МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ   
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Допущен к защите

Заведующий кафедрой ПМИ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Разова

**Разработка системы ссылочного ранжирования**

Курсовой проект по дисциплине  
«Теория алгоритмов»

Выполнил студент группы ФИб-3301-51-00     / Д. О. Ощепков/

Руководитель к.т.н.,

доцент кафедры ПМИ     / А. Г. Татаринова/

Работа защищена с оценкой     \_\_\_\_.\_\_\_\_.2024 г.

Члены комиссии:     / /

    / /

Киров 2024

Содержание

[Введение 3](#_Toc168439809)

[1 Разбор алгоритмов ссылочного ранжирования 5](#_Toc168439810)

[1.1 Постановка задачи 5](#_Toc168439811)

[1.2 Метрики оценки качества 6](#_Toc168439812)

[1.2.1 Pairwise orderedness 6](#_Toc168439813)

[1.2.2 Precision 7](#_Toc168439814)

[1.2.3 Recall 7](#_Toc168439815)

[1.2.4 метрика и ее частный случай метрика 8](#_Toc168439816)

[1.3 Обзор рассматриваемых алгоритмов 8](#_Toc168439817)

[1.3.1 Pagerank 8](#_Toc168439818)

[1.3.2 HITS 13](#_Toc168439819)

[1.4 Выводы по разделу 18](#_Toc168439820)

[2 Применение алгоритмов на практике 19](#_Toc168439821)

[2.1 Google Web Graph 19](#_Toc168439822)

[2.1.1 Pagerank 20](#_Toc168439823)

[2.1.2 HITS 21](#_Toc168439824)

[2.2 MovieLens 20M Dataset 22](#_Toc168439825)

[2.2.1 Pagerank 24](#_Toc168439826)

[2.2.2 HITS 27](#_Toc168439827)

[2.2.3 Сравнение с NetworkX 29](#_Toc168439828)

[2.3 Выводы по разделу 32](#_Toc168439829)

[Заключение 34](#_Toc168439830)

[Библиографический список 36](#_Toc168439831)

[Приложения 37](#_Toc168439832)

[Приложение А. Листинг программы 37](#_Toc168439833)

# Введение

Ранжирование – процесс упорядочивания элементов по значимости, релевантности или другим критериям. Самый простой пример ранжирования – это сортировка.

Однако, не всегда есть явный критерий, который позволит упорядочить множество объектов по наибольшей значимости для какой-либо задачи. В этом курсовом проекте мы рассмотрим способы присвоения ранга объектам, которые имеют сложные отношения между друг другом.

Рассмотрим более конкретный пример. Мы можем рассматривать статическую сеть, состоящую из HTML-страниц вместе с гиперссылками между ними, как ориентированный граф, в котором каждая веб-страница является узлом, а каждая гиперссылка — направленным ребром. [1, стр. 389] Требуется определить значимость каждой страницы для графа, образованным таким образом.

Если сформулировать задачу для абстрактных взаимосвязанных объектов, то ссылочное ранжирования заключается в том, чтобы определить степень важности каждого элемента на основе структуры графа, образованного самими элементами и связями между ними.

Алгоритмы, решающие задачи выше, часто применяются в поисковых системах для ранжирования результатов поиска [2], в социальных сетях для определения влиятельных пользователей или контента [3], в анализе сетей для выявления ключевых узлов [4].

Первым алгоритмом ссылочного ранжирования был Pagerank и его вариации, такие как personalized pagerank, weighted pagerank, personalized weighted pagerank. Они используют итеративный процесс для расчета весов страницы, учитывая количество и качество входящих ссылок [2]. Однако, эти алгоритмы оказались уязвимы к манипуляциям, так как спамеры научились искусственно наращивать ссылочную массу сайтов.

В ответ на эту проблему был разработан алгоритм TrustRank. В отличие от PageRank, который рассматривает все ссылки равнозначно, TrustRank отдает предпочтение ссылкам с доверенных сайтов.

Помимо PageRank и TrustRank, существует еще один важный алгоритм ссылочного ранжирования. Используется для определения важности веб-страниц на основе их характеристик как "хабов" и "авторитетов". Хабы (Hubs) ­ это страницы, которые содержат много ссылок на другие страницы. Чем больше ссылок у страницы, тем она более "хабовая". Авторитеты (Authorities) ­ это страницы, на которые много ссылается. Такие страницы считаются важными и качественными и обладают высоким "авторитетом". [5, стр. 607]

В курсовом проекте были реализованы Pagerank и HITS.

Проблема исследования заключается в отсутствии четких рекомендаций или наставлений для выбора конкретного алгоритма ссылочного ранжирования классификации в конкретной задаче. Каждый из методов обладает своими сильными и слабыми сторонами, а также применим к определенным условиям и задачам.

Для решения данной проблемы необходимо провести сравнительный анализ различных алгоритмов ссылочного ранжирования в различных сценариях и задачах.

Для этого в рамках курсового проекта предлагается исследовать возможности алгоритмов ссылочного ранжирования при решении задачи поиска рекомендаций в зависимости от пользовательских предпочтений.

Это позволит определить, какие методы лучше справляются с определенными типами данных, объемом данных, структурой сети и характеристиками узлов. Кроме того, такой анализ может учитывать требования к точности ранжирования, скорости работы алгоритма

Целью курсового проекта является формирование рекомендации по выбору алгоритма ссылочного ранжирования.

Для достижения цели необходимо решить следующие задачи:

1. Обзор литературы. Изучить существующие научные статьи, книги и исследования, посвященные метрическим методам классификации. Выявить основные методы и их характеристики, а также рассмотреть примеры исследований, где эти методы применялись в различных задачах.
2. Описание и анализ методов ранжирования
3. Подготовка набора данных. Собрать или использовать доступные наборы данных, соответствующие различным предметным областям. Обеспечить разнообразие данных в терминах признаков, классов, объема выборки и сложности задачи.
4. Выбор метрик для сравнения алгоритмов.
5. Реализация и применение каждого из методов для решения задач ранжирования на различных наборах данных.
6. Анализ результатов и сравнение эффективности каждого метода на различных наборах данных. Определить подходящие метрики оценки качества классификации и провести серию экспериментов, сравнивая различные метрические методы классификации на подготовленных наборах данных. Сравнивать производительность методов в терминах точности, полноты, F-меры.
7. Сделать выводы о применимости каждого метода в различных сценариях и задачах ранжирования.

Курсовой проект включает два раздела. В первом разделе проведен обзор литературы и алгоритмов по предметной области. Во втором разделе был проведен анализ алгоритмов ранжирования.

# Разбор алгоритмов ссылочного ранжирования

## Постановка задачи

Дано:

* Множество объектов , где - количество объектов.
* Ориентированный граф , представляющий связи между объектами. Ребро означает, что объект "ссылается" на объект .

Алгоритм должен присвоить каждому объекту числовой ранг , отражающий его относительную "важность" в контексте графа .

## Метрики оценки качества

В дальнейшем нам понадобится следующее:

|  |  |  |  |
| --- | --- | --- | --- |
| Category | |  | |
| YES | NO |
|  | YES |  |  |
| NO |  |  |

Таблица 1.2‑1. The Contingency Table for Category . [7, p. 38]

*Таблиц*а 1.2‑2*.* адаптирована под курсовой проект

Здесь – Истинно положительный (True Positive), – Ложно Положительный (False Positive), – Ложно Отрицательный (False Negative), – Истинно Отрицательный (True Negative).

Метрики ниже взяты из статьи [6, 578]

Формализуем человеческую оценку в функциональный вид

Где , а – множество вершин графа страниц

Хорошая страница или плохая можно определить по-разному на практике. Этой оценкой может быть, например, то, что значение, даваемое экспертом, выше, чем среднее значение по всем оценкам страниц.

### Pairwise orderedness

Для данной метрики понадобиться вспомогательная функция, которая будет сигнализировать, если плохая страница получила равный или более высокий рейтинг.

– функция, возвращающая ранг вершины согласно алгоритму, а – это вершины графа.

Теперь можем сформулировать pairwise orderedness

Если pairord равна 1, не существует случаев, когда T неверно оценил пару. Наоборот, если pairord равна нулю, то T неверно оценил все пары.

### Precision

Показывает долю высоко оцененных вершин, которые действительно являются хорошими, от всех высоко оцененных вершин.

Precision показывает способность модели точно предсказывать положительный класс. Если вы хотите избегать ложных срабатываний и быть уверенными в том, что предсказанный положительный класс действительно является положительным, вам нужно стремиться к высокому значению precision.

Формулу можно преобразовать в следующее:

### Recall

Показывает долю хороших вершин, которые высоко оценены алгоритмом, от всех хороших вершин.

Recall показывает способность модели обнаруживать все положительные примеры. Если важно, чтобы модель обнаруживала как можно больше положительных случаев, даже если это приводит к большему количеству ложных срабатываний, то стоит стремиться к высокому значению recall.

Формулу можно преобразовать в следующее.

### метрика и ее частный случай метрика

Является объединением Precision и Recall, показывает приоритет одной метрики над другой. При метрики учитываются в равной степени.

-мера полезна если данные несбалансированны, так как она учитывает и точность, и полноту, что позволяет получить общую оценку производительности модели, не учитывая смещение в сторону преобладающего класса.

## Обзор рассматриваемых алгоритмов

### Pagerank

Представьте себе случайного пользователя, который начинает с веб-страницы (узла веб-графа) и выполняет случайное блуждание по сети следующим образом. На каждом временном шаге пользователь переходит с текущей страницы A на случайно выбранную веб-страницу, на которую A ссылается гиперссылкой. На рисунке 1.3.1‑1 показан пользователь в узле A, из которого есть три гиперссылки на узлы B, C и D; на следующем временном шаге пользователь переходит к одному из этих трех узлов с равными вероятностями 1/3.

По мере того как пользователь переходит от узла к узлу в этом случайном блуждании, он посещает некоторые узлы чаще, чем другие; интуитивно понятно, что это узлы, на которые ссылается множество других часто посещаемых узлов. Идея, лежащая в основе PageRank, заключается в том, что страницы, которые посещаются чаще в этом блуждании, являются более важными [1, стр. 424].

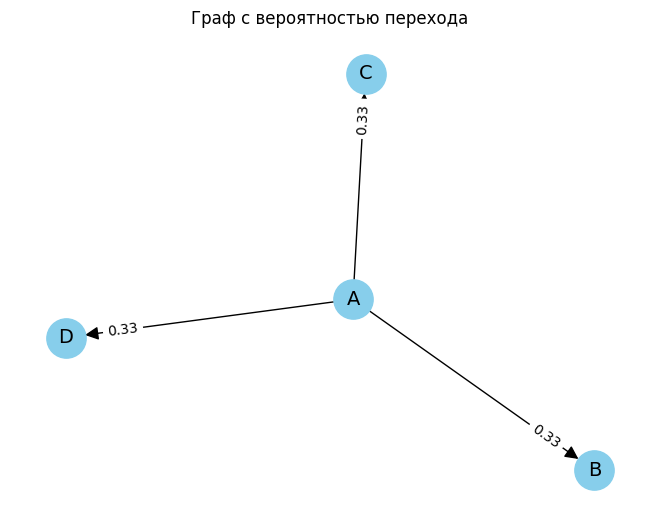


Рисунок 1.3.1‑2 Иллюстрация для объяснения идеи Pagerank

Входные данные:

