}$  — эталонный список рейтингов,  $z^{(i)}$  — список рейтингов, созданный ранжирующей моделью. В данном алгоритме предлагается использовать вероятностные модели для вычисления функции потерь  $L$  по списку. Пусть набор объектов, подлежащих ранжированию, идентифицируется числами  $1, 2, \dots, n$ . Перестановка  $\pi$  на объектах определяется как биекция из  $\{1, 2, \dots, n\}$  в саму себе, т. е.  $\pi = (\pi(1), \dots, \pi(n))$ . Здесь  $\pi(j)$  обозначает объект в позиции  $j$  в перестановке. Множество всех возможных перестановок из  $n$  объектов обозначается как  $\Omega_n$ . Пусть существует функция ранжирования, которая присваивает оценки  $n$  объектам. Обозначим список оценок как  $s = (s_1, s_2, \dots, s_n)$ , где  $s_j$  — оценка  $j$ -го объекта.

Предполагается, что возможна любая перестановка, но разные переста-

новки могут иметь разную вероятность. Определим вероятность перестановки таким образом, чтобы она обладала желаемыми свойствами. Тогда вероятность перестановки  $\pi$ , заданная списком оценок  $s$ , определяется как:

$$P_s(\pi) = \prod_{j=1}^n \frac{\phi(s_{\pi(j)})}{\sum_{k=j}^n \phi(s_{\pi(k)})}, \quad (2.12)$$

где  $s_{\pi(j)}$  — оценка объекта в позиции  $j$  перестановки  $\pi$ ,  $\phi$  — возрастающая и строго положительная функция.

Для любого списка, если поменять позицию объекта с высоким и низким баллом местами, получим список с меньшей вероятностью перестановки. Список объектов, отсортированных на основе функции ранжирования, имеет наибольшую вероятность перестановки, вычисленную по формуле 2.12, в то время как список объектов, отсортированных в обратном порядке, имеет наименьшую вероятность. То есть наиболее вероятной перестановкой является, отсортированная с помощью функции ранжирования.

Имея два списка оценок, мы можем сначала вычислить из них распределения вероятностей перестановок, а затем вычислить расстояние между двумя распределениями как функцию потерь по списку. Однако, поскольку число перестановок равно  $n!$ , вычисление может оказаться крайне трудозатратным.

Чтобы справиться с данной проблемой, рассмотрим использование вероятности того, что  $j$ -ый документ будет ранжирован выше всех, учитывая оценки остальных документов

$$P_s(j) = \frac{\phi(s_j)}{\sum_{k=1}^n \phi(s_k)}. \quad (2.13)$$

Используя перекрестную энтропию в качестве метрики, функцию потерь (2.11) можно записать:

$$L(y^{(i)}, z^{(i)}) = - \sum_{j=1}^n P_{y^{(i)}}(j) \log(P_{z^{(i)}}(j)). \quad (2.14)$$

Обозначим функцию ранжирования, основанную на модели  $\omega$  как  $f_\omega$ . Функция  $f_\omega$  присваивает вектору признаков  $x_j^{(i)}$  значение рейтинга. Определим  $\phi$  как экспоненциальную функцию, тогда функцию вероятности (2.13)

можно записать как:

$$P_s(j) = \frac{\phi(s_j)}{\sum_{k=1}^n \phi(s_k)} = \frac{\exp(s_j)}{\sum_{k=1}^n \exp(s_k)}. \quad (2.15)$$

Имея запрос  $q^{(i)}$  ранжирующая функция создаёт список рейтингов

$$z^{(i)}(f_\omega) = (f_\omega(x_1^{(i)}), f_\omega(x_2^{(i)}), \dots, f_\omega(x_{n^{(i)}}^{(i)})).$$

Тогда функцию вероятности (2.15) можно записать как:

$$P_{z^{(i)}(f_\omega)}(x_j^{(i)}) = \frac{\exp(f_\omega(x_j^{(i)}))}{\sum_{k=1}^{n^{(i)}} \exp(f_\omega(x_k^{(i)}))}. \quad (2.16)$$

С перекрестной энтропией в качестве метрики, функции потери (2.14) примет вид:

$$L(y^{(i)}, z^{(i)}(f_\omega)) = - \sum_{j=1}^{n^{(i)}} P_{y^{(i)}}(x_j^{(i)}) \log(P_{z^{(i)}(f_\omega)}(x_j^{(i)})). \quad (2.17)$$

Градиент функции потери можно найти по следующей формуле:

$$\begin{aligned} \Delta\omega &= \frac{\partial L(y^{(i)}, z^{(i)}(f_\omega))}{\partial \omega} = - \sum_{j=1}^{n^{(i)}} P_{y^{(i)}}(x_j^{(i)}) \frac{\partial f_\omega(x_j^{(i)})}{\partial \omega} + \\ &\quad + \frac{1}{\sum_{j=1}^{n^{(i)}} \exp(f_\omega(x_j^{(i)}))} \sum_{j=1}^{n^{(i)}} \exp(f_\omega(x_j^{(i)})) \frac{\partial f_\omega(x_j^{(i)})}{\partial \omega}. \end{aligned} \quad (2.18)$$

Метод ListNet заключается в итерационном пересчете  $\Delta\omega$  и обновлении весов модели:  $\omega = \omega - \eta \Delta\omega$ . Используется метод градиентного спуска, описанный выше. Формулы (2.11)–(2.18) [8].

### 3 Сравнение алгоритмов обучения ранжированию

#### 3.1 Описание данных

Сравнение алгоритмов производилось по наборам данных таких коллекций как: AOL, LETOR 3, LETOR 4, MSLR, WCL2R, Yahoo! Learning to Rank Challenge. В датасетах не содержатся сами документы, а только векторы их признаков. Таким образом, любая разница в производительности ранжирования обусловлена только алгоритмом ранжирования. Также в датасетах присутствуют сами запросы соответствующие документам.

LETOR 3 состоит из двух частей: «Gov» и OHSUMED. В «Gov» содержится около одного миллиона документов. В нём определены три задачи поиска: определение темы (TD), поиск домашней страницы (HP) и поиск именованной страницы (NP). Определение темы направлено на поиск списка точек входа на веб-сайты, в основном посвященные данной теме, то есть сайтам содержащим ссылки на искомые страницы. Поиск домашней страницы направлен на возврат некоторой главной страницы, соответствующей запросу. Поиск именованной страницы заключается в поиске страницы, название которой идентично запросу. В принципе, существует только один ответ для поиска домашней страницы и поиска именованной страницы. OHSUMED является подмножеством базы данных о медицинских публикациях MEDLINE. Он состоит примерно из 0,3 миллиона записей из 270 медицинских журналов. Поля записи включают название, аннотацию, ключевые слова, автора, источник и тип публикации [11].

LETOR 4 содержит два набора данных (MQ2007, MQ2008) для четырех настроек ранжирования: контролируемое, полу-контролируемое, ранжирование по списку и агрегирование рангов [12].

Набор данных AOL был создан на основе логов коммерческой поисковой системы. Логи представляют собой выборку поисковой активности 658000 анонимизированных пользователей из США за трехмесячный период. Набор данных содержит 4,8 миллиона запросов и 1,8 миллиона URL-адресов. В этом наборе данных предполагается, что релевантность документа для запроса

пропорциональна количеству раз, когда пользователи кликали не него [12].

Два набора данных Yahoo происходят из разных регионов. Первый соответствует США, другой - азиатским странам. Оба набора данных фактически являются подмножеством всего обучающего набора, используемого внутри компании для обучения функций ранжирования поисковой системы «Yahoo!» [13].

Набор WCL2R создан на основе логов чилийской поисковой системы TodoCL. Данные собирались в течении 10 месяцев [14].

Наборы данных MSLR-WEB10K состоит из 10000 запросов. Наборы данных состоят из векторов объектов, извлеченных из пар «запрос – URL», а также меток оценки релевантности. Оценки получены на основе устаревшего набора меток коммерческой поисковой системы в Интернете Bing [15].

Набор данных Istella использовался в прошлом для обучения одного из этапов конвейера ранжирования Stellar production. Является одним из самых больших общедоступных наборов данных. Полный набор данных состоит из 33018 запросов и 220 объектов, представляющих каждую пару запрос–документ. Он состоит из 10454629 примеров, помеченных значениями релевантности [16].

### 3.2 Критерии сравнения

Наиболее распространенный подход к оценке эффективности ранжирующего отображения основан на проверке критериев точности и полноты поиска, так как эти факторы являются основными требованиями предъявляемыми к поисковым системам [9]. Поэтому в качестве критериев сравнения в данной работе используются метрики MAP и NDCG.

Рассмотрим множество документов  $D$ . Для каждого запроса  $q$  результатом работы алгоритма ранжирования будет отображение  $f : d \rightarrow R$ , ставящее в соответствие каждой странице ей рейтинг, который тем больше, чем более документ  $d$  релевантен к запросу  $q$ . Для оценки качества ранжирования нужен эталон  $r_t(d)$ , с которым будут сравниваться результаты.

MAP — метрика средней точности нахождения релевантных документов. Она применяется в том случае, если  $r_t(d)$  принимает бинарные значения, т. е. каждый документ полностью релевантен либо нет. Данная метрика бы-

ла выбрана, так как она непосредственно отражает точность ранжирующей модели.

Precision at  $N$  или точность на  $N$  элементах позволяет оценить долю релевантных документов среди первых  $N$  элементов ранжированного списка и рассчитывается по формуле:

$$p@N = \frac{1}{N} \sum_{k=1}^N r_t(P'(k)), \quad (3.1)$$

где  $P'$  — обратная перестановка, то есть  $P'(k)$  — документ на  $k$ -ом месте после ранжирования.

Average precision at  $N$  позволяет учесть место, на котором оказался документ в ранжированном списке. Гораздо важнее увидеть релевантный документ на 1-ой позиции, чем в конце списка. Формула качества принимает следующий вид:

$$ap@N = \frac{1}{N} \sum_{k=1}^N [r_t(P'(k)) \cdot p@k]. \quad (3.2)$$

Данная величина уже зависит от порядка. Она достигает максимума, если все релевантные документы находятся вверху ранжированного списка.

Mean average precision at  $N$ , в отличии от предыдущих метрик вычисляется сразу для всех запросов множества  $Q$  и является их средним. Пусть  $|Q| = K$ , тогда метрика записывается как:

$$map@N = \frac{1}{K} \sum_{j=1}^K ap@N_j. \quad (3.3)$$

Цель метрики MAP аналогична NDCG, но в отличии от предыдущей группы метрик, рассмотренной выше, NDCG можно использовать и при небинарных значениях  $r_t$ , то есть позволяет учесть что какой-то элемент может быть более или менее релевантным, чем другой. Эта метрика была выбрана, так как она позволяет одновременно учитывать порядок и релевантность документов.

Cumulative gain at  $N$ . В простейшем случае суммируем все значения

релевантностей документов среди  $N$  первых.

$$CG@N = \sum_{k=1}^N r_t(P'(k)). \quad (3.4)$$

Discounted cumulative gain at  $N$ . Модификация позволяет учесть порядок элементов в топе.

$$DCG@N = \sum_{k=1}^N \frac{2^{r_t(P'(k))} - 1}{\log_2(k + 1)}. \quad (3.5)$$

В данной метрике чем более релевантен документ, тем больше числитель. Знаменатель штрафует за позицию документа в списке. Если документ очень релевантен, но занимает низкую позицию, то штраф будет большим, иначе маленьким. Метрика достигает максимума, если все релевантные документы находятся в начале поисковой выдачи.

Normalized discounted cumulative gain at  $N$ . Призвана нормализовать результаты предыдущей метрики.

$$nDCG@N = \frac{DCG@N}{maxDCG@N}, \quad (3.6)$$

где  $maxDCG@N$  — значение метрик при идеальном ранжировании.

Рассмотренные метрики позволяют с высокой точностью проанализировать результаты ранжирования. Формулы (3.1)–(3.6) [9].

В таблицах 3.1 – 3.4 приведены значения метрик рассматриваемых алгоритмов на различных наборах данных. Значения для таблицы 3.1 приведены из работ Тао Квина [11], Чена Ванга [17], Максима Волкова [18]. Значения для таблицы 3.2 приведены из работ Тао Квина [11], Антонио Френо [12], Роберта Буса–Факете [19], Тизиано Папини [20]. Значения для таблицы 3.3 приведены из работ Тизиано Папини [20], Хинуй Даля [21], Минг Тана [22]. Значения для таблицы 3.4 приведены из работ Тао Квина [11], Хай–Тао Ю [23].

Таблица 3.1 – Значение метрик для алгоритма Linear Regression

Датасет	NDCG@3	NDCG@5	NDCG@10	MAP
TD2003	0,3071	0,2984	0,33	0,2409
TD2004	0,3352	0,3257	0,30	0,2078
NP2003	0,6135	0,6423	0,665	0,5644
NP2004	0,5554	0,6135	0,653	0,5142
HP2003	0,5097	0,5463	0,594	0,4968
HP2004	0,5752	0,613	0,646	0,5256
OHSUMED	0,4426	0,4278	0,411	0,422
MQ2007	0,3935	0,4111	0,4288	0,4497
MQ2008	0,4289	0,4773	-	-
Среднее	0,4623	0,4839	0,5034	0,4277

Таблица 3.2 – Значение метрик для алгоритма Ranking SVM

Датасет	NDCG@3	NDCG@5	NDCG@10	MAP
TD2003	0,3441	0,3621	0,346	0,2628
TD2004	0,3467	0,324	0,307	0,2237
NP2003	0,7654	0,7823	0,8	0,6957
NP2004	0,7503	0,7957	0,806	0,6588
HP2003	0,7749	0,7954	0,807	0,7408
HP2004	0,7147	0,7512	0,768	0,6675
OHSUMED	0,4207	0,4164	0,414	0,4334
MQ2007	0,40628	0,41426	0,44386	0,46448
MQ2008	0,42858	0,46954	0,22792	0,46956
MSLR-WEB10K	-	-	0,735	0,498
AOL	0,603	0,647	0,705	-
Yahoo1	-	-	0,803	0,702
Yahoo2	-	-	0,747	-
Istella	-	-	0,808	0,773
WCL2R	0,353	-	0,395	0,432
Среднее	0,5371	0,5758	0,6075	0,5401

Таблица 3.3 – Значение метрик для алгоритма LambdaRank

Датасет	NDCG@3	NDCG@5	NDCG@10	MAP
OHSUMED	0,4942	0,478	0,4503	-
MSLR-WEB10K	0,4498	0,4528	0,60225	0,498
AOL	0,596	0,644	0,7	-
Yahoo1	-	-	0,802	0,7
Istella	-	-	0,81	0,776
Среднее	0,5133	0,5249	0,6729	0,6580

Таблица 3.4 – Значение метрик для алгоритма ListNet

Датасет	NDCG@3	NDCG@5	NDCG@10	MAP
TD2003	0,3365	0,3393	0,348	0,2753
TD2004	0,3573	0,3325	0,317	0,2231
NP2003	0,7579	0,7843	0,801	0,6895
NP2004	0,7587	0,7965	0,812	0,672
HP2003	0,8128	0,8298	0,837	0,7659
HP2004	0,7213	0,7694	0,784	0,6899
OHSUMED	0,4732	0,4432	0,441	0,4457
MQ2007	0,4091	0,417	0,444	0,4652
MQ2008	0,4324	0,4747	0,2303	0,4775
MSLR-WEB10K	0,4324	0,4349	0,45	-
Среднее	0,5492	0,5622	0,5464	0,5227

Из таблиц 3.1, 3.2, 3.4 видно, что алгоритмы Linear Regression, Ranking SVM, ListNet соответственно лучше всего справляются с датасетами NP и HP, то есть способны хорошо находить сайты, имеющие заголовок, схожий с запросом. Хуже всего он справляется с датасетами TD, плохо ранжируют сайты, соответствующие тематике запроса. Согласно таблице 3.3 алгоритм LambdaRank справляется примерно одинаково со всеми датасетами, на которых он тестировался.

В таблицах 3.1–3.4 для алгоритмов Linear Regression, Ranking SVM, LambdaRank, ListNet соответственно можно наблюдать рост значения метрики NDCG с увеличением  $N$ . Из этого можно сделать вывод, что они не лучшим образом ранжируют страницы, находящиеся в самом топе поисковой выдачи. То есть страницу в топ 10 будут расположены ближе к эталону,

чем в топ 3. Это особенно заметно у алгоритма LambdaRank. А вот алгоритм ListNet на некоторых наборах справляется с ранжированием первых 5 элементов, лучше чем с первых 10.

Точность не будет расти постоянно с увеличением  $N$ . Это видно из значений метрики МАР. Она показывает, что точность определения релевантности элементов, находящихся далеко от топа, тоже не велика. Хотя это не является проблемой, потому что порядок элементов, плохо подходящих по запросу, не имеет большого значения и алгоритмы не уделяют большего внимания таким документам.

### 3.3 Сравнение алгоритмов

Для оценки общей эффективности обучения методам ранжирования используется показатель «выигрышное число» [10]. Оно определяется как количество алгоритмов, которое выбранный алгоритм может превзойти на наборе датасетов, или более формально:

$$WN_i(M) = \sum_{j=1}^n \sum_{k=1}^m I_{\{M_i(j) > M_k(j)\}}, \quad (3.7)$$

где  $j$  — номер набора данных,  $n$  — кол-во наборов данных,  $i, k$  — индексы алгоритмов,  $M_i(j)$  — производительность  $i$ -го алгоритма на  $j$ -ом наборе данных и  $I_{\{M_i(j) > M_k(j)\}}$  — индикаторная функция, такая что

$$I_{\{M_i(j) > M_k(j)\}} = \begin{cases} 1 & \text{если оба определены и } M_i(j) > M_k(j), \\ 0 & \text{иначе.} \end{cases}$$

Будем использовать нормализованное «выигрышное число», для удобства интерпретации результатов, определённое как:

$$NWN_i(M) = \frac{WN_i(M)}{IWN_i(M)}, \quad (3.8)$$

где  $IWN_i(M)$  — идеальное «выигрышное число», то есть теоретически наибольшее число, которое было бы у алгоритма в случае, если бы он был наи-

более точным. Определяется как:

$$\text{IWN}_i(M) = \sum_{j=1}^n \sum_{k=1}^m D_{\{M_i(j) > M_k(j)\}}, \quad (3.9)$$

где индикаторная функция имеет значение

$$D_{\{M_i(j) > M_k(j)\}} = \begin{cases} 1 & \text{если оба определены ,} \\ 0 & \text{иначе .} \end{cases}$$

Формулы (3.7)–(3.9) [10].

Исходные данные взяты из таблиц 3.1 – 3.4. Результаты сравнения алгоритмов посчитаны по методике изложенной выше и приведены в таблице 3.5. Данная таблица показывает «выигрышное число» и количество наборов данных, используемых для сравнения, конкретного алгоритма для различных метрик ранжирования.

Таблица 3.5 – Результаты замеров производительности методов обучения ранжированию

Метод	NDCG@3		NDCG@5		NDCG@10		MAP	
	NWN	к.д	NWN	к.д	NWN	к.д.	NWN	к.д.
Linear Regression	0,1053	9	0,2105	9	0	8	0	8
Ranking SVM	0,5000	11	0,4000	10	0,5217	16	0,5500	13
LambdaRank	0,8000	3	0,8000	3	0,6250	5	0,3333	3
ListNet	0,8000	10	0,8000	10	0,8500	10	0,8824	9

### 3.4 Вывод

В рамках выбранных методов лучше всех себя показал ListNet, так как его «выигрышные числа» самые высокие. Чуть хуже показал себя алгоритм LambdaRank, а именно он хуже ранжирует документы находящиеся не в топе поисковой выдачи. Алгоритм Ranking SVM обладает хорошим «выигрышным числом» для метрики MAP, то есть достаточно неплохо ранжирует полный список документов. Алгоритм Linear Regression оказался самым худшим по всем метрикам.

Из таблицы 3.5 можно сделать вывод, что поточенный подход является самым простым, но наименее эффективным. Списочный подход является его противоположностью. Попарный подход является нечто средним. Из сравнения алгоритмов LambdaRank и ListNet видно, что метод с попарным подходом может ранжировать элементы в топе выдачи, от которых и требуется наибольшая точность, также хорошо как и списочный алгоритм, но является проще в разработке и обучении.

# ЗАКЛЮЧЕНИЕ

Цель, которая была поставлена в начале научно-исследовательской работы, была достигнута: рассмотрены алгоритмы машинного обучения, применяемые в поисковых системах.

Решены все поставленные задачи:

- изучены основные понятия алгоритмов обучения ранжированию;
- описаны и классифицированы существующие алгоритмы;
- произведён сравнительный анализ рассмотренных алгоритмов.

В ходе исследования были определены особенности, преимущества и недостатки рассмотренных методов обучения ранжированию.

Списочные алгоритмы превосходят остальные подходы по показателям производительности, но они сложны в разработке и подборе обучающих наборов данных, что подтвердилось в данной работе.

Для алгоритмов с попарным подходом могут быть непосредственно применены существующие методологии классификации, что ускорит и упростит процесс их разработки, а также они обладают хорошими показателями производительности. Однако у данного подхода есть ряд недостатков:

- целью обучения таких алгоритмов является минимизация ошибок в классификации пар документов, а не минимизация ошибок в их ранжировании;
- процесс обучения требует больших вычислительных затрат, поскольку количество пар документов очень велико;
- количество сгенерированных пар документов в значительной степени варьируется от запроса к запросу, что приводит к обучению модели, ориентированной на запросы с большим количеством пар.

Алгоритмы с поточенным подходом являются самыми простыми и наименее производительными.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Google Search Statistics [Электронный ресурс]. — URL: <https://www.internetlivestats.com/google-search-statistics/> (дата обращения: 24.02.2023)
2. Бурков А. Машинное обучение без лишних слов / А. Бурков. — СПб.: Питер. — 2020. — 192 с.
3. Городилов А. Методы машинного обучения для задачи ранжирования / А. Городилов // Труды Третьей Российской конференции молодых ученых по информационному поиску. — 2009. — С. 12–20.
4. Борисовская А. А. Применение методов машинного обучения в задачах ранжирования в поисковых информационных системах. / А. А. Борисовская // Информационные технологии в управлении. — 2021. — С. 1–13.
5. Cossack D. Subset ranking using regression / Cossack D., Zhang T.// In Proceedings of the 19th Annual Conference on Learning Theory (COLT). — 2006. — P. 605–619.
6. Herbrich R. Support vector learning for ordinal regression / Herbrich R., Graepel T., Obermayer K. // In International Conference on Artificial Neural Networks. — 1999. — P. 97–102.
7. Burges C. J. Learning to rank with nonsmooth cost functions / Burges C. J., Ragno R., Le Q. V. // Advances in Neural Information Processing Systems 19. — 2007. — P. 193–200.
8. Z. Cao Learning to rank: From pairwise approach to listwise approach / Z. Cao, T. Qin, T.-Y. Liu et al. // Proceedings of the 24th International Conference on Machine Learning. — 2007. — P. 129–136.
9. Беляков А.В Метрики качества в задачах ранжирования информации / Беляков А.В // Информационные технологии, межвузовский сборник научных трудов. — Рязань: Рязанский государственный радиотехнический университет имени В.Ф. Уткина. — 2019. — С. 36–39.

10. Niek Tax A Cross-Benchmark Comparison of 87 Learning to Rank Methods / Niek Tax, Sander Bockting, Djoerd Hiemstra // Information Processing and Management. —2015. — P. 757–772.
11. Tao Qin LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval / Tao Qin, Tie-Yan Liu, Jun Xu et el. // Information Retrieval. — 2010. — P. 346–374.
12. Antonino Freno Learning to Rank Using Markov Random Fields / Antonino Freno, Tiziano Papini, Michelangelo Diligenti // 2011 10th International Conference on Machine Learning and Applications and Workshops. — 2011. — P. 257–262.
13. Olivier Chapelle Yahoo! Learning to Rank Challenge Overview / Olivier Chapelle, Yi Chang // Proceedings of the learning to rank challenge. — 2011. — P. 1–24.
14. Otavio D. A. Alcantara WCL2R: a benchmark collection for learning to rank research with clickthrough data / Otavio D. A. Alcantara, Alvaro R. Pereira Jr., Humberto M. de Almeida // Journal of Information and Data Management. — 2010. — P. 551–551.
15. Microsoft Learning to Rank Datasets [Электронный ресурс]. — URL: <https://www.microsoft.com/en-us/research/project/msl/> (дата обращения: 05.03.2023)
16. LETOR Dataset [Электронный ресурс]. — URL: <https://istella.ai/data/letor-dataset/> (дата обращения: 05.03.2023)
17. Chieh-Jen Wang Automatic Construction of An Evaluation Dataset from Wisdom of the Crowds for Information Retrieval Applications / Chieh-Jen Wang, Hung-Sheng Huang, Hsin-Hsi Chen // 2012 IEEE International Conference on Systems. — 2012. — P. 490–495.
18. Maksims N. Volkovs Loss-sensitive Training of Probabilistic Conditional Random Fields / Chieh-Jen Wang, Hung-Sheng Huang, Hsin-Hsi Chen // arXiv preprint arXiv:1107.1805. — 2011.

19. Robert Busa-Fekete Tune and mix: Learning to rank using ensembles of calibrated multi-class classifiers / Robert Busa-Fekete, Balazs Kegl, Tamas Elteto et al. // Machine Learning. – 2013. — P. 261–292.
20. Tiziano Papini Learning-to-rank with Prior Knowledge as Global Constraints / Tiziano Papini, Michelangelo Diligenti // WORKSHOP ON COMBINING. – 2012. — P. 35–40.
21. Xinyi Dai U-rank: Utility-oriented Learning to Rank with Implicit Feedback / Xinyi Dai, Jiawei Hou, Qing Liu et al. // Proceedings of the 29th ACM international conference on information and knowledge management. – 2020. — P. 2373–2380.
22. Ming Tan Direct Optimization of Ranking Measures for Learning to Rank Models / Ming Tan, Tian Xia, Lily Guo et al. // Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. – 2013. — P. 856–864.
23. Hai-Tao Yu PT-Ranking: A Benchmarking Platform for Neural Learning-to-Rank / Ming Tan, Tian Xia, Lily Guo et al. // arXiv preprint arXiv:2008.13368. – 2020. — P. 856–864.

# **ПРИЛОЖЕНИЕ А**

## **Презентация научно-исследовательской работы**

Презентация научно-исследовательской работы содержит 15 слайдов, на которых представлено краткое описание научно-исследовательской работы.



Министерство науки и высшего образования  
Российской Федерации Федеральное государственное  
бюджетное образовательное учреждение высшего  
образования «Московский государственный  
технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)  
(МГТУ им. Н.Э. Баумана)

**НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА НА ТЕМУ:**  
**«Применение машинного обучения в поисковых системах»**

Студент: Волков Г.В.  
Руководитель: Шаповалова М.С.

# Цель и задачи работы

Цель работы - изучить алгоритмы машинного обучения, применяемые в поисковых системах

Для достижения поставленной цели следует решить следующие задачи:

- изучить основные понятия алгоритмов обучения ранжированию
- описать и классифицировать существующие алгоритмы
- произвести сравнительный анализ рассмотренных алгоритмов

# Машинное обучение

Машинное обучение — раздел информатики, посвященный созданию алгоритмов, опирающихся на набор данных о каком-либо явлении.

Алгоритмы формируют статистическую модель на основе специально подобранных обучающих данных, которую потом используются для решения практических задач.

Выделяются три основных способа обучения: с учителем, без учителя и с подкреплением.

# Обучение ранжированию

Существует множество методов подбора формул для ранжирования, но один из самых популярных – на основе машинного обучения, а именно обучение ранжированию с учителем. Целью этих методов является подбор ранжирующей модели, которая способна наилучшим образом приблизить и обобщить способ ранжирования на новые данные.

Для получения набора примеров используются ассессоров, которые оцениваю степень релевантности документа запросу.

# Классификация алгоритмов

Существующие алгоритмы обучения ранжированию делятся на три группы по подходу к обучению :

- поточечный
- попарный
- списочный

# Алгоритмы обучения ранжированию

В данной работе рассмотрено несколько популярных алгоритмов:

- Linear Regression (поточечный)
- Ranking SVM (попарный)
- LambdaRank (попарный)
- ListNet (списочный)

На этапе ранжирования методы имеют схожий алгоритм. Для каждого документа вычисляется рейтинг релевантности, который зависит от вектора признаков документа и параметров метода ранжирования. Затем рейтинги сортируются по убыванию и получается ранжированный список документов.

# Linear Regression

Метод обучения на основе регрессии для решения задачи оптимизации метрик DCG.

Для решения проблемы ранжирования можно использовать простой подход, основанный на регрессии.

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n L(f, S_i, \{y_{i,j}\}_j)$$

$$\begin{aligned} L(f, S, \{y_j\}) &= \\ &= \sum_{j=1}^m w(x_j, S)(f(x_j, S) - y_j)^2 + u \max_j w'(x_j, S)(f(x_j, S) - \delta(x_j, S))^2_+ \end{aligned}$$

# Ranking SVM

Ключевая идея алгоритма заключается в использовании метода SVM для попарного сравнения документов на то, какой из них более релевантный.

