МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)»

Институт №8 «Информационные технологии и прикладная математика» **Кафедра 810Б**
Направление подготовки 02.04.02 ФИИТ Группа М80-210М-19
Квалификация (степень) магистр

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

На тему: Программные средства для предметно-ориентированного поиска и рекомендаций научных статей по вычислительному материаловедению

Автор квалификационной работы Сомик Александр Николаевич (_____)
(Фамилия, имя, отчество)

Научный руководитель Абгарян Каринэ Карленовна (_____)
(Фамилия, имя, отчество)

Рецензент Харченко Вячеслав Александрович (_____)
(Фамилия, имя, отчество)

К з а щ и т е д о п у с т и т ь

Зав. кафедрой Абгарян Каринэ Карленовна (_____)
(Фамилия, имя, отчество)

“4” июня 2019г.

Москва 2021г.

РЕФЕРАТ

Магистерская диссертация содержит 38 страниц, 19 рисунков, 1 таблицу. Список использованных источников содержит 10 позиций.

ПРЕДМЕТНО-ОРИЕНТИРОВАННЫЙ ПОИСК, ТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ, АНАЛИЗ ДОКУМЕНТОВ В ОБЛАСТИ ВЫЧИСЛИТЕЛЬНОГО МАТЕРИАЛОВЕДЕНИЯ, АРХИТЕКТУРА СЕРВЕРНОГО ПРИЛОЖЕНИЯ

Магистерская диссертация посвящена реализации программного комплекса по поиску семантически близких документов в коллекции. В рамках данной работы были проанализированы методы получения векторных представлений для документов, выбрана и обучена наиболее эффективная на основе результатов подзадачи модель, сформирована коллекция научных статей в области вычислительного материаловедения. Реализован и интегрирован с уже существующим программный комплекс, позволяющий добавлять документ в коллекцию с использованием Google Drive, предлагать пользователю семантически близкие документы коллекции при поиске. Данный проект позволит специалистам и студентам кафедры существенно сократить время поиска предметной литературы при исследовании, обзоре аналогов и обучении.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В своей научной деятельности специалисты в области вычислительного материаловедения часто сталкиваются с проблемой поиска научных работ, связанных с темой их исследования. Классические системы поиска по короткому запросу зачастую не позволяют описать в запросе характеристики исследуемой темы. В данной работе предлагается использовать иную парадигму информационного поиска – поиск по документу. Для решения этой задачи предлагается получить семантические векторные представления для документов коллекции и документа-запроса, и затем находить релевантные документу-запросу результаты как наиболее близкие по косинусной мере. Таким образом, задача поиска релевантных документов сводится к задаче поиска наилучшей модели векторизации текста.

На данный момент на базе кафедры реализован программный комплекс для индексирования и предметно-ориентированного поиска для полнотекстового поиска в коллекции документов (далее - Crystal-search). Данный поиск не позволяет оперативно найти семантически близкие документы, что является неотъемлимой частью исследовательской и образовательной деятельности. Таким образом возникла задача реализации программного средства для преметно-ориентированного поиска в большой коллекции документов кафедры.

Задача была решена с помощью языка программирования Python, средства управления базами данных PostrgeSQL и BigARTM – библиотеки, поддерживающей методы обучения тематических моделей.

В качестве решения было предложено разработать приложение, состоящее из трех частей:

1. База данных
2. Серверное приложение
3. Модуль по работе с бибилотекой BigARTM

В рамках данной работы была разработана архитектура приложения, а также произведена успешная интеграция с существующим программным комплексом.

ОСНОВНАЯ ЧАСТЬ

1.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. МЕТОДЫ ПОЛУЧЕНИЯ ВЕКТОРНЫХ ПРЕДСТАВЛЕНИЙ ДОКУМЕНТОВ

В настоящее время самым распространенным видом поиска информации является поиск по небольшим запросам в поисковой строке. Данный поиск позволяет найти материалы, в которых встречается поисковая строка. В сфере вычислительного материаловедения поиск по одному или нескольким словам (например по хмимическим элементам или греческим обозначениям) может оказаться неточным, т.к. разные комбинации могут относиться к различным подтемам данной сферы. Когда студент или начинающий специалист только начинает изучение какой то области, появляется проблема отсутствия знания предметной области и, таким образом, грамотных запросов к системе. Они вынуждены изучить огромное количество документов, многие из которых будут нерелевантны тематике при полнотекстовом поиске, как уже упоминалось ранее. В данном случае, решение этой проблемы – поиск по документу.

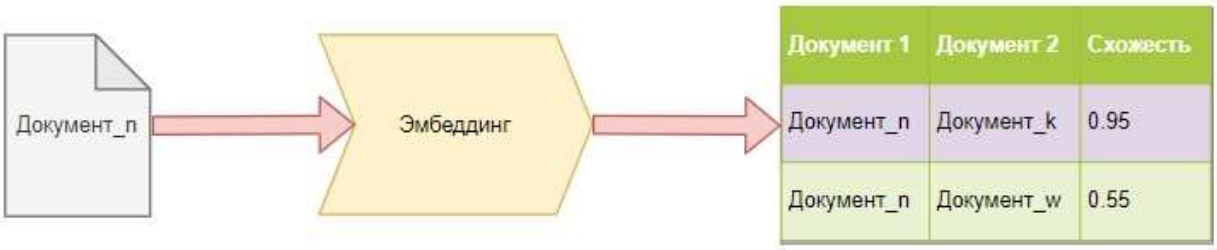

Рис. 1.1 Поиск по документу

Данный подход, представленный на (Рис. 1.1) позволяет на вход системе отправлять не просто запрос в виде короткого предложения, а целый документ или черновик документа. С помощью метода получения векторного представления документа, выбираются наиболее схоже документы по семантической близости. Семантическая близость – косинусная мера между векторными представлениями двух документов. Формула нахождения представлена ниже:

$$cossim = \frac{\vec{q} \cdot \vec{d}}{\|\vec{q}\| \cdot \|\vec{d}\|} = \frac{\sum_{i=1}^{n} \overrightarrow{q_i} \cdot \overrightarrow{d_i}}{\sqrt{\sum_{i=1}^{n} q_i} \sqrt{\sum_{i=1}^{n} d_i}} \quad (1.1)$$

Далее осуществляется сравнение исходного документа со всеми документами коллекции и выбираются документы с наибольшей косинусной мерой.

В работе были рассмотрены наиболее популярные методы получения семантических векторых представлений из документа: BERT, FastText, Doc2Vec, Word2Vec, TF-IDF и ARTM.

1.1.1. BERT

BERT, сокращение от Bidirectional Encoder Representations from Transformers (двунаправленная нейронная сеть-кодировщий). BERT показывает одни из лучших результатов в сфере генерации текста и его обработки. Это модель представления языка, которая использует сразу два направления в каждом внутреннем слое. То есть результаты проходов совмещаются, что значительно увеличивает устойчивость и повышает результат модели на типовых задачах.

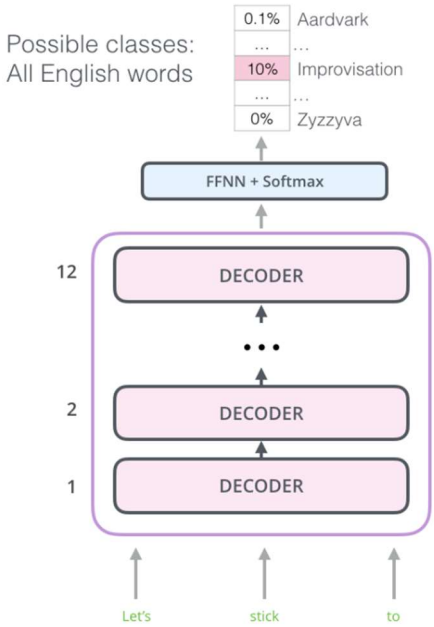

Рис. 1.2 Трансформер

BERT использует стратегию под названием «трансформер», структура которого представлена на (Рис. 1.2). Это механизм, способный изучать контексты между словами предложения. Трансформер состоит из двух различных механизмов: кодировщика и декодера. Первый занимается обработкой и векторизацией данных, а второй – генерацией предсказаний на основе декодировщика [1].

BERT обучается на огромных объемах данных с помощью мощных вычеслительных калстеров. Простой пользователь может использовать готовую сеть как базу для своей задачи.

1.1.2. Word2Vec, FastText, Doc2Vec

Word2Vec и развивающие данный подход модели Doc2Vec и FastText – модели на основе искусственных нейронных сетей, предназначенные для получения векторных представлений слов на естественном языке.

Традиционный способ представления слов - это «one-hot» вектор, который, представляет собой вектор, в котором только один целевой элемент равен 1, а остальные - 0. Длина вектора равна размеру уникального словаря в корпусе. Пример вектора представлен на (Рис. 1.3):

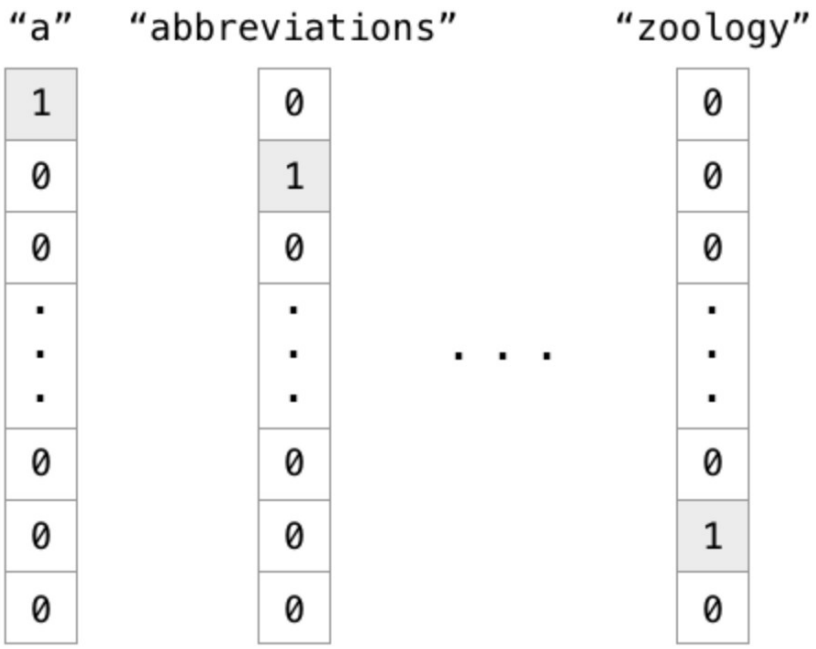

Рис. 1.3 «One-hot» векторы

Существует два типа Word2Vec: Skip -gram и Continuous Bag of Words (CBOW).

Метод Skip-gram позволяет по входящему слову предсказывать окружающие его слова. Вход и выход представляет из себя одинаковые по длине «one-hot» вектора [1] .

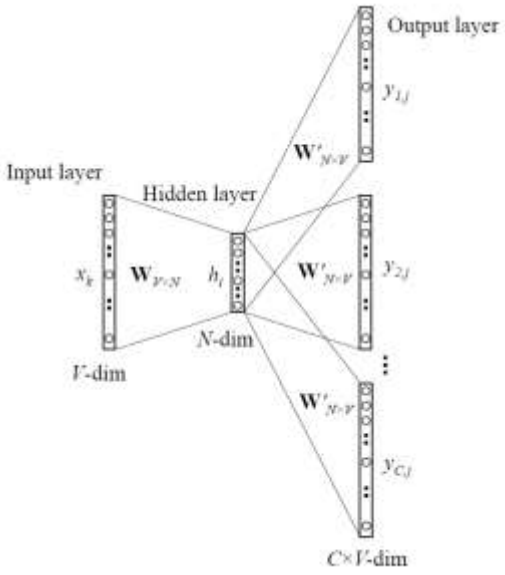

Рис. 1.4 Skip-gram

Сеть содержит 1 скрытый слой (Рис. 1.4), размер которого выбирается на первоначальной настройке. Выходы – вероятностные оценки ожидаемых

слов при прохождении softmax-слоя. Такая сеть по сути выбирает наиболее вероятный набор слов вокруг входного. Как уже говорилось, вход и выход имеют один размер, заранее заданный пользователем.

(CBOW) – архитектура мешка слов . Его задача – понять по контексту (или окну слов) о каком слове идет речь. Архитектура напоминает Skip-gram в зеркальном отражении [2].

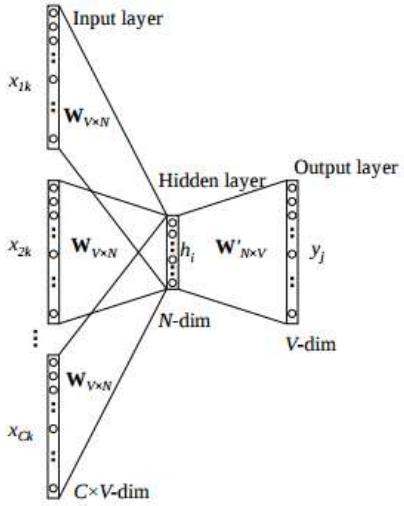

Рис. 1.5 CBOW

Структура нейросети CBOW продемонстрирована на (Рис. 1.5). Самая большая разница между Skip-Gram и CBOW заключается в способе генерирации векторных представлений слов. Выходной слой вычисляется как среднее арифметическое от скрытого, в отличии от Skip-gram.

В данном случае в качестве выхода в сети используется не окно слов, а лишь одно представление слова в виде «one-hot» вектора.

По производительности Skip-грамм и CBOW показывают в среднем одинаковые результаты [2].

Преимущество FastText над Word2Vec заключается в том, что для получения векторного представления слов одновременно используются модели Skip-gram и CBOW, что существенно увеличивает скорость обучения и использования модели по сравнению с классическим подходом.

Цель Doc2Vec - создать числовое представление документа независимо от его длины:

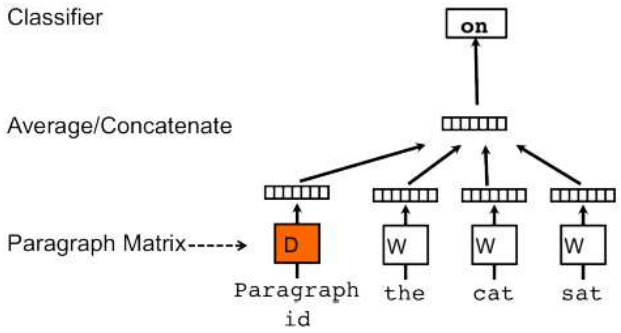

Рис. 1.6 Doc2Vec

Но вместо того, чтобы использовать только слова для предсказания следующего слова, добавлен еще один вектор признаков, который уникален для документа, что продемонстрировано на (Рис. 1.6).

Таким образом, при обучении векторов слов W, вектор документа D также обучается, и в конце обучения он содержит числовое представление документа, которое затем можно использовать для решения других задач.

Данный подход более ориентирован на работу с большими документами, именно за счет использования данной компоненты.

1.1.3. TF-IDF

TF-IDF (от англ. term frequency и inverse document frequency — частота слова и обратная частота документа) — одна из разновидностей оценок статистических величин, которая используется для того, чтобы в соответсвии с контекстом документа, хранящегося в массиве документов, вычислить оценку важности нужного слова в нем. Вес слова прямо пропорционален тому, насколько часто данное слово употребляется в текущем документе и обратно пропорционален частоте использования слова в остальных документах коллекции.

TF (частота слов) является характеристикой, которая представляет отношение числа появлений определенного слова к полному набору слов в документе. Данная частота пропорциальна весомсти слова в рамках анализиуемого документа.

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D) \quad (1.2)$$

IDF (обратная частота документа) является характеристикой, которая показывает инверсию частотности для необходимого слова в рамках раасматриваемого документа. Используя данную оценку, существует возможность снизить важность логически незначимых слов — например, служебных частей речи.

Метрика TF-IDF нашла примененеие в алгоритмах система поиска, использующихся для анализа документов. К примеру, описанная метрика используется в алгоритме расчета меры релевантности содержания по поисковому запросу пользователя. Также данная оценка используется при выявлении степени сходства нескольких документов [1].

1.1.4. APTM

APTM – метод аддитивной регуляризации тематических моделей. Данный подход позволяет обучать тематические модели, учитывающие специфику предметной области.

Пусть D – множество документов (коллекция), W – множество всех употребляемых в коллекции токенов (словарь коллекции). Документ коллекции $d \in D$ представляет из себя последовательность токенов $(w_1, \dots, w_{n_d}) \in W$, причем каждому токену w ставится в соответствие число n_{dw} его вхождений в документ d.

Коллекция представляется в виде:

$$F = (f_{wd})_{W \times D} \quad (1.3)$$

$$f_{wd} = \frac{n_{dw}}{n_d} \quad (1.4)$$

Предполагается, что существует некоторое количество тем, которые распределяются между всей коллекцией документов. Тогда коллекция представляет собой множество $(w_i, d_i, t_i), i = 1 \dots n$, порожденных дискретным распределением $p(w, d, t)$, определенном на конечном вероятностном пространстве $W \times D \times T$.

В тематическом моделировании темы представляются дискретными распределениями на множестве токенов, а документы – дискретными распределениями на множестве тем. Пусть $p(w|t)$ – вероятность, с которой токен w встречается в теме t, а $p(t|d)$ – вероятность, с которой тема t встречается в документе d. Вычисление таких вероятностей для всех $t \in T, w \in W, d \in D$ равна вычислению матриц:

$$\Phi = p(w|t)_{W \times T} \quad (1.5)$$

$$\Theta = p(t|d)_{T \times D} \quad (1.6)$$

Такие матрицы называются матрицами терминов-тем (1.5) и тем-документов (1.6). Таким образом, задача тематического моделирования сводится к оценке параметров $\phi_{wt} = p(w|t)$ и $\theta_{td} = p(t|d)$ по коллекции D. Это задача стохастического матричного разложения матрицы F терминов-документов на произведение матриц Φ терминов-тем и матрицы Θ тем-документов (Рис. 1.7):

$$F_{W \times D} \approx \Phi_{W \times T} \times \Theta_{T \times D} \quad (1.7)$$

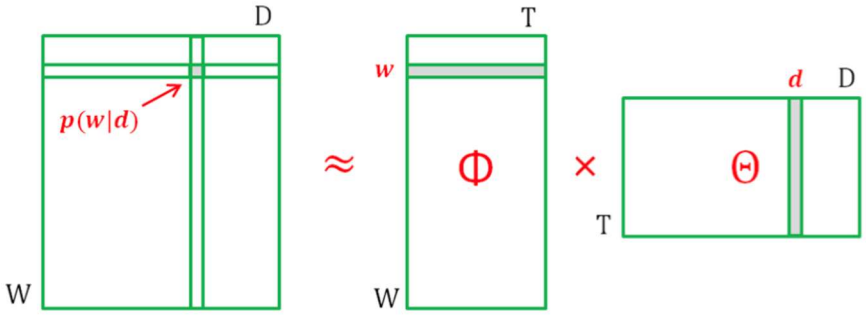

Рис. 1.7 Стохастическое матричное разложение матрицы терминов-документов

Задача оценки параметров $\phi_{wt} = p(w|t)$ и $\theta_{td} = p(t|d)$ решается путем максимизации логарифма правдоподобия, с условием нормировки столбцов матрицы Φ и строк матрицы Θ и неотрицательности всех элементов этих матриц, а также с дополнительным компонентом – регуляризацией[3]:

$$\mathrm{L}(\Phi,\Theta) = \sum_{d\in D}\sum_{w\in d} n_{dw} \ln \sum_{t\in T} \phi_{wt}\theta_{td} + \sum_{i} \tau_i R_i(\Phi,\Theta) \to \max_{\Phi,\Theta} \qquad (1.8)$$

$$\sum_{w\in W} \phi_{wt} = 1, \qquad \phi_{wt} \geq 0, \qquad \sum_{t\in T} \theta_{td} = 1, \qquad \theta_{td} \geq 0,$$

где $\tau_i \geq 0$ – коэффициент регуляризации.

Методом решения данной задачи является итерационный EM-алгоритм, который представляет собой метод простых итераций для решения системы нелинейных уравнений.

$$\begin{cases} p_{tdw} = p(t|d,w) = \underset{t\in T}{\text{norm}}(\phi_{wt}\theta_{td}) \\ \phi_{wt} = \underset{w\in W}{\text{norm}}\left(\sum_{d\in D} n_{dw}p_{tdw} + \phi_{wt}\frac{\partial R(\Phi,\Theta)}{\partial \phi_{wt}}\right) \\ \theta_{td} = \underset{w\in W}{\text{norm}}\left(\sum_{w\in W} n_{dw}p_{tdw} + \theta_{td}\frac{\partial R(\Phi,\Theta)}{\partial \theta_{td}}\right), \end{cases} \quad (1.9)$$

где $\underset{t\in T}{\text{norm}}(x_t) = \frac{\max\{x_t,0\}}{\sum_{s\in T}\max\{x_t,x_s\}}$ – операция нормирования вектора.

E-шаг (expectation) – первое уравнение системы. Для каждой пары термин-документ вычисляется условная вероятность на основании значений, вычесленных не предыдущем M-шаге.

M-шаг (maximization) представлен на втором и третьем уравнении. На данном этапе вычисляются новые значения параметров ϕ_{wt} и θ_{td} [4].

В качестве дополнительного слагаемого в функционале используется два вида регуляризаторов: разреживания и декоррелирования.

Регуляризатор разреживания матрицы тем-документов Θ формализует так называемую гипотезу разреженности, состоящую в том, что каждый документ относится к малому количеству тем.

Регуляризатор декоррелирования формализует предположение о различности тем, как распределений на множестве токенов.

Качество модели может быть рассчитано с помощью перплексии[5]:

$$\mathcal{P}(D,p) = \exp\left(-\frac{1}{n}\mathrm{L}(\Phi,\Theta)\right) = \exp\left(-\frac{1}{n}\sum_{d\in D}\sum_{w\in d} n_{dw}\ln p(w|d)\right) \quad (1.10)$$

Модель тем лучше, чем меньше ее перплексия.

1.2. ВЫБОР ОПТИМАЛЬНОЙ МОДЕЛИ

Очень важной задачей данной работы был выбор наиболее эффективной модели векторизации документа. К сожалению, размеченную коллекцию

данных (например пар схожих документов) найти не удалось. Поэтому было принято решение собрать собственную коллекцию документов, посвященных сфере вычислительного мтериаловедения, с ресурса arxiv.org.

	pdf_key	title	author	journal	year	abstract	url	memory (Byte)
0	14113136v1.pdf	Shannon Entropy and Many-Electron Correlations...	['Luigi Delle Site']	International Journal of Quantum Chemistry	2014-11-12T10:59:24Z	In this paper I will discuss the overlap betwe...	https://export.arxiv.org/pdf/1411.3136v1	223318
1	191206192v2.pdf	Investigating solvent effects on the magnetic ...	['Loïc Halbert', 'Małgorzata Olejniczak', 'Val...	International Journal of Quantum Chemistry	2019-12-12T20:19:22Z	We investigate the ability of mechanical and e...	https://export.arxiv.org/pdf/1912.06192v2	16801744
2	condmat0507292v1.pdf	Interaction induced magnetic field asymmetry o...	['Markus Buttiker', 'David Sanchez']	International Journal of Quantum Chemistry	2005-07-13T08:50:28Z	We demonstrate that the nonlinear I-V characte...	https://export.arxiv.org/pdf/cond-mat/0507292v1	186070

Рис 1.8 Коллекция документов

На (Рис. 1.8) показан пример элементов коллекции. Для каждого документа имеется список авторов, дата публикации, журнал статьи, название и объем занимаемого пространства на диске. Размер коллекции – 3752 документа.

1.2.1. Решение подзадачи

На базе коллекции необходимо было решить так называемую «подзадачу».

Позадача (downstream task) - задача контролируемого обучения, в которых используется предварительно обученная модель или компонент.

В данном случае предварительно обученная модель – это один из методов векторизации текста, представленных в предыдущей подглаве, а задача контролируемого обучения – задача классификации. Предполагается, что модель должна уметь определять по документу о каком химическом элементе идет речь.

Был разработан алгоритм решения задачи выбора оптимальной модели:

1. Загрузка коллекции с ресурса.
2. Произвести предобработку текста – токенизация, лемматизация, удаление стоп-слов.

3. С помощью регулярных выражений обнаружены химические элементы и их синонимы (а именно – Al, He, O2). Осуществляется автоматическая разметка коллекции по наличию в документе элемента. Далее элемент удаляется из документа.
4. Для каждой модели векторизации производится обучение для получения векторных представлений документов (кроме BERT) и далее решается задача классификации. Используемая модель – логистическая регрессия. Количество предобученных моделей – 18. Средний размер выборки – 500 документов. Тренировочная выборка – 80%, тестовая – 20%.

1.2.2. Результаты моделирования

Результаты моделей могут оцениваться с помощью критериев Precision, являющейся мерой точности поиска, и Recall, являющейся мерой полноты. Данные метрики имеют следующий вид:

$$\text{Precision} = \frac{TP}{k} \tag{1.11}$$

$$\text{Recall} = \frac{TP}{P}, \tag{1.12}$$

где TP $(True\ Positive)$ – количество релевантных документов в поисковой выдаче, P $(Positive)$ – количество релевантных документов во всей коллекции.

В качестве метрики была использована агрегированная оценка качества поиска, называемая F1-мера , представляющая собой среднее гармоническое между Precision@ и Recall@:

$$F_1 = \frac{\text{Precision} + \text{Recall}}{2 \cdot \text{Precision} \cdot \text{Recall}} \tag{1.13}$$

В итоге были получены результаты на тестовой выборке для каждой модели, представленные в (Таблице 1.1):

Таблица 1.1 Качество поиска для различных моделей

Модель \ Хим. элемент	O2	He	Al
Word2Vec	0.55	0.59	0.72
Doc2Vec	0.69	0.71	0.81
FastText	0.68	0.7	0.85
TF-IDF	0.75	0.74	0.86
Bert	0.46	0.32	0.67
APTM	0.72	0.78	0.89

Наиболее эффективной моделью при решении подзадачи оказалась APTM. Подробнее об обучении данной модели будет рассказано в практической части.

1.3. ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОМУ ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

Разрабатываемый программный комплекс должен обеспечить необходимый функционал для пользователя для операций ввода/вывода.

Ключевая задача заключается в успешной интеграции данного комплекса с уже существующим Crystal-search. В настоящее время комплекс кафедры осуществляет лишь полнотекстовой поиск по словам или словосочетаниям. Соответственно задача состоит в добавлении рекомендаций семантически близких документов к просматриваемой статье.

1.3.1. Сценарии использования

Программное средство должно поддерживать определенные сценарии использования.

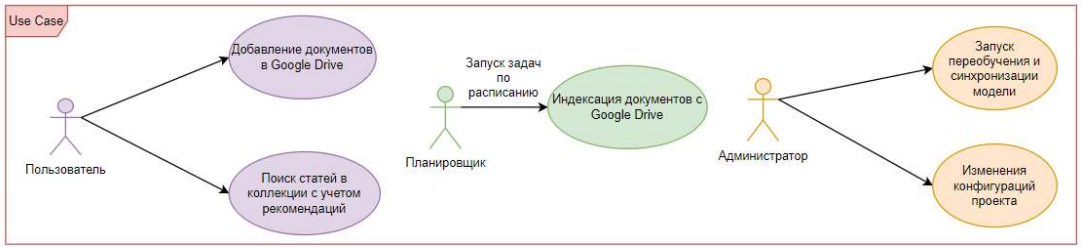

Рис. 1.9 Сценарии использования

На (Рис. 1.9) первым сценарием является работа обычного пользователя с программным комплексом. Соответственно, это рекомендации статей для пользователя, а также возможность добавления собственных документов и статей на английском языке в формате pdf. Ранее возможность добавления в комплексе в данном сценарии отсутствовала.

Google Drive – это сервис хранения, редактирования и синхронизации файлов, разработанный компанией Google. Его функции включают хранение файлов в Интернете, общий доступ к ним и совместное редактирование[6]. Данный инструмент прекрасно знаком и понятен специалистам и студентам. Соответственно его использование позволит комфортно и привычно производить процедуру ввода информации в комплекс.

Второй сценарий – это планировщий задач. Данный сценарий позволяет запускать интегрируемый модуль по расписанию или вручную с помощью одной команды, что очень удобно в рамках работы с серверным приложением в системе Linux.

Третий сценарий – интерфейс для работы администратора. Помимо изменений конфигурационных файлов, администратор также должен иметь возможность запускать режим переобучения модели при необходимости, например количество данных в базе превысило некий объем, а также производить синхронизацию коллекции при первичном запуске программного средства для корректного заполнения баз данных. Об этом будет более рассказно в практической части работы.

1.3.1. Серверное приложение

В качестве программного средства для реализации серверного приложения был выбран Python 3.8.

Python – интерпретируемый язык высокого уровня, то есть программируя на нем, не надо задумываться о деталях взаимодействия с памятью ЭВМ. Данный язык программирования использует строгую динамическую типизацию. Перечисленные характеристики языка позволяют повышать качество работы разработчика за счет возможности писать понятный код, абстрагированный от лишних низкоуровневых подробностей, а также за счет кроссплатформенности, то есть отсутсвии необходимости переписывать код, при наличиии потребности его запуска на другой платформе. Python полностью поддерживает основные парадигмы объектно ориентрированного программирования: полиморфизм, наследование и инкапсуляцию, следует принципу “все являеся объектом”. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации [7]. Среди основных областей применения данного языка можно выделить следующие: разработка серверных приложений, написание различного рода скриптов (удобно, с учетом того, что язык является интерпретируемым), машшиное обучение и анализ данных.

Основная причина выбора именно этого языка – наличие выбора большгого количество реаизованных библиотек для работе в сфере машинного обучения, работы с базами данных и сторонними программными интерфейсами, простота развертывания проекта на сервере, а также минималистичный синтаксис.

1.3.2. Хранение данных

Двумя основными направлениями в области хранения данных является реляционные и нереляционные (NoSql) базы данных.

Характерной особенностью реляционных баз данных является возможность на уровне базы данных указать связи между структурами данных(таблицами) для работы с коррелирующими элементам информации. Как следует из названия, данный тип баз данных базируется на реляционной теории — предствлении данных в так называемых таблицах. Каждая строка в подобной таблице реляционной базы данных, является записью, содержащей уникальный идентификатор, так нзываемый первичный ключ. Столбцы таблицы содержат атрибуты данных, а каждая запись, как правило, включает значение для каждого атрибута. Данная особенность позволяет устанавливать корреляцию между таблицами базы данных.

Нереляционная база данных – тип базы данных, для которой не характерно, по сравнению с большинством традиционных типов баз данных, использование табличной схемы хранения данных. В данном типе базы данных применяется способ хранения данных, оптимизированный под конкретные потребности хранимых данных. К примеру, формат данных может является словарем, хранящем записи в виде пар "ключ — значение", также могут использоваться документы в виде JSON(документные базы данных) или граф, состоящий из ребер и вершин [8].

В данном проекте для работы с реляционными базами данных была выбрана СУБД PostgreSql. В ней хранятся данные такого типа как логи, описание коллекции документов, статусы и результаты моделирования.

PostgreSql – это свободно распростроняемая система управления базами данных объектно-реляционного типа [9]. Выбор пал именно на эту СУБД из-за простоты ее эксплуатации и использования, а также из-за возможности масштабирования на текущем кластере.

Для хранения текста была выбрана нереляционная база данных – ElasticSearch.

ElasticSearch - это высокомасштабируемая распределенная поисковая система полнотекстового поиска и анализа данных, работающая в режиме реального времени. Предоставляет возможность хранить, производить поиск

и анализ по большим объемам данных. Обычно используется в качестве базового механизма/технологии, помогая приложениям со сложными функциями поиска [10].

2.ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1. Архитектура комплекса

Реализуемый программный комплекс, должен стать частью существующего решения Crystal-search. Он представляет из себя дополнительный сервис по работе со входящим потоком документов и их обработке и ранжированию (Рис. 2.1):

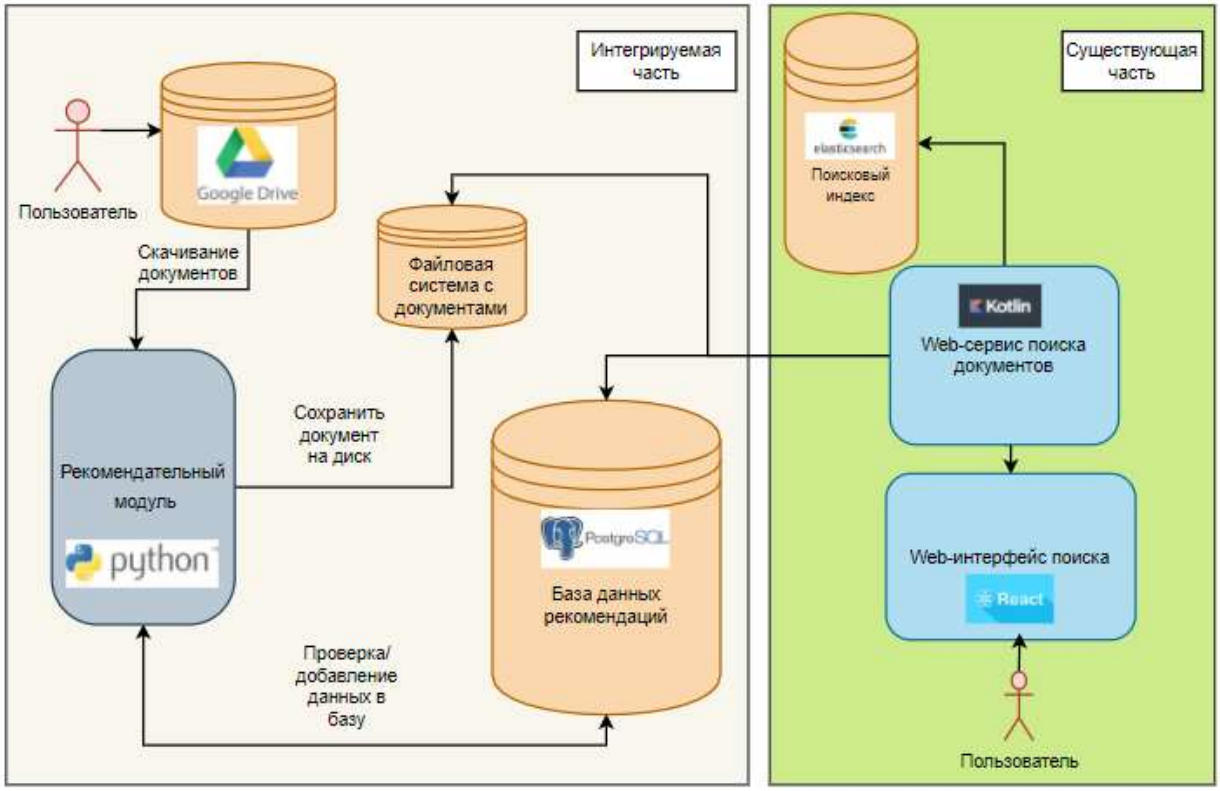

Рис. 2.1. Архитектура платформы

Описание интегрируемого программного средства:

- Google Drive – облачное хранилище для документов. Используется в качестве входного потока документов (в формате pdf).
- Рекомендательный модуль (Python) – модуль по работе с моделью векторизации документов. Релаизуемый функционал: обработка, запись в базу, ранжирование документов на основе модели.
- База данных рекомендаций (PostgreSql) – база для хранения логов, информации по текущим документам коллекции, статусов обработки моделями, ранжированных пар документов коллекции.

- Файловая система с документами – серверное хранилище исходных документов. Служит в качестве связующего звена двух компонент итогового комплекса.

Описнание архитектуры текущего программного комплекса Crystal-search:

- Поисковой индекс (ElasticSearch) – компонент для индексирования документов, а также их хранения. Позволяет производить полнотекстовой поиск.
- Web-сервис поиска документов (Kotlin) – сервис для работы с PostgreSql и ElasticSearch и поставки данных до веб-интерфейса, а именно – результатов поисковых выдач, а также рекомендаций к ним.
- Web-интерфейс поиска (React) – графический интерфейс для работы конечно пользователя с программным комплексом.

2.2. Обучение тематической модели

Как уже упоминалось, для реализации модели тематического моделирования использовалась библиотека BigARTM, реализованная на C++ и Python. Для обучения данной модели использовалась коллекция данных, полученная с ресурса arxiv.org. Был создан специальный пакет, также реализованный на языке Python по взаимодействую с данным ресурсом, не входящий в архитектуру программного комплекса. Данный пакет обеспечивает возможность скачивания с портала коллекции документов по выбранному научному журналу или конференции напрямую на Google Drive.

```
{
    {
        "SCOPES": ["https://www.googleapis.com/auth/drive"] (different scopes for GD API),
        "SERVICE_ACCOUNT_FILE" : path-to-GD-service-account-credentials,
        "api_folder_path": id-of-folder-in-GD,
        "data_folder" : path-to-temp-folder,
        "csv_name": csv-name-with-meta-data,
        "query" : [query-to-arxiv-api<journal, all etc.>]
    }

}
```

Рис. 2.2 Конфигурационный файл пакета

Как видно на (Рис. 2.2), в поле «query» может быть записана последовательность научных журналов или конференций. Как уже упоминалось, было скачано 3752 научных статей, относящихся к сфере вычислительного материаловедения. Далее следовала итерация обработки данных. Стоит отметить, что с помощью бибилотеки Pandas, получилось релаизовать данную обработку в несколько потоков, что существенно уменьшило время выполнения.

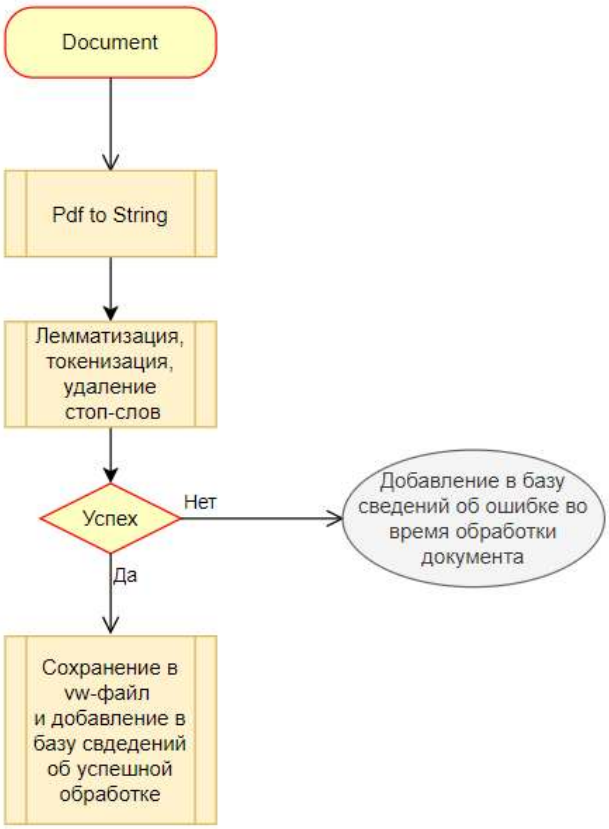

Рис. 2.3 Обработка документа

На (Рис. 2.3) продемонстрированы ключевые этапы обработки входного документа. Фаза «Pdf to String» конвертирует документ из формата pdf в строку с помощью Python – бибилотеки Fitz. Далее осуществляется лемматизация, токенизация и удаление стоп-слов. В случае ошибки во время данной иетрации информация отправляется в базу данных для последующей ручной обработке специалистом. В случае успеха, также отправляется информация уже об успешной обработке, а также полученная строка помещается в специальный файл с расширением .vw.

VW файл – текстовой файл, содержащий в себе строковые представления документов следующего формата, который представлен на (Рис. 2.4)::

```
doc1 this text will be processed not as bag of words | Some_Author
```

Рис. 2.4 VW-файл

Далее производится инициализация и непосредственное обучение модели.

```
model_artm = artm.ARTM(num_topics=NUM_TOPICS,
                       num_processors=THREADS,
                       theta_columns_naming='title',
                       # show_progress_bars=True,
                       theta_name="prod_theta",
                       class_ids={'@modal': 1},
                       cache_theta=True
                       )
dictionary = artm.Dictionary()
dictionary.gather(data_path=batch_vectorizer_mono.data_path)
dictionary.filter(min_df_rate=0.01, min_tf=10, inplace=True)
model_artm.initialize(dictionary=dictionary)
logger.info("Dictionary stage completed")
model_artm.scores.add(artm.SparsityPhiScore(name='sparsity_phi_score', class_id='@modal'))
model_artm.scores.add(artm.SparsityThetaScore(name='sparsity_theta_score'))
model_artm.scores.add(artm.TopTokensScore(name='top_tokens_score', class_id="@modal"))
model_artm.scores.add(artm.PerplexityScore(name='perplexity_score', class_ids={'@modal': 1}))
model_artm.regularizers.add(
    artm.DecorrelatorPhiRegularizer(name='decorrelator_phi_lab', tau=1.0e+5, class_ids=['@modal']))

model_artm.num_document_passes = 1
ITERATIONS = [8, 8, 8]
logger.info("Model initialization stage completed")
model_artm.fit_offline(batch_vectorizer=batch_vectorizer_mono, num_collection_passes=ITERATIONS[0])
logger.info("1/3 trainig stage")
model_artm.regularizers.add(artm.SmoothSparseThetaRegularizer(name='sparse_theta_regularizer', tau=-1.5))
model_artm.fit_offline(batch_vectorizer=batch_vectorizer_mono, num_collection_passes=ITERATIONS[1])
logger.info("2/3 trainig stage")
model_artm.fit_offline(batch_vectorizer=batch_vectorizer_mono, num_collection_passes=ITERATIONS[2])
logger.info("3/3 trainig stage")
logger.info("Training is completed")
```

Рис. 2.5 Фрагмент обучения модели

На (Рис. 2.5) представлен цикл обучения модели. Инициализируется объект ARTM, производится фильтрация словаря, добавление регуляризатора декоррелирования. Было произведено множество процессов обучения и наименьшую перплексию показал следующий алгоритм обучения:

1. Длина векторного представления документа - 100
2. 8 итераций обучения EM-алгоритма с добавлением регуляризатора декоррелирования (τ = 10e+5)
3. 16 итераций с добавлением регуляризатора разреживания (τ = 1.5)

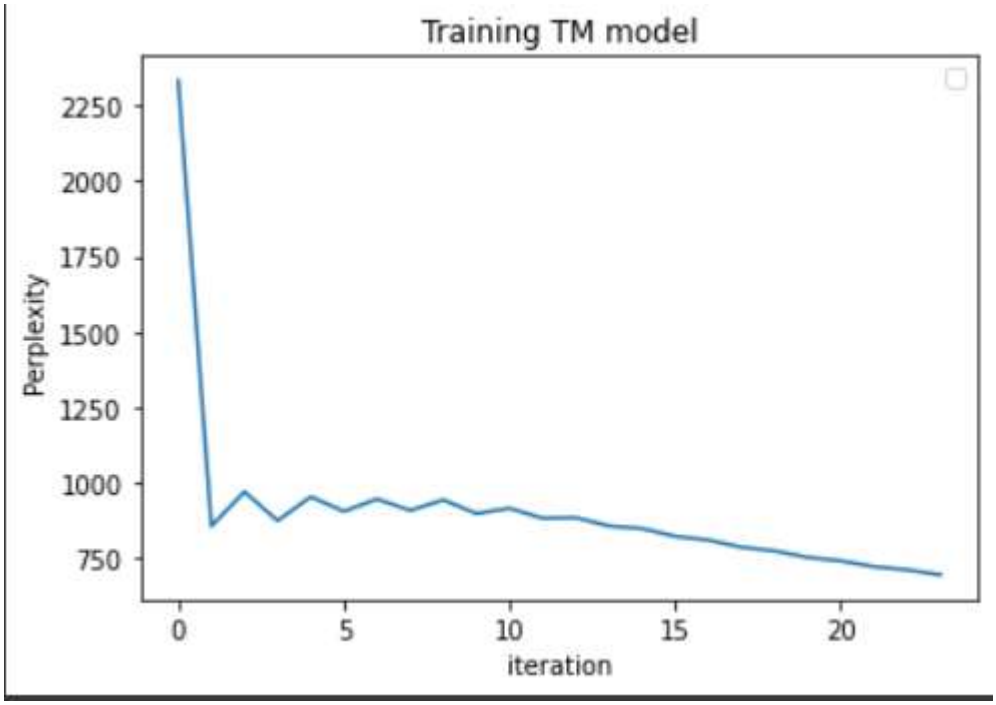

Рис. 2.6 Перплексия

На (Рис. 2.6) изображена зависимость перплексии от количества итераций. После 24 итерации изменение перплексии ничтожно мало. Значение перплексии, равное 746 – минимальное, среди всех исследований.

В качестве обученной модели, подразумевается матрица тем-документов, представленная на (Рис. 2.7):

	14113136v1.pdf	191206192v2.pdf	condmat0507292v1.pdf	condmat0602455v1.pdf	physics0503116v3.pdf	physics0503180v1.pdf	quantph9511040
topic_0	0.012528	0.000000	0.007582	0.002970	0.018870	0.009005	0.
topic_1	0.007563	0.000000	0.010568	0.006050	0.009668	0.005615	0.
topic_2	0.004831	0.000000	0.012146	0.010333	0.006307	0.009031	0.
topic_3	0.012412	0.000000	0.008135	0.002688	0.016552	0.011078	0.
topic_4	0.005378	0.376917	0.018256	0.001826	0.000000	0.000171	0.
...	...	...	...	...	...	...	
topic_95	0.008390	0.000000	0.007018	0.010297	0.016725	0.015070	0.
topic_96	0.006905	0.000000	0.012191	0.006903	0.007306	0.010180	0.
topic_97	0.009416	0.000000	0.009493	0.004527	0.011927	0.010958	0.
topic_98	0.007324	0.000000	0.011810	0.008349	0.006520	0.007379	0.
topic_99	0.011912	0.000000	0.008888	0.004729	0.013790	0.009984	0.

100 rows × 3685 columns

Рис. 2.7 Матрица тем-документов

При транспонировании данной матрицы мы получаем векторные представления для каждого документа в обучающей коллекции.

2.3. Разработка серверного приложения

В реализуемом приложении на серверную часть возложена следующие обязанности: обрабатывать файлы, находящиеся на Google Drive, обрабатывать документы, получать векторные представления документов, находить пары схожих документов и заносить необходимую информацию в базу данных.

Как уже упоминалось, существует несколько режимов работы: администраторский и пользовательский.

Для администратора реализована возможность переобучения и первичной синхронизации данных. Это методы – retrain и synchronize, соответственно.

Также у администратора есть возможность с помощью конфигурационного файла произвести настройку параметров системы.

Перечень параметр:

- Размер тематического вектора
- Количество потоков во время переобучения модели
- Ограчниение по количеству хранения схожих документов
- Минимальное значение косинусной меры для добавления в таблицу схожестей

Далее рассмотрим стандартную обработку потока данных.

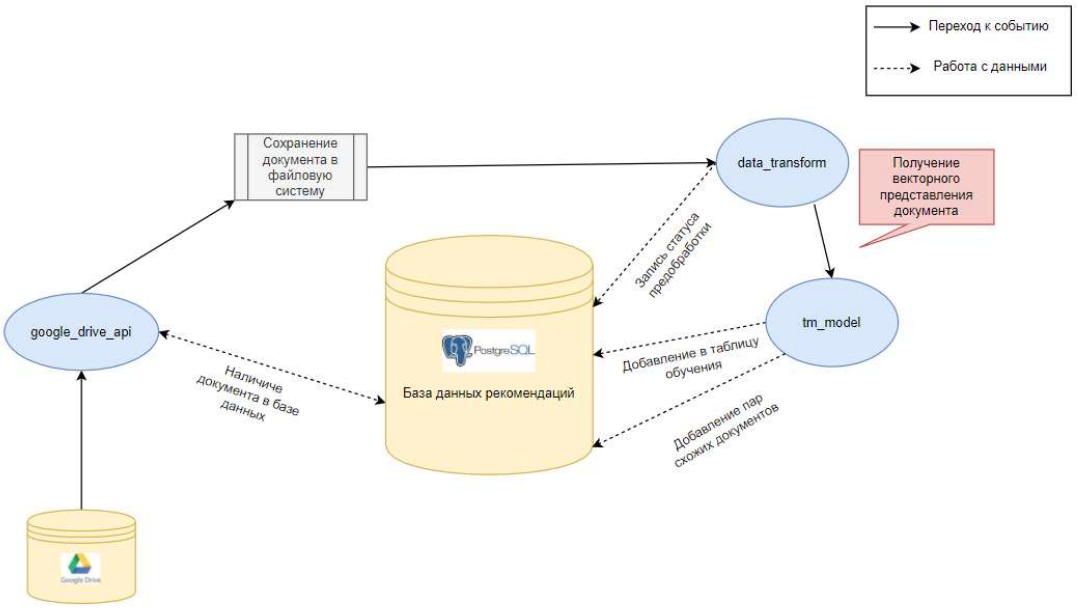

Рис. 2.8 Поток данных

Алгоритм обработки входящих документов продемонстрирован на (Рис. 2.8). Далее перечислены наиболее важные этапы работы серверного приложения:

1. С помощью пакета google_drive_api файлы с Google Drive перемещаются в файловую систему. Осуществляется проверка даты добавления файла на диск и существования документа в базе.
2. Пакет data_transform обрабатывает каждый документ и конвертирует из формата PDF в строковое представление. В случае ошибки данные передаются в базу с флагом «Ошибка». Иначе, файл добавляется в базу с флагом «Успех».
3. Далее следует самый важный этап – добавление строкового представления в модель и получения тематического вектора. Название файла добавляется в таблицу обученных документов.
4. Финальный этап – нахождение схожих ко входящему документов коллекции и занесение этой информации в таблицу пар схожих документов.

Также был создан функционал поддержки логирования с перенаправлением потока вывода в текстовой файл. Реализованная серверная

часть имеет удобный функциональный интерфейс, который сможет продолжить другой программист. Также пданный функционал предусматривает добавление иных методов получения веткорных представлений.

2.4 Интеграция с комплексом Crystal-search

Как видно на Рисунке 2.1, две части программного комплекса имеют две связные точки – файловую систему и базу данных рекомендаций. В файловой системе хранятся все документы, синхронизируемые с Google Drive. Они будут использоваться и для рекомендательной системы и для полнотекстового поиска с помощью ElasticSearch. Собственно, одной из целей данного проекта являлось предоставление конечному пользователю рекомендаций документов во время показа поисковой выдачи. Данные по семантической близости документов хранятся в базе данных на базе PostgreSql. В комплексе Crystal-search уже предусмотрен метод по считыванию данных из базы и их показу конечному пользователю в Web – интерфейсе.

2.5. Хранение данных

В качестве средства управления базой данных была выбрана PostgreSql за счет простоты реализации табличных схем, возможности индексирования данных, а также – масштабирования и репликации. Схема базы данных представлена на (Рис. 2.9):

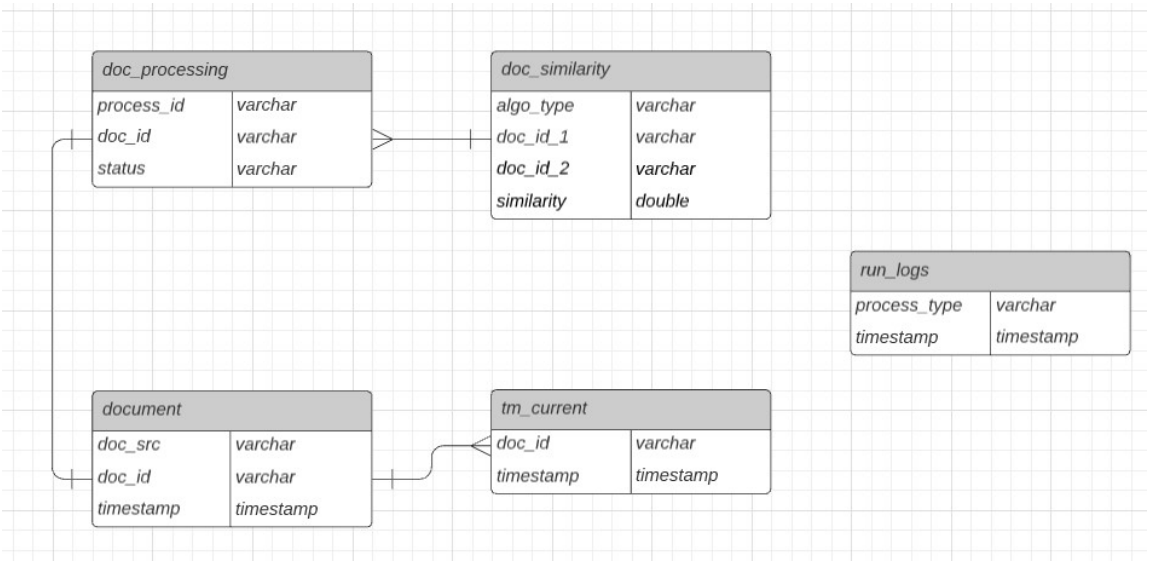

Рис. 2.9 Схема базы данных

Для работы с базой данных, как уже упоминалось был разработан модуль postgres_api.

Фунционал компонента postgres_api:

1. Сохранