



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.

# РЕФЕРАТ

Расчетно-пояснительная записка 29 с., 1 рис., 11 источников, 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 Linear Regression . . . . .	12
2.2.2 Ranking SVM . . . . .	14
2.2.3 LambdaRank . . . . .	16
2.2.4 ListNet . . . . .	17
<b>3 Сравнение алгоритмов обучения ранжированию</b>	<b>21</b>
3.1 Критерии сравнения . . . . .	21
3.2 Сравнение алгоритмов . . . . .	23
3.3 Вывод . . . . .	24
<b>ЗАКЛЮЧЕНИЕ</b>	<b>26</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	<b>27</b>
<b>ПРИЛОЖЕНИЕ А Презентация научно-исследовательской работы</b>	<b>29</b>

# ВВЕДЕНИЕ

Машинное обучение — термин, обозначающий набор методов и инструментов, которые помогают компьютерам самостоятельно обучаться и адаптироваться. Алгоритмы машинного обучения помогают ИИ учиться, не будучи явно запрограммированным на выполнение желаемого действия, обрабатывая 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. Большинство людей ежедневно совершают множество 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.

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.

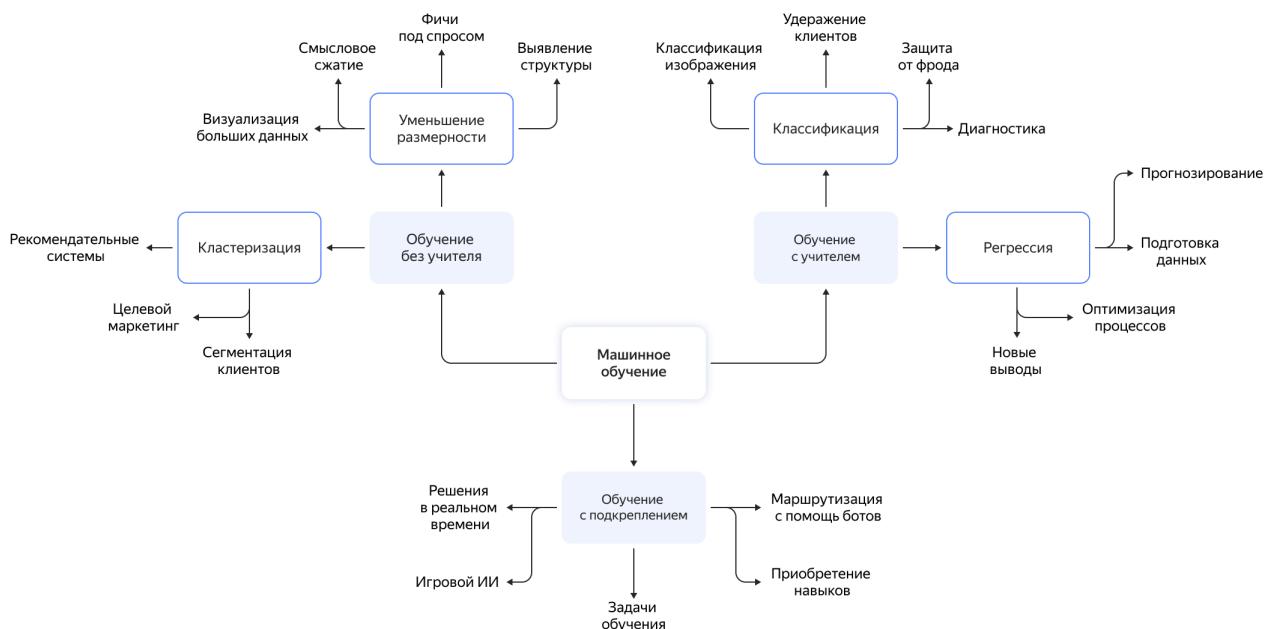


Рисунок 1.1 – Способы обучения

В обучении с учителем набор данных организован как коллекция размеченных образцов вида «известный вход — известный выход», называемый обучающей выборкой. Существует некоторая зависимость между входом и выходом, но она неизвестна. При обучении создаётся модель, отражающая эту зависимость, способная для любых входных данных выдать достаточно точный результат. Каждый вход является вектором признаков, описывающих некоторые характеристики образца. Выход может быть элементом конечного

множества, вещественным числом или более сложной структурой. В основном используется для задач регрессии и классификации [2].

Обучение без учителя — класс задач машинного обучения, в которых известны только описания множества объектов и требуется обнаружить внутренние взаимосвязи. На вход 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. Используются для задач кластеризации, уменьшения размерности и выявления аномалий [2].

Обучение с подкреплением — это раздел машинного обучения, в котором предполагается, что машина «живет» в определенном окружении и способна воспринимать его как вектор характеристик. Идея заключается в том, что компьютер взаимодействует со средой, параллельно обучаясь получает вознаграждение за выполнение действий т. е. 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 удовлетворяющие его информационную потребность. При этом основными критериями качества результатов информационного поиска являются полнота, точность и оперативность. Сайты в поисковой выдаче сортируются в соответствии с факторами, используемыми поисковым движком т. е. проходят процедуру ранжирования.

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 характеризу-

ются 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 по подходу к обучению [4]: поточечный, попарный и списочный.

Поточечный подход — частный случай задачи регрессии или классификации, если множество оценок конечно, в ходе ее решения возможно использовать метод градиентного спуска, опорных векторов и т. д непосредственно. На практике этот подход дает не очень качественный результат, так как каждый документ ранжируется независимо от других. В качестве примеров для обучения используют пары «признаки — значение релевантности». Данная задача может быть аппроксимирована задачей регрессии — учитывая единственную пару «запрос — документ», спрогнозировать ее оценку. Точечный подход направлен на изучение функции, предсказывающей реальное значение или порядковый номер документа, используя функцию потерь. Окончательное ранжирование достигается путем простой сортировки списка результатов по оценке документа, выданной обученной моделью [5].

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 релевантнее. Классификатор должен принимать два документа в качестве входных данных, его цель состоит в том, чтобы минимизировать функцию потерь. Недостатком также является невозможность учета всех документов запроса. Целью обучения таких алгоритмов является минимизация ошибок в классификации пар документов, а не минимизация ошибок в ранжировании документов, поэтому процесс обучения требует больших вычислительных затрат, а количество сгенерированных пар документов в значительной степени варьируется от запроса к запросу, что приводит к обучению модели, ориентированной на запросы с большим количеством пар. При ранжировании алгоритмы получают пару документов на вход и определяет их порядок относительно друг друга [5].

Списочный подход заключается в построении модели, на вход которой поступают сразу все документы, соответствующие запросу, а на выходе получается их перестановка. Подгонка параметров модели осуществляется для прямой максимизации одной метрик ранжирования. Но это часто 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 ранжирования, списочные методы решают исходную постановку задачи. Являются одними из самых лучших методов на данный момент, но сложны в разработке и подборе обучающих наборов данных. При таком подходе алгоритмы непосредственно просматривают весь список документов и подбирают для него оптимальный порядок [5].

## 2.2 Алгоритмы обучения ранжированию

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

Перед описанием алгоритмов стоит 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 на сколько документ  $d_{i,j}$  релевантен к запросу  $q_i$ ;
- $y_i$  — вектор меток  $y_{i,j}$ .

На этапе ранжирования методы имеют схожий алгоритм. Для каждого документа  $d_{i,j}$  вычисляется рейтинг релевантности, который зависит от вектора признаков документа  $x_{i,j}$  и параметров метода ранжирования  $\omega$ . Затем

рейтинги сортируются по убыванию и получается ранжированный список документов.

При ранжировании 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 информативно отражают 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, удовлетворяющих 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 соответствующий запросу пользователю. 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 Linear Regression

Метод обучения на основе регрессии используется для решения задачи оптимизации метрик DCG [6]. Пусть  $F$  — функциональное пространство, содержащее функции  $X \times S \rightarrow F$ , 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  $\{y_{i,j}\}_j = \{y_{i,1}, \dots, y_{i,m}\}$ . 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 в общем случае. 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 проводиться так, чтобы оказаться как можно ближе к обучающим примерам [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 меток.

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:

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

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

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

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  $m$  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 вычислительно эффективным даже при большом  $m$ . Формулы (2.1) и (2.2) [6].

### 2.2.2 Ranking SVM

Ключевая идея алгоритма, предложенного Р. Хербрихом [7], заключается в использовании метода SVM для попарного сравнения документов на то, какой из элементов более релевантный.

Метод опорных векторов — алгоритм обучения с учителем, используемый для задач классификации и регрессии. Суть метода заключается в построении гиперплоскости, 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 взаимопритяжения 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 поставленная 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:

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

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

$$y_i(w \cdot x - b) \geq 1. \tag{2.5}$$

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) [7].