* - матрица переходов графа, где - вероятность перехода от узла к узлу .
* - вектор персонализации, отражающий предпочтения пользователя к определенным страницам.
* - коэффициент, определяющий вес матрицы переходов.
* - коэффициент, определяющий вес вектора персонализации.
* - коэффициент, необходимый для чтобы каждая веб-страница имела самоцикл с ненулевой вероятностью, а также каждая пара веб-страниц была связана.
* - количество итераций.
* Алгоритм:

1. Инициализация:  
   Создаем вектор PageRank и устанавливаем начальные значения:
2. Итеративный процесс:

Для от 1 до :

Выходные данные:

* вектор PageRank, где представляет рейтинг узла .

Замечания:

* Коэффициенты , и должны удовлетворять условию .
* Вектор персонализации должен быть нормализован, так что сумма его элементов равна 1.
* Количество итераций iterations влияет на точность результатов.

Использование разреженных матриц для представления графа значительно улучшает эффективность алгоритма PageRank, особенно для больших графов с небольшим количеством связей.

Временная сложность:

Цикл выполняется iterations раз.

Умножение разреженной матрицы на вектор: - поскольку мы перебираем только ненулевые элементы матрицы.

Умножение вектора на вектор: .

Сложение векторов:

Итого: Временная сложность алгоритма PageRank с использованием разреженных матриц составляет . Если количество связей намного больше, чем количество вершин, то можно упростить до .

Приведем простой пример.

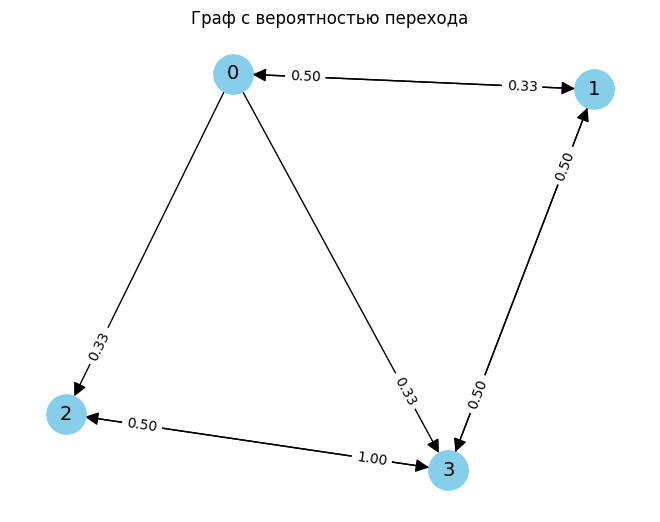


Рисунок 1.3.1‑3 Демонстрационный граф

Сделаем несколько итераций алгоритма с параметрами a=0.85, b=0 (нет персонализации), c=0.15. Матрица переходов имеет вид (каждое значение равно , если существует связь)

Инициализируем вектор .

Итерация 1:

Итерация 2:

Итерация 3:

Получаем, что узел 3 является наиболее важным в этом графе, что легко интерпретировать, потому что к нему идет больше всех сильных связей.

### HITS

Предназначен для поиска по широкой теме. Для любого запроса мы вычисляем два ранжированных списка результатов вместо одного. Ранжирование одного списка определяется оценками концентрации, а другого — оценками авторитетности.

Этот подход основан на особом понимании процесса создания веб-страниц, а именно на том, что существует два основных типа веб-страниц, полезных в качестве результатов поиска по широким темам. Под поиском по широкой теме мы подразумеваем информационный запрос, такой как «Я хочу узнать о лейкемии». Существуют авторитетные источники информации по этой теме; в данном случае страницей такого рода была бы страница Национального института рака о лейкемии. Мы будем называть такие страницы авторитетами; в вычислениях, которые мы собираемся описать, это страницы, которые будут иметь высокие показатели авторитетности [1, стр. 433].

Входные данные:

* : Разреженная матрица, представляющая ориентированный граф, где ненулевое, если есть ребро от узла к узлу .
* iterations: Число итераций алгоритма.

Алгоритм:

1. Инициализация:
   * Создать векторы hubs и authorities размера n (число узлов в графе).
2. Итеративный процесс:
   * Повторить iterations раз:
     + Обновление authorities:
       - Для каждого узла j:
     + Обновление hubs:
       - Для каждого узла j:
     + Нормализация:
       - Разделить все элементы hubs и authorities на их евклидову норму.

Где ­ мера авторитетности узла , — мера хабности узла . и ­ обновленные значения и соответственно [8, стр. 23].

Выходные данные:

* hubs: Вектор, содержащий hub оценки для каждого узла.
* authorities: Вектор, содержащий authority оценки для каждого узла.

Замечания:

* Использование разреженных матриц для представления графа значительно улучшает эффективность алгоритма HITS, особенно для больших графов с небольшим количеством связей.
* Нормализация предотвращает экспоненциальный рост значений hubs и authorities в процессе итераций.

Временная сложность:

Цикл выполняется iterations раз.

Умножение разреженной матрицы на вектор: - поскольку мы перебираем только ненулевые элементы матрицы.

Нормализация: – нахождение суммы вектора и деление каждого элемента на эту сумму.

Итого: Временная сложность алгоритма PageRank с использованием разреженных матриц составляет . Если количество связей намного больше, чем количество вершин, то можно упростить до .

