МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РЕАЛИЗАЦИЯ СИСТЕМЫ ЛОКАЛЬНОГО ПОИСКА ФАЙЛОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Синкевича Артема Александровича

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванова

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С увеличением вычислительной мощности компьютеров и ёмкости накопителей у пользователей появляется возможность хранить и использовать большое количество различных файлов, таких как текстовые документы, изображения, аудио- и видеофайлы. Однако возникает проблема: приходится тратить много времени на поиск полезной информации в океане данных, а иногда даже когда-либо увиденные или использованные пользователями файлы не могут быть легко найдены. Некоторые пользователи пытаются решить проблему, создавая сложную иерархию каталогов, но вручную поддерживать такую структуру трудно, и многие файлы остаются в неупорядоченных папках. Простого поиска по названиям файлов во многих случаях оказывается недостаточно, что привело к появлению продвинутых локальных поисковиков. Благодаря им стало возможно искать файлы по их содержимому и метаданным с помощью ключевых слов, фильтровать результаты по разным параметрам. Также существуют программы и системы, предназначенные для поиска изображений, в том числе по содержимому (поиск похожих картинок). Многие из этих решений доступны только для Windows или имеют закрытый исходный код. В отличие от веб-поисковиков, таких как Google, Яндекс, Bing, Baidu, большинство локальных поисковиков было разработано в 2000-х, и сейчас они не развиваются или не поддерживаются, из-за чего в них не используются более эффективные методы поиска с использованием нейронных сетей.

Целью данной работы является реализация системы локального поиска с применением методов машинного обучения, оценка качества её работы и сравнение с имеющимися решениями. Для достижения этой цели были поставлены задачи:

- исследование и описание методов поиска по различным форматам файлов;
- реализация системы локального поиска с применением выбранных методов;
- реализация методов оценки времени работы и качества поиска;
- оценка качества поиска и времени работы на выбранных наборах данных;
- сравнение системы с другими решениями.

1 Теоретическая часть

1.1 Текстовый поиск

1.1.1 Классические алгоритмы

Необходимость ускорения доступа к информации с помощью машин была осознана ещё в 1940-х годах, а идея автоматического ранжирования текстов на основе их релевантности определённому запросу возникла в 1960-х годах как альтернатива традиционным методам библиотечного индексирования. Салтоном и др. была предложена модель векторного пространства, в которой документы и запросы представляются как «мешки слов» (bag-of-words) с использованием разреженных векторов в соответствии с некоторой схемой взвешивания термов (слов или последовательностей символов), например TF-IDF, а сходство документов и запросов вычисляется с помощью косинусной меры (или, в более общем случае, скалярного произведения).

При использовании методов, основанных на точном сопоставлении термов, оценка релевантности между документом и запросом рассчитывается как сумма некоторой функции $f(t)$ от терма t и связанных значений по всем термам, встречающимся и в запросе, и в документе. Наиболее важные значения — частота терма (Term Frequency, сколько раз терм встречается в документе), частота документа (Document Frequency, количество документов, содержащих этот терм) и длина документа. Из первых двух величин получается функция TF-IDF, а одна из наиболее успешных схем взвешивания термов, Okapi BM25, учитывает все эти значения и до сих пор служит отправной точкой для многих подходов к ранжированию текстов.

Так как перебирать все документы при каждом запросе может быть очень долго, и пользователи могут просмотреть только небольшую их часть, то применяется структура данных «инвертированный индекс», в которой для каждого терма, встречающегося в корпусе, хранится список ссылок на документы, в которых он есть, и его частота в документах. Так как в документах встречаются далеко не все термы из словаря корпуса, то в среднем на запрос будет обработана только небольшая часть от всех документов.

1.1.2 Проблема несоответствия лексики

Хотя схемы взвешивания термов могут моделировать их важность на основе статистических свойств текстов, методы точного соответствия прин-

ципиально бессильны в тех случаях, когда термы в запросах и документах вообще не совпадают. Это происходит довольно часто, когда ищущие пользователи используют термины, отличные от тех, которые использовали авторы соответствующих документов. Таким образом, возникает фундаментальная «проблема несоответствия лексики».

Наиболее распространённым способом устранить разрыв между запросом и документами является дополнение запроса новыми словами, например, на основе обратной связи по релевантности, или с помощью синонимов. Альтернативой расширению запросов служит расширение документов. Часто применяется стемминг (удаление окончаний, суффиксов) или лемматизация (приведение слова к канонической форме) для обработки разных форм слов. Другой подход — устанавливать соответствия между запросами и документами в некотором семантическом пространстве, выведенном из корпуса документов.