Теперь рассматривая разность векторов как новые объекты, получаем стандартную постановку SVM алгоритма.

$$\begin{cases} \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^N \xi_i \rightarrow \min_{\omega, \xi}, \\ y_i(\omega, x_i^1 - x_i^2) \leq 1 - \xi_i, \\ \xi_i \geq 0, \end{cases}$$

# LambdaRank

В поточечных и попарных методах ранжирования итоговый функционал при обучении обычно не дифференцируемый. Алгоритм LambdaRank не определяет непрерывный приближенный функционал, вместо этого он определяет градиент функционала на всем списке документов:

$$\frac{\partial L}{\partial s_i} = -\lambda(s_1, y_1, \dots, s_n, y_n)$$

$$\lambda_i = \frac{\partial L}{\partial s_i} = \frac{1}{G_{max}} \sum_j \left( \frac{1}{1 + \exp(s_j - si)} \right) (G(y_j) - G(y_i)) (D(\pi_j) - D(\pi_i))$$

# LambdaRank

$\lambda$  – показывает насколько надо увеличить рейтинг  $i$ -го документа. Для этого надо изменить веса  $\omega$ :

$$\frac{\partial L}{\partial \omega} = \sum_{i=1}^n \frac{\partial s_i}{\partial \omega} \sum_{j \in P_i} \frac{\partial L(s_i, s_j)}{\partial s_i} + \sum_{j=1}^n \frac{\partial s_j}{\partial \omega} \sum_{i \in P_j} \frac{\partial L(s_i, s_j)}{\partial s_j}$$

Таким образом, алгоритм LambdaRank заключается в итерационном пересчете весов:

$$\omega = \omega - \eta \frac{\partial L}{\partial \omega}$$

# ListNet

Цель обучения формализована как минимизация общих потерь в отношении обучающих данных. В данном алгоритме используется вероятностные модели для вычисления функции потерь по списку.

$$P_{z^{(i)}(f_\omega)}(x_j^{(i)}) = \frac{\exp(f_\omega(x_j^{(i)}))}{\sum_{k=1}^{n^{(i)}} \exp(f_\omega(x_k^{(i)}))}$$

$$L(y^{(i)}, z^{(i)}(f_\omega)) = - \sum_{j=1}^{n^{(i)}} P_{y^{(i)}}(x_j^{(i)}) \log(P_{z^{(i)}(f_\omega)}(x_j^{(i)}))$$

# ListNet

Градиент функции потери можно найти по следующей формуле:

$$\begin{aligned}\Delta\omega = \frac{\partial L(y^{(i)}, z^{(i)}(f_\omega))}{\partial \omega} &= - \sum_{j=1}^{n^{(i)}} P_{y^{(i)}}(x_j^{(i)}) \frac{\partial f_\omega(x_j^{(i)})}{\partial \omega} + \\ &+ \frac{1}{\sum_{j=1}^{n^{(i)}} \exp(f_\omega(x_j^{(i)}))} \sum_{j=1}^{n^{(i)}} \exp(f_\omega(x_j^{(i)})) \frac{\partial f_\omega(x_j^{(i)})}{\partial \omega}.\end{aligned}$$

Для минимизации целевой функции используется градиентный спуск

# Критерии сравнения

В качестве критериев сравнения используются основные метрики оценки качества ранжирования: MAP, NDCG

MAP — метрика средней точности нахождения релевантных документов.

NDCG — мера качества

$$map@N = \frac{1}{K} \sum_{j=1}^K ap@N_j$$

$$DCCG@N = \sum_{k=1}^N \frac{2^{r_t(P'(k))} - 1}{\log_2(k + 1)}$$

# Сравнение

Для оценки общей эффективности обучения методам ранжирования используется показатель «выигрышное число». Оно определяется как количество алгоритмов, которое алгоритм может превзойти на наборе датасетов

Метод	NDCG@3		NDCG@5		NDCG@10		MAP	
	NWN	К.д.	NWN	К.д.	NWN	К.д.	NWN	К.д.
Linear Regression	0,1053	9	0,2105	9	0	8	0	8
Ranking SVM	0,5000	11	0,4000	10	0,5217	16	0,5500	13
LambdaRank	0,8000	3	0,8000	3	0,6250	5	0,3333	3
ListNet	0,8000	10	0,8000	10	0,8500	10	0,8824	9

## Заключение

Цель, которая была поставлена в начале научно-исследовательской работы, была достигнута: рассмотрены алгоритмы машинного обучения, применимые в поисковых системах.

Решены все поставленные задачи:

- изучены основные понятия алгоритмов обучения ранжированию
- описаны и классифицированы существующие алгоритмы
- произведён сравнительный анализ рассмотренных алгоритмов