### 2.2.3 LambdaRank

В поточечных и попарных методах ранжирования итоговый функционал при обучении обычно не дифференцируется, так как он зависит от порядка элементов. В рассматриваемом алгоритме, описанном Бургес [8], вместо исходного функционала рассматривается непрерывный аналог, который легко оптимизировать. Рассмотрим работу метода для одного поискового запроса. Алгоритм LambdaRank не определяет непрерывный приближенный функционал  $L$ , вместо этого он находит 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 эффективности алгоритма — увеличение градиента документов на первых позициях, то есть нужно сделать более значимыми перестановки с первыми элементами. 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 ка-

честв, однако в стандартном варианте  $\lambda$  вычисляется по метрике NDCG:

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

где  $G, D$  — функции преобразования релевантности документа в его рейтинг и порядковый номера 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  $P_i, P_i$  — множество пар документов с индексами  $j$  и  $i$ , для которых 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) [8].

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 ListNet

Впервые алгоритм был описан З. КАО [9]. 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. 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  $n$  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 перестановки могут иметь разную вероятность, рассчитанную на основе функции ранжирования. 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  $\phi(s_{\pi(j)})$  — оценка объекта в позиции  $j$  перестановки  $\pi$ ,  $\phi$  — возрастающая и строго положительная функция.

Для любого списка ранжирования, основанного на данной функции, 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, 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 два 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 нейронной сети  $\omega$  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:

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

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) [9]

### 3 Сравнение алгоритмов обучения ранжированию

#### 3.1 Критерии сравнения

Наиболее распространенный подход к оценке эффективности ранжирующего отображения основан на проверке критериев точности и полноты поиска, так как они и являются основными требованиями к поисковым системам [10]. 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 больше документ  $d$  подходит к запросу  $q$ . Для оценки качества ранжирования нужен эталон  $r_t(d)$ , с которым будут сравниваться результаты.

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

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

В отличии от предыдущей группы метрик рассмотренные ниже NDCG можно использовать и при небинарных значениях  $r_t$ . Эта метрика была выбрана, так как она позволяет 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 зависит от позиции документа, он 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 топе 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, являясь довольно простыми для восприятия и позволяют с высокой точностью проанализировать результаты ранжирования. Формулы (3.1)–(3.6) [10].

## 3.2 Сравнение алгоритмов

Сравнение алгоритмов производилось по наборам данных таких коллекций как: AOL, LETOR 2.0, LETOR 3.0, LETOR 4.0, MSLR, WCL2R, Yahoo! Learning to Rank Challenge, Yandex Internet Mathematics 2009 contest. В самих датасетах содержатся не документы, а векторы их признаков. Таким образом, любая разница в производительности ранжирования обусловлена алгоритмом ранжирования, а не используемыми функциями.

Для оценки общей эффективности обучения методам ранжирования используется показатель «выигрышное число» [11]. 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 удоб-

ства 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 наиболее точным. Определяется как:

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

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) [11].

Результаты замеров метрик взяты из исследования Ника Такса [11] и приведены приведены в таблице 3.1. Данная таблица показывает «выигрышное число» и количество наборов данных, используемых для сравнения, конкретного алгоритма для различных метрик ранжирования.

Таблица 3.1 – 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	0.0754	9	0.1099	9	0.0829	8	0.0650	8
Ranking SVM	0.3014	12	0.3613	11	0.4496	17	0.3400	13
LambdaRank	0.2000	1	0.2000	1	0.5714	2	-	-
ListNet	0.4480	12	0.4911	12	0.5982	12	0.4504	12

### 3.3 Вывод

В рамках выбранных методов лучше всех себя показал ListNet по всем метрикам, так как его «выигрышные числа» самые высокие. Хотя по срав-

нению с остальными алгоритмами из [11] он является довольно средним, поскольку все его NWN колеблется рядом с 0.5, то есть превосходит примерно половину методов, с которыми он сравнивался по всем метрикам . Linear Regression же проигрывает почти девяноста процентам всех алгоритмов, с которыми его сравнивали. Важно отметить, что данная метрика является нормированной.

Алгоритмы Ranking SVM, LambdaRank и ListNet лучше ранжируют большой объём документов, но плохо справляются с сайтами в самом топе поисковой выдачи в отличие от остальных алгоритмов в исследовании [11]. Данный вывод был сделан, поскольку метрика NDCG будет увеличиваться пропорционально учитываемому количеству сайтов. Это особенно заметно у LambdaRank. Также эти методы обладают не лучшей точностью поисковой выдачи. В итоге можно понять, что выбранные алгоритмы являются средними по показателям, поскольку практически все их метрики меньше 0.5. При этом они довольно сбалансированы, так как нет большой просадки в значениях «выигрышных чисел».

# 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/>
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. Li H. Learning to Rank for Information Retrieval and Natural Language Processing. — Morgan and Claypool Publishers, 2011.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.

11. 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.

# **ПРИЛОЖЕНИЕ А**

## **Презентация научно-исследовательской работы**

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