1.1.3 Learning to Rank

BM25 и другие алгоритмы назначения весов термам считаются методами «без учителя», хотя содержат параметры, которые можно настроить в зависимости от имеющегося корпуса. Альтернативный подход, «Learning to Rank» («Обучение ранжированию»), использует методы машинного обучения с учителем для создания моделей ранжирования. Для этого вручную создаются и отбираются признаки, например, статистические свойства термов в документах. При этом признаки могут быть статичными (не зависеть от запроса), динамическими (зависеть от запроса) или быть признаками самого запроса (например, его длина). Выделяют три подхода обучения моделей: поточечный, попарный и списочный.

1.1.4 Глубокое машинное обучение

После начала широкого использования в области компьютерного зрения, а затем обработки естественного языка, глубокое обучение (Deep Learning) начало использоваться и для ранжирования текстов. Алгоритмы глубокого обучения освобождают текстовый поиск от ограничений точного сопоставления термов и необходимости трудоёмкого ручного создания признаков.

Нейронные модели ранжирования до BERT можно разделить на две группы: модели на основе представлений и модели на основе взаимодействий. Модели, основанные на представлениях, обучают для вычисления «плотных»

векторных представлений запросов и документов по отдельности для их сравнения с помощью простых метрик, таких как косинусная мера и скалярное произведение. Известным примером является DSSM, которая строит символьные n-граммы из входных данных и пропускает их через несколько полносвязных слоёв для создания векторного представления. С другой стороны, модели, основанные на взаимодействиях, такие как DRMM, анализируют отношения между термами запроса и документа, фиксируя их взаимодействие как матрицу схожести, которая затем используется для расчёта оценки релевантности. Оба типа моделей могут включать различные компоненты (например, свёрточные и рекуррентные сети) для извлечения информации о релевантности. При обучении моделей, основанных на представлениях или взаимодействиях, обычно используются оценки релевантности в качестве входных данных. Эти модели используют только термы запроса и документа, не включая никаких дополнительных признаков.

1.1.5 BERT

BERT (Bidirectional Encoder Representations from Transformers) — это модель для генерации контекстных представлений токенов входных последовательностей, которые можно применять во множестве задач обработки естественного языка. BERT использует кодирующую часть (энкодер) архитектуры «трансформер».

1.1.6 BERT для переранжирования

Первая модель на основе BERT для ранжирования текстов, monoBERT, принимает на вход текст запроса и текст документа и возвращает оценку релевантности документа по отношению к запросу. Но применение такой модели к каждому тексту в корпусе для каждого пользовательского запроса непрактично из-за длительности обработки крупной нейронной сетью. Поэтому используется подход «retrieve and rerank», который предполагает использование поиска по ключевым словам для определения текстов-кандидатов, после чего этот список переранжируется.

Модели, принимающие на вход одновременно запрос и текст документа, как monoBERT, называются «кросс-энкодерами» (cross-encoder). Но существуют и архитектуры, называемые «би-энкодерами» (bi-encoder), которые позволяют обработать запрос и документ независимо друг от друга (преобразовать

в векторы) и проводить оценку релевантности с помощью простой функции (например, косинусной меры).

Так как длина входной последовательности BERT ограничена, то возникает проблема обработки длинных документов. Одно из решений представлено в BERT-MaxP. Для обучения модели документы просто разделяются на перекрывающиеся отрывки, и все фрагменты из релевантных документов считаются релевантными, а из нерелевантных документов — как нерелевантные. При применении модели документ аналогично делится на фрагменты, оценивается релевантность каждого, а затем оценки агрегируются с помощью простого метода, такого как выбор максимальной оценки. Также существуют модели, агрегирующие представления фрагментов длинного текста, а не их оценки, такие как PARADE.

Кроме monoBERT и других моделей, использующих поточечный подход, существуют модели с попарным и списочным подходом. Кроме BERT могут использоваться и модели на основе полной архитектуры «трансформер», такие как T5, BART, UniLM. Например, была предложена модель monoT5, базовая версия которой оказывается лучше большей по количеству параметров модели monoBERT на MS MARCO. При этом самая большая модель достигает лучших известных результатов на некоторых наборах данных.

Многие модели имеют достаточно большое число параметров, а значит могут быть недостаточно быстрыми для локального использования. Для уменьшения моделей используется дистилляция знаний — метод, при котором меньшая модель («ученик») учится подражать большей модели («учителю»). Эта идея была применена и к BERT — было создано множество моделей, одна из наиболее эффективных — MiniLM, для которой доступна многоязычная версия.

Почти все модели переранжирования поддерживают только английский язык, так как распространённые наборы данных для их обучения тоже на английском. В работе Бонифацио и др. на основе MS MARCO с помощью машинного перевода был создан набор данных mMARCO, а также дообучены на нём модели mT5 и mMiniLM. Было показано, что обе модели показывают близкие результаты на тестовой выборке, сравнимые с англоязычными моделями.

1.1.7 BERT для поиска с помощью плотных векторных представлений

В отличие от кросс-энкодеров, с помощью би-энкодера можно предпосчитать векторные представления документов и сохранить в специальной базе данных. Для выполнения поиска достаточно вычислить представление текста запроса и найти в БД заданное количество ближайших документов по определённой метрике, для чего используются алгоритмы приблизительного поиска соседей (ANN, Approximate Nearest Neighbor). При этом модели для кодирования запросов и документов могут отличаться, но должны отображать их в одно пространство.

Модели Sentence-BERT позволяют определять сходство текстов, но их можно применить и для поиска, при этом и для запросов, и для документов будет использоваться одна модель. В качестве основы используется BERT или RoBERTa, а контекстные представления токенов усредняются. Кроме того, в рамках проекта Sentence-Transformers с помощью дистилляции знаний были созданы многоязычные модели, поддерживающие более чем 50 языков. Также были рассмотрены другие би-энкодеры, такие как DPR и ANCE, и модель, генерирующая несколько векторов для каждого текста — ColBERT.

1.2 Поиск изображений

1.2.1 Классические алгоритмы

Для работы с большими базами изображений существует потребность в эффективных механизмах поиска в них. Поиск изображений на основе текста с помощью ключевых слов страдает от необходимости ручного аннотирования изображений, неточности этих описаний и проблемы несоответствия лексики. Поэтому применяется поиск изображений по содержанию (Content-based Image Retrieval, CBIR), поскольку его можно полностью автоматизировать. Изначально CBIR подразумевал поиск похожих изображений по изображению-запросу, при этом использовались простые признаки, и такой подход сталкивался с «семантическим разрывом» между низкоуровневыми характеристиками и высокоуровневыми понятиями.

Признаки изображений можно разделить на глобальные и локальные. Глобальные признаки, такие как цвет, форма, текстура, обеспечивают общее представление, позволяют быстро находить изображения и вычислять сходства, но плохо работают на комплексных изображениях. Локальные признаки

определяются как ключевые точки или части изображения, такие как углы, области и края, и являются более устойчивыми к изменениям масштаба, поворотам, переносу, изменениям фона, частичным перекрытиям. Одним из наиболее известных локальных дескрипторов является SIFT, извлекающий особые точки, описывающие изображение. SIFT устойчив к поворотам, изменениям яркости и размеров изображений, частичным перекрытиям, но плохо справляется с сопоставлением при высокой размерности признаков.

1.2.2 Глубокое машинное обучение

Кроме классических алгоритмов машинного обучения, в системах поиска изображений могут применяться искусственные нейронные сети, в частности, свёрточные НС, с помощью которых были получены выдающиеся результаты для различных задач компьютерного зрения, таких как распознавание и классификация объектов, сегментация изображений. В отличие от предыдущих методов, для СНС не требуются вручную созданные и отобранные признаки, но для достижения высокого качества нужна большая и разнообразная обучающая выборка изображений с известными классами. В одной из первых работ о использовании СНС для поиска изображений было показано, что с помощью моделей, преодобученных на больших наборах изображений (ImageNet), можно получить результаты, превосходящие предыдущие методы. Для этого использовались представления, полученные последними (полносвязными) слоями СНС.

Модель CLIP (Contrastive Language-Image Pre-training) завоевала популярность в области поиска изображений и текстов, так как обеспечивает высокую точность поиска без дообучения путём предобучения на большом количестве пар текст-изображение. CLIP является би-энкодером и состоит из двух частей: энкодера изображений и и энкодера текста. Эти модели вместе обучались создавать такие представления, что косинусная мера правильных пар изображение-текст как можно больше, а неподходящих — меньше. Так как оригинальный текстовый энкодер поддерживает только английский язык, в рамках проекта Sentence-Transformers была создана его многоязычная версия, поддерживающая более 50 языков, с помощью дистилляции знаний из оригинала в многоязычный DistilBERT.

1.3 Метрики качества ранжирования

Метрики качества ранжирования необходимы для численной оценки качества списка результатов поиска. Так как пользователи поисковых систем обращают внимание в основном на первые результаты в списке, то многие метрики сопоставляют позиции результатов с идеальным ранжированием документов, определяемым на основе оценок релевантности, созданных вручную. Эти метрики обычно вычисляются по первым K документам, полученным системой, и усредняются по всем запросам в тестовом наборе.

Стандартными метриками при использовании бинарных оценок релевантности являются точность (Precision) и полнота (Recall). Наиболее распространённой метрикой, учитывающей как позиции документов, так и степени релевантности, является NDCG (Normalized Discounted Cumulative Gain). Так как в реальных наборах данных не для всех пар запрос-документ известна оценка релевантности, то можно рассматривать метрику NDCG', вычисленную для списка результатов после удаления из него неоценённых документов.

1.4 Наборы данных

TREC 2004 Robust Track (Robust04) — широко используемый в литературе набор данных для текстового поиска, представленный на TREC (Text REtrieval Conference). Содержит 528 тысяч документов (1.7 ГиБ текста, средняя длина — 548 слов) — новостных статей за 1989-1996 годы и 250 тестовых запросов, для которых доступен заголовок, короткое описание (одно предложение) и полное описание. Набор данных содержит 311 тысяч оценок релевантности, в среднем по 1250 на запрос. На основе этого набора с помощью машинного перевода создана многоязычная версия mRobust04, включающая в себя 9 языков, в том числе и русский.

Microsoft COCO (Common Objects in Context) — распространённый набор данных, используемый для поиска изображений по тексту. Он содержит фотографии повседневных сцен, содержащих распространённые объекты. Версия 2017 г. включает в себя 118 тысяч изображений в обучающей выборке, 5 тысяч в валидационной и 41 тысячу в тестовой. Для каждого изображения доступно 5 описаний (предложений) на английском языке. Для данной работы использовалась валидационная выборка (размером 777 МиБ), для которой были созданы описания на русском языке с помощью машинного перевода.

2 Практическая часть

Проект состоит из основного веб-сервера `indexer`, служащего бэкендом и выполняющего индексацию и поиск, фронтенда — веб-интерфейса `client_ui`, библиотеки `common_lib`, сервера нейронных сетей `nn_server`, программы запуска `launcher` и программы измерения качества и скорости поиска `benchmarks`. Для всех компонентов был выбран язык Rust из-за сочетания производительности и безопасной работы с памятью и потоками. Разработанная система состоит из нескольких микросервисов: кроме `indexer` и `nn_server` используется Apache Tika для извлечения текста и метаданных и Elasticsearch — база данных для полнотекстового и векторного поиска. Микросервисный подход позволяет разделить систему на автономные части, взаимодействующие через сетевые вызовы, используя программные интерфейсы (API). Благодаря этому можно использовать сервисы, написанные на разных языках, а также отключать или заменять части системы.

Система поддерживает ОС Windows и Linux. Для работы рекомендуется ГП NVIDIA с 4 ГБ ОЗУ, но возможно использование нейронных сетей на ЦП. Система поддерживает обработку множества различных форматов: текстовых файлов, изображений, аудио- и видеофайлов, документов, архивов.

2.1 Выбор алгоритмов

Среди алгоритмов, описанных в разделе 1.1 для текстового поиска и в 1.2 для поиска изображений, были выбраны следующие:

- Поиск по ключевым словам с помощью BM25 с использованием стемминга и фильтрации стоп-слов для английского и русского языков.
- Текстовый поиск с помощью би-энкодера `sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2` — наименьший многоязычный би-энкодер из проекта Sentence-Transformers.
- Переранжирование результатов текстового поиска с помощью кросс-энкодера `unicamp-dl/mMiniLM-L6-v2-mmarco-v2`, выбранного по тем же признакам, что и би-энкодер.
- Поиск изображений по текстовому описанию или изображению-примеру с помощью CLIP. Выбраны следующие модели:
  - Для изображений — `sentence-transformers/clip-ViT-B-32`, наименьшая версия, использующая ViT;

– Для текстов — `sentence-transformers/clip-ViT-B-32-multilingual-v1`, многоязычная версия оригинального энкодера.

2.2 Основной сервер

Основной веб-сервер `indexer` реализует REST API, использующее протокол HTTP и формат данных JSON, позволяющее получать и устанавливать настройки, запускать индексацию и получать её статус, выполнять поиск, получать файлы для предпросмотра и статичные файлы интерфейса.

Процесс индексирования начинается при запросе пользователя или получении события о изменении отслеживаемых файлов. Затем вычисляются необходимые изменения на основе списков файлов в ФС и БД. Далее файлы обрабатываются парсерами: считываются с диска, с помощью Apache Tika определяется тип, извлекается текст, изображение, метаданные, затем с помощью сервера НС вычисляются представления текстов и изображений. Команды для добавления, изменения, удаления файлов со всеми полученными дополнительными данными отправляются в БД Elasticsearch.

Запрос поиска может быть получен от пользователя через интерфейс или от программы измерения качества и скорости поиска. Для семантического поиска выполняются запросы к соответствующим моделям в сервере НС. Затем выполняется поиск в БД по ключевым словам по заданным полям и поиск ближайших векторов к представлениям запроса. Полученный список результатов переранжируется с помощью оценок от кросс-энкодера.

2.3 Сервер нейронных сетей

Так как выбранные нейронные сети используют фреймворк PyTorch и библиотеки Transformers, Sentence-Transformers, то первая версия сервера была создана на языке Python с помощью этих библиотек как отдельный сервис. Но занимаемый размер приложения на диске и использование оперативной памяти оказались достаточно велики, и была создана версия на языке Rust, использующая ONNX Runtime — библиотеку выполнения моделей машинного обучения, доступную для многих языков, поддерживающую разные устройства и во многих случаях позволяющую уменьшить использование ресурсов.

Итоговая версия `nn_server` реализует REST API, позволяющее получать представления текстов и изображений с помощью CLIP, представления текстов би-энкодером и оценки пар запрос-документ кросс-энкодером. Для

обработки длинных текстов би-энкодером был выбран подход модели BERT-MaxP: текст разбивается на отрывки (100 слов) с помощью скользящего окна (с шагом в 75 слов), и их представления усредняются. Также выбираются несколько (3) отрывков для дальнейшего переранжирования результатов.

2.4 Общая библиотека

В общую для всего проекта библиотеку `common_lib` были вынесены описания структур и их методов, необходимых в разных программах.

2.5 Программа запуска системы

Так как для работы требуется запуск и ожидание всех компонентов системы, была создана программа автоматизации запуска `launcher`.

2.6 Интерфейс

Для создания графического интерфейса пользователя использовался язык Rust и компиляция в WebAssembly, что позволяет упростить работу с остальной системой. Веб-интерфейс позволяет использовать браузеры, установленные практически у всех пользователей.

Интерфейс состоит из трёх вкладок: поиск, статус индексации и настройки. На основной вкладке можно выбрать тип запроса (текст или изображение), ввести его, выбрать тип поиска, папку поиска, тип файлов, дополнительные фильтры. После выполнения поиска будут показаны результаты — названия файлов, пути к ним, кнопки для предпросмотра и открытия файлов, и метаданные, отображаемые при необходимости. Для изображений (и аудио-, видеофайлов, если возможно) отображается миниатюра справа. Также для текстовых файлов показывается подходящий запросу отрывок.

На странице статуса отображается количество файлов в индексе, размер индекса, статус индексации (количество добавляемых, обновляемых, удаляемых файлов и прогресс индексации) в реальном времени, а также находятся кнопки индексации и очистки индекса. На последней странице можно редактировать настройки: URL всех сервисов, индексируемые папки, параметры индексации, отображения результатов, параметры сервера нейронных сетей.

2.7 Определение параметров нейронных сетей

Для оптимальной работы нейронных сетей необходимо подобрать размер пакета — количество одновременно обрабатываемых данных. Была прове-

дена индексация наборов данных MS COCO и mRobust04 с разными размерами пакетов, построены графики и выбраны оптимальные значения.

Также для наилучшего качества поиска нужно выбрать коэффициенты, с которыми будут складываться оценки файлов, полученные с помощью BM25, би-энкодера и кросс-энкодера. Были построены графики метрик NDCG@20 и NDCG'@20 на наборе mRobust04 в зависимости от коэффициентов би-энкодера и кросс-энкодера и выбраны оптимальные значения.

2.8 Сравнение реализаций серверов нейронных сетей

Первая реализация сервера нейронных сетей была создана на языке Python с использованием PyTorch, а затем — на Rust с ONNX Runtime. Обе версии были протестированы при индексации набора данных, полученного совмещением валидационной выборки MS COCO и подмножества английской версии mRobust04 (10504 документов). Индексация версией на Rust заняла на 20% меньше, при этом использование ОЗУ уменьшилось на 37%, а время запуска сервера — приблизительно в 3 раза.

2.9 Сравнение с другими решениями

Большая часть локальных поисковиков доступна только для Windows и имеет закрытый код: Поиск Windows, Likasoft Archivarius 3000, Copernic Desktop Search, а многие уже не поддерживаются: Yahoo! Desktop Search, Google Desktop, «Персональный поиск Яндекса», но есть и открытые кросс-платформенные системы: DocFetcher, Recoll. Все они используют поиск по ключевым словам, и немногие поддерживают стемминг для русского языка.

Такие программы, как NoClone, FIRE, LIRE (LIRE Solr), digiKam, imgSeek, Img(Rummager) поддерживают поиск изображений, похожих на изображение-запрос, но не обладают возможностью искать изображения по содержимому с помощью текстового запроса, поэтому сравнение с ними системы, созданной в данной работе, не проводилось.

2.10 Программа измерения качества поиска

Для измерения качества и скорости поиска на выбранных наборах данных была создана программа `benchmarks`. Для работы она использует то же API основного сервера, что и пользовательский интерфейс. Для набора данных mRobust04 программа позволяет выполнить поиск по всем запросы из

файла и сохранить результат в подходящем для утилиты `trec_eval` формате, с помощью которой можно вычислить метрики. Для MS COCO аналогично можно получить таблицы результатов всех запросов.

2.11 Сравнение методов текстового поиска

Было проведено сравнение конфигураций системы для текстового поиска: только BM25, только би-энкодер, их объединение, а также добавление кросс-энкодера. Использованные модели не были обучены на данных для тестирования. Были протестированы и сторонние решения: Recoll и DocFetcher.

Для англоязычного mRobust04 выяснилось, что DocFetcher индексирует быстрее всего и созданный индекс имеет меньший размер, Recoll и разработанная система в конфигурации только с BM25 выполняют индексирование за сравнимое время, а при использовании би-энкодера время индексации увеличивается на порядок. Время обработки запроса у всех конфигураций невелико, и значительно увеличивается только при использовании кросс-энкодера. Разработанная система при использовании только BM25 превосходит по всем метрикам и DocFetcher, и Recoll. Также выяснилось, что хотя отдельно би-энкодер хуже BM25, вместе их качество существенно лучше отдельных компонентов. Переранжирование результатов дополнительно улучшает качество.

Все протестированные системы оказались немного медленее и хуже на русскоязычной версии mRobust04. С помощью би-энкодера оказался возможен поиск документов на одном языке с помощью запросов на другом, но с немного худшим качеством, чем при одноязычном поиске. Также было проведено сравнение с лучшими известными решениями, и сделан вывод, что при заданных ограничениях ресурсов система показывает высокое качество.

2.12 Оценка качества поиска изображений

Также была проведена оценка качества поиска изображений по их описанию в MS COCO. Были использованы как описания на английском, так и созданные с помощью машинного перевода на русском языке. Выяснилось, что для англоязычных запросов в половине случаев нужное изображение будет среди первых 5, а для русскоязычных — 8. В большинстве случаев релевантное изображение оказывается среди первых 20, и почти всегда — среди первых 100. При этом для русскоязычных запросов значения метрик качества немного меньше, что отчасти можно объяснить качеством машинного перевода.

ЗАКЛЮЧЕНИЕ

В данной работе была создана система локального поиска с графическим интерфейсом, использующая современные методы машинного обучения. Была изучена литература о поиске текстов и изображений, описаны различные алгоритмы и выбраны наиболее подходящие. Для проекта было создано несколько компонентов: основной веб-сервер, сервер нейронных сетей, общая библиотека, веб-интерфейс, программы запуска системы и оценки качества поиска и времени работы. Были описаны метрики оценки качества, найдены наборы данных для тестирования, подобраны настройки всех компонентов, выполнено сравнение разных конфигураций разработанной системы и существующих решений. Выяснилось, что система показывает высокие результаты на использованных наборах данных и обладает другими преимуществами по сравнению с имеющимися локальными поисковиками.

В ходе написания работы были решены следующие задачи:

- изучены и описаны методы поиска по различным форматам файлов;
- реализована система локального поиска с применением выбранных методов;
- реализованы методы оценки времени работы и качества поиска;
- оценено качество поиска и время работы на выбранных наборах данных;
- проведено сравнение системы с другими решениями.