Приведем простой пример на графе из рисунка 1.3.2‑1.

Сделаем несколько итераций алгоритма. Матрица переходов имеет вид (каждое значение равно , если существует связь)

Инициализируем вектор .

Для упрощения введем оператор и функцию , которая считает суммирует все строки матрицы в вектор. Тогда

.

Итерация 1:

Итерация 2:

Итерация 3:

Наибольший авторитет также имеет узел 3.

Существует еще один подход нахождения hits, который вытекает следующей теоремы.

Теорема 3.2 a – это главный собственный вектор матрицы , а h – главный собственный вектор матрицы [9, стр. 11].

Согласно выводу выше достаточно найти главный собственный вектор – это будет a, затем по формуле итеративного нахождения hubness найти h.

Существует много способов найти собственный вектор матрицы, давайте воспользуемся сингулярным разложением матрицы для этой цели. Любую матрицу M можно разложить следующим образом:

При этом – это диагональная матрица, по диагонали которой расположены собственные числа. При этом одно из свойств разложения, говорит, что колонки матрицы V – это собственные вектора матрицы Остается только найти главный собственный вектор. Стоит отметить, что существуют библиотеки, которые позволяют найти, только один собственный вектор при сингулярном разложении, что довольно эффективно по времени.

Приведем пример для матрицы M.

Разложим матрицу с помощью математического пакета.

Главный собственный вектор – это вектор, соответствующий максимальному собственному значению (в нашем случае столбец 3). Получаем

## Выводы по разделу

В данной главе были рассмотрены теоретические основы ссылочного ранжирования и два его ключевых алгоритма: PageRank и HITS.

PageRank рассматривает всю сеть как единое целое и оценивает важность страницы, исходя из количества и "веса" входящих ссылок, интерпретируя ссылку как "голос" одной страницы за другую. Метод использует итеративный алгоритм, базирующийся на матрице переходов, для вычисления стационарного распределения PageRank в сети. Важным преимуществом PageRank является его устойчивость к случайным изменениям в структуре сети и способность выделять авторитетные узлы в масштабных графах.

Алгоритм HITS, в отличие от PageRank, ориентирован на контекст конкретного запроса и анализирует только подграф страниц, релевантных этому запросу. Он вычисляет две метрики: авторитетность (authority) и концентрацию (hub) страницы. Авторитетность отражает важность информации на самой странице, а концентрация – ее роль как "путеводителя" по другим важным ресурсам. HITS позволяет эффективно находить как авторитетные страницы, так и узлы-хабы, релевантные заданной теме.

# Применение алгоритмов на практике

Для реализации итеративного HITS и Pagerank использовался С++ версии 20, так как в ходе работы выяснилось, что ручные реализации на Python работали медленно по сравнению с готовыми решениями.

Для реализации HITS с помощью сингулярного разложения использовался Python версии 3.12. Выбор был обусловлен высокопроизводительными алгоритмами из библиотеки scipy.

В работе были использованы следующие библиотеки языка Python:

* NumPy обеспечивает эффективную работу с матрицами смежности графов.
* SciPy предоставляет алгоритмы линейной алгебры
* Pandas упрощает работу с таблицами
* Matplotlib позволяет визуализировать результаты работы алгоритмов, например

В работе были использованы следующие библиотеки языка C++:

* Eigen обеспечивает эффективную работу с матрицами, что важно для представления графов и выполнения операций линейной алгебры
* RapidCSV упрощает чтение и запись данных в формате CSV, позволяя легко загружать и сохранять информацию о графах и результатах работы алгоритмов

Результаты, рассмотренные далее, сравнились с известной библиотекой NetworkX[[1]](#footnote-1).

## Google Web Graph[[2]](#footnote-2)

Этот набор данных представляет собой граф веб-страниц, где узлы соответствуют веб-страницам, а ориентированные ребра - гиперссылкам между ними. Данные были опубликованы Google в 2002 году в рамках конкурса Google Programming Contest.

Узлы: 875 713

Рёбра: 5 105 039

### Pagerank

На рисунке 2.2.1‑1 представлена зависимость времени выполнения PageRank от размера графа (в частях от полного набора данных).

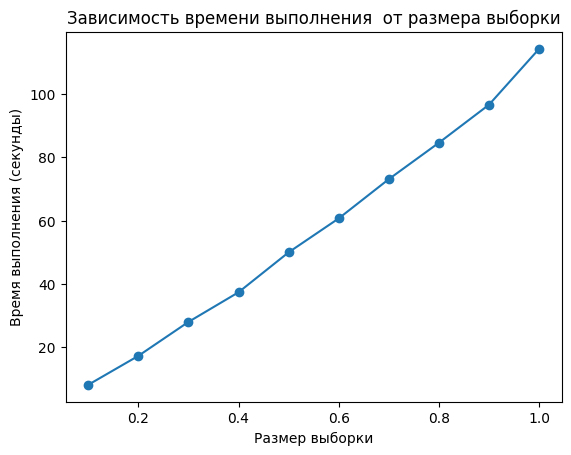


Рисунок 2.2.1‑1 Время выполнения 100 итераций pagerank в зависимости от размера датасета Google Web Graph.

|  |  |  |
| --- | --- | --- |
| Часть от целого датасета | Время выполнения | Время считывания |
| 0,1 | 7,60 | 1,05 |
| 0,2 | 18,20 | 1,93 |
| 0,3 | 29,20 | 2,66 |
| 0,4 | 42,50 | 3,35 |
| 0,5 | 59,31 | 4,28 |
| 0,6 | 66,06 | 5,22 |
| 0,7 | 75,62 | 5,70 |
| 0,8 | 84,39 | 6,37 |
| 0,9 | 96,59 | 7,10 |
| 1,0 | 109,33 | 7,87 |

Таблица 2.2‑1 Время выполнения Pagerank и преобразования данных для него на датасете Google Web Graph.

Эта таблица подтверждает линейную зависимость времени выполнения алгоритма в зависимости от m, что согласуется с теоретической оценкой.

### HITS

На рисунке представлена зависимость времени выполнения PageRank от размера графа (в частях от полного набора данных).

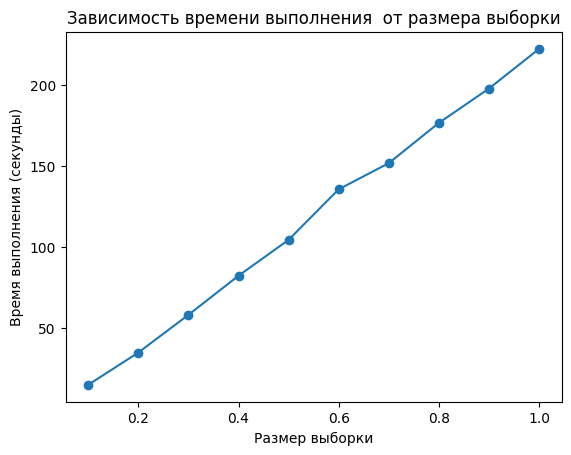


Рисунок 2.2.2‑1 Время выполнения 100 итераций HITS в зависимости от размера датасета Google Web Graph.

|  |  |  |
| --- | --- | --- |
| Часть от целого датасета | Время выполнения | Время считывания |
| 0,1 | 14,84 | 1,06 |
| 0,2 | 34,76 | 2,01 |
| 0,3 | 58,06 | 2,82 |
| 0,4 | 82,25 | 3,68 |
| 0,5 | 104,43 | 4,33 |
| 0,6 | 135,66 | 4,94 |
| 0,7 | 151,82 | 5,99 |
| 0,8 | 176,71 | 6,76 |
| 0,9 | 197,91 | 7,59 |
| 1,0 | 222,45 | 8,36 |

Таблица 2.2‑2 Время выполнения HITS и преобразования данных для него на датасете Google Web Graph.

Эта таблица подтверждает линейную зависимость времени выполнения алгоритма в зависимости от m, что согласуется с теоретической оценкой.

## MovieLens 20M Dataset[[3]](#footnote-3)

Набор данных описывает рейтинги и теги, присвоенные пользователями сервиса рекомендаций фильмов MovieLens. Он содержит 20 000 263 рейтинга и 465 564 тега, относящихся к 27 278 фильмам. Эти данные были созданы 138 493 пользователями в период с 9 января 1995 года по 31 марта 2015 года. Сам набор данных был сгенерирован 17 октября 2016 года.

Пользователи для включения в набор данных выбирались случайным образом. Все выбранные пользователи оценили не менее 20 фильмов.

Содержание:

Демографическая информация не включена. Каждый пользователь представлен идентификатором, и никакой другой информации не предоставляется.

Данные содержатся в шести файлах:

tag.csv содержит теги, примененные пользователями к фильмам:

* userId
* movieId
* tag
* timestamp

rating.csv содержит рейтинги фильмов, выставленные пользователями:

* userId
* movieId
* rating
* timestamp

movie.csv содержит информацию о фильмах:

* movieId
* title
* genres

link.csv содержит идентификаторы, которые можно использовать для связи с другими источниками:

* movieId
* imdbId
* tmbdId

genome\_scores.csv содержит данные о релевантности тегов к фильмам:

* movieId
* tagId
* relevance

genome\_tags.csv содержит описания тегов:

* tagId
* tag

В рамках исследования рекомендательных систем был построен двунаправленный граф на основе данных о рейтингах пользователей. Вершинами графа являются пользователи сервиса, а фильмами - объекты рекомендаций. Ребро между пользователем и фильмом проводится, если пользователь поставил оценку этому фильму. Вес ребра соответствует значению оценки.

### Pagerank

На рисунке 2.1.1‑1 представлена зависимость времени выполнения PageRank от размера графа (в частях от полного набора данных).

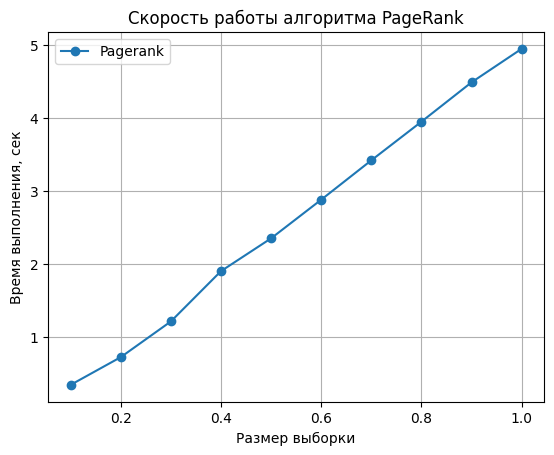


Рисунок 2.1.1‑1 Время выполнения 100 итераций pagerank в зависимости от размера датасета MovieLens.

|  |  |  |
| --- | --- | --- |
| Часть от целого датасета | Время выполнения | Время считывания |
| 0,1 | 0,34 | 2,02 |
| 0,2 | 0,72 | 4,00 |
| 0,3 | 1,21 | 6,10 |
| 0,4 | 1,9 | 7,93 |
| 0,5 | 2,35 | 10,35 |
| 0,6 | 2,88 | 12,09 |
| 0,7 | 3,42 | 13,76 |
| 0,8 | 3,95 | 15,91 |
| 0,9 | 4,49 | 18,44 |
| 1 | 4,95 | 19,79 |

Таблица 2.1‑1 Время выполнения Pagerank и преобразования данных для него на датасете MovieLens.

Эта таблица подтверждает линейную зависимость времени выполнения алгоритма в зависимости от m, что согласуется с теоретической оценкой.

На данном датасете возможно оценить качество полученных результатов. Воспользуемся метриками из главы 1.2. Для этого нужно определить функцию Пусть она будет иметь вид

Тогда усреднено для всех пользователей получаем следующие значения:

Accuracy: 0.5340

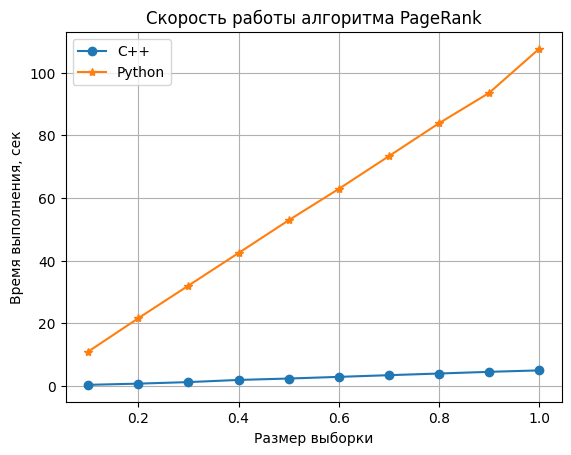
Precision: 0.5250

Recall: 0.9333

F1 Score: 0.6720

Pairorder: 0.6701

Как упоминалось ранее, в ходе работы была реализация на языке Python, которая работает гораздо медленнее, чем реализация на C++. Не будем рассматривать ее подробно. Таблица и рисунок ниже демонстрирует преимущества реализации на С++.



|  |  |  |
| --- | --- | --- |
| Часть от целого датасета | Время выполнения Python | Время выполнения С++ |
| 0,1 | 10,95 | 0,34 |
| 0,2 | 21,61 | 0,72 |
| 0,3 | 32,03 | 1,21 |
| 0,4 | 42,43 | 1,9 |
| 0,5 | 52,88 | 2,35 |
| 0,6 | 62,89 | 2,88 |
| 0,7 | 73,35 | 3,42 |
| 0,8 | 83,86 | 3,95 |
| 0,9 | 93,54 | 4,49 |
| 1,0 | 107,65 | 4,95 |

### HITS

На рисунке 2.1.2‑2 представлена зависимость времени выполнения итеративного HITS от размера графа (в частях от полного набора данных).

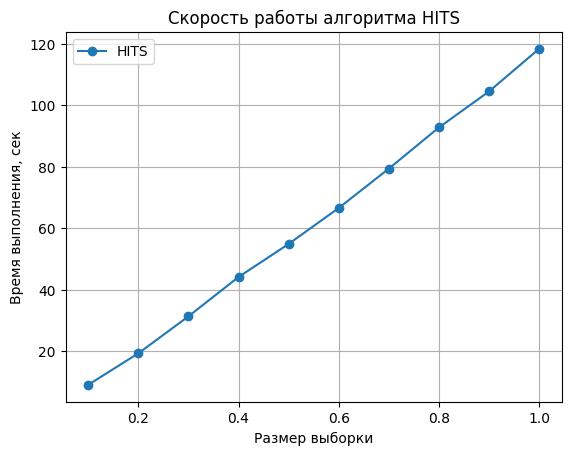
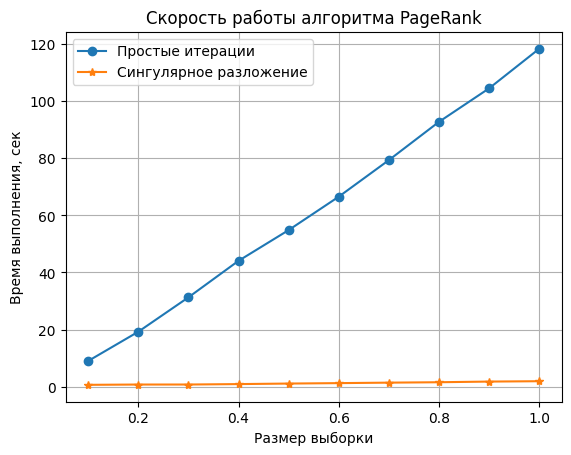


Рисунок 2.1.2‑1 Время выполнения 100 итераций HITS в зависимости от размера датасета MovieLens.

|  |  |  |
| --- | --- | --- |
| Часть от целого датасета | Время выполнения | Время считывания |
| 0,1 | 8,98 | 2,26 |
| 0,2 | 19,23 | 4,02 |
| 0,3 | 31,28 | 6,41 |
| 0,4 | 44,09 | 8,39 |
| 0,5 | 54,86 | 10,20 |
| 0,6 | 66,57 | 12,80 |
| 0,7 | 79,39 | 15,42 |
| 0,8 | 92,81 | 17,00 |
| 0,9 | 104,55 | 19,61 |
| 1,0 | 118,43 | 21,41 |

Эта таблица подтверждает линейную зависимость времени выполнения алгоритма в зависимости от m, что согласуется с теоретической оценкой.

Второй вариант подсчета рейтинга HITS (с помощью сингулярного разложения) показал гораздо лучшее время и при этом, он имеет такие же показатели метрик. Скорость работы можно сравнить по графику и таблицам ниже.



|  |  |  |
| --- | --- | --- |
| Часть от целого датасета | Время варианта с простыми итерациями | Время варианта с сингулярным разложением |
| 0,1 | 8,98 | 0,61 |
| 0,2 | 19,23 | 0,71 |
| 0,3 | 31,28 | 0,71 |
| 0,4 | 44,09 | 0,86 |
| 0,5 | 54,86 | 1,05 |
| 0,6 | 66,57 | 1,20 |
| 0,7 | 79,39 | 1,37 |
| 0,8 | 92,81 | 1,51 |
| 0,9 | 104,55 | 1,75 |
| 1,0 | 118,43 | 1,87 |

На рассматриваемом датасете возможно оценить качество полученных результатов. Воспользуемся метриками из главы 1.2. Для этого нужно определить функцию Пусть она будет иметь вид

Тогда усреднено для всех пользователей получаем следующие значения:

Accuracy: 0.5681

Precision: 0.5492

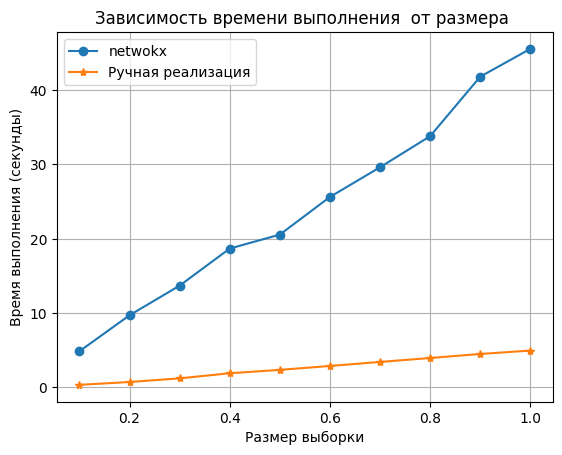
Recall: 0.8666

F1 Score: 0.6724

Pairorder: 0.6712

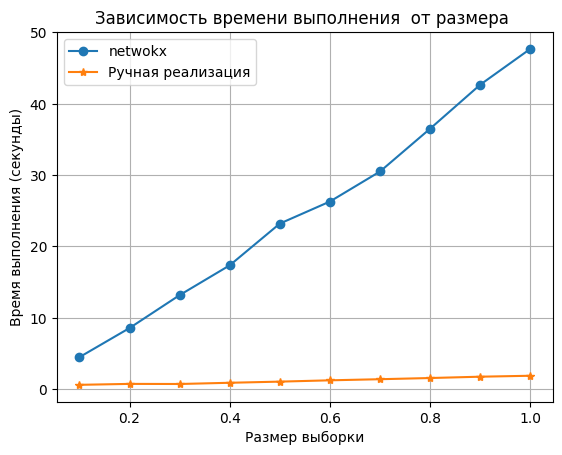
### Сравнение с NetworkX

Благодаря тому, что код написан на C++ и отсутствию промежуточного представления графа (в ручной реализации граф сразу хранится в матрице смежности) мы получаем в несколько раз превосходящую производительность.



|  |  |  |
| --- | --- | --- |
| Часть от целого датасета | Ручная реализация | networkX |
| 0,1 | 0,34 | 4,85 |
| 0,2 | 0,72 | 9,70 |
| 0,3 | 1,21 | 13,68 |
| 0,4 | 1,9 | 18,67 |
| 0,5 | 2,35 | 20,55 |
| 0,6 | 2,88 | 25,61 |
| 0,7 | 3,42 | 29,61 |
| 0,8 | 3,95 | 33,79 |
| 0,9 | 4,49 | 41,80 |
| 1,0 | 4,95 | 45,54 |

В алгоритме HITS ручная реализация также показывает свое превосходство. Но стоит отметить, что в такая скорость достигается только за счет того, что в ручной реализации граф сразу хранится необходимой для алгоритма матрице смежности.



|  |  |  |
| --- | --- | --- |
| Часть от целого датасета | Ручная реализация | networkX |
| 0,1 | 0.6090 | 4,4874 |
| 0,2 | 0.7408 | 8,5709 |
| 0,3 | 0.7261 | 13,1860 |
| 0,4 | 0.8972 | 17,3650 |
| 0,5 | 1.0580 | 23,2126 |
| 0,6 | 1.2409 | 26,3041 |
| 0,7 | 1.3964 | 30,5148 |
| 0,8 | 1.5628 | 36,5171 |
| 0,9 | 1.7489 | 42,6828 |
| 1,0 | 1.8806 | 47,7066 |

## Выводы по разделу

PageRank демонстрирует более высокую скорость работы по сравнению с HITS. Это связано с тем, что в HITS на каждую итерацию приходится 2 умножения матрицы на вектор, в то время как в PageRank только одно, что является менее ресурсоемкой задачей. Скорость работы хорошо демонстрирует рисунок ниже.

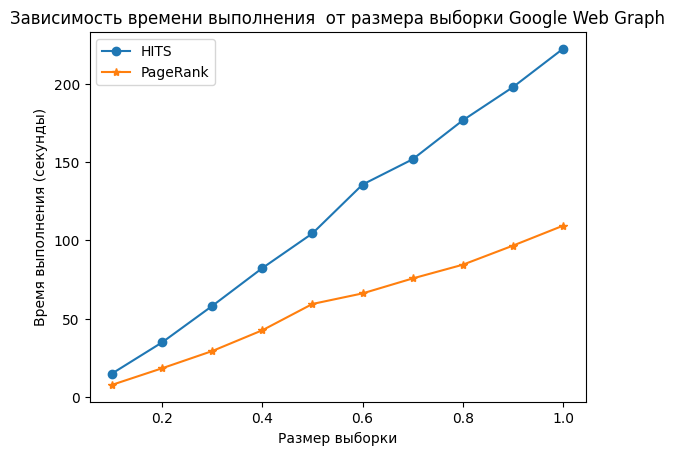


Рисунок 2.2.2‑1Зависимость времени выолениня от размера выборки алгоритмов PageRank и HITS на датасете Google Web Graph

Выполнение обоих алгоритмов линейно зависит от количества ссылок в сети. С увеличением количества ссылок, время выполнения алгоритмов возрастает прямо пропорционально.

Сфера применения: HITS лучше подходит для поиска информации по конкретным темам, так как позволяет выявлять страницы-концентраторы. PageRank же более универсален и применим для общего ранжирования веб-страниц.

Анализ результатов применения алгоритмов PageRank и HITS к набору данных MovieLens демонстрирует различия в их способности ранжировать и рекомендовать фильмы (таблица 2.3‑1)

PageRank, продемонстрировав более высокий показатель Recall, эффективнее находит релевантные фильмы, то есть, с большей вероятностью включит в рекомендации все фильмы, которые пользователь потенциально оценит высоко. Это делает его предпочтительным выбором для рекомендательных систем, нацеленных на охват максимального количества подходящих вариантов.

HITS, в свою очередь, выделяется более высоким показателем Precision, что говорит о его точности в отборе фильмов для рекомендаций. Алгоритм сфокусирован на минимизации ложных срабатываний, то есть, включения в рекомендации фильмов, которые пользователю не понравятся. HITS оптимален в сценариях, где критически важно рекомендовать исключительно те фильмы, которые гарантированно понравятся пользователю, пусть даже ценой пропуска некоторых релевантных вариантов.

|  |  |  |
| --- | --- | --- |
| Метрика | HITS | PageRank |
| Accuracy | 0.5681 | 0.5340 |
| Precision | 0.5492 | 0.5250 |
| Recall | 0.8666 | 0.9333 |
| F1 Score | 0.6724 | 0.6720 |
| Pairorder | 0.6712 | 0.6701 |

Таблица 2.3‑1 Сравнение метрик на датасете MovieLens.

# Заключение

В данном курсовом проекте было проведено комплексное исследование алгоритмов ссылочного ранжирования.

В ходе работы были детально изучены два фундаментальных алгоритма: PageRank и HITS. Теоретический анализ позволил выявить их базовые принципы, особенности реализации и сферы применения. PageRank, рассматривая сеть как единое целое, эффективно ранжирует страницы по глобальной важности, в то время как HITS концентрируется на поиске авторитетных источников информации и узлов-хабов в рамках конкретного тематического запроса.

Практическая часть проекта включала сравнительный анализ алгоритмов на наборе данных MovieLens, а также оценку времени их работы на нем и на Google Web Graph. PageRank продемонстрировал преимущества в скорости обработки данных и показателе Recall, что делает его предпочтительным для систем с высокой нагрузкой, нацеленных на максимальный охват релевантных результатов. HITS, в свою очередь, выделился высокой точностью рекомендаций (Precision), что критически важно в сценариях, где необходима гарантия релевантности каждого предложенного варианта.

Дальнейшие исследования в данной области могут быть направлены на:

Разработку гибридных алгоритмов, сочетающих сильные стороны PageRank и HITS.

Адаптацию алгоритмов к анализу динамических сетей с учётом временного фактора.

Исследование эффективности алгоритмов на различных типах данных.

Изучение возможности распараллеливания алгоритмов при помощи видеокарты.

В целом, проведенное исследование вносит вклад в понимание принципов работы и областей применения алгоритмов ссылочного ранжирования, подчеркивая их важность для решения широкого спектра практических задач в области анализа данных и информационного поиска.

# Библиографический список

1. Schütze, Hinrich, Christopher D. Manning, and Prabhakar Raghavan. Introduction to information retrieval. Vol. 39. Cambridge: Cambridge University Press, 2008.
2. Brin, Sergey, and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine." *Computer networks and ISDN systems* 30.1-7 (1998): 107-117.
3. Shahriari, Moshen, and Mahdi Jalili. "Ranking nodes in signed social networks." *Social network analysis and mining* 4 (2014): 1-12.
4. Zhu, Darui, et al. "Identification of key nodes in a power grid based on modified PageRank algorithm." *Energies* 15.3 (2022): 797.
5. Kleinberg, Jon M. "Authoritative sources in a hyperlinked environment." *Journal of the ACM (JACM)* 46.5 (1999): 604-632.
6. Gyöngyi, Zoltán, Hector Garcia-Molina, and Jan Pedersen. "Combating web spam with trustrank." *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. 2004.
7. Sebastiani, Fabrizio. "Machine learning in automated text categorization." *ACM computing surveys (CSUR)* 34.1 (2002): 1-47.
8. Tanudjaja, Francisco. *Using web graph structures to personalize search*. Diss. Massachusetts Institute of Technology, 2001.
9. Kleinberg, Jon M. "Authoritative sources in a hyperlinked environment." *Journal of the ACM (JACM)* 46.5 (1999): 604-632.

# Приложения

## Приложение А. Листинг программы

1. URL: <https://github.com/networkx/networkx> (дата обращения 27.05.2024) [↑](#footnote-ref-1)
2. URL: <https://www.kaggle.com/datasets/pappukrjha/google-web-graph> (дата обращения 27.05.2024) [↑](#footnote-ref-2)
3. URL: <https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset> (дата обращения 27.05.2024) [↑](#footnote-ref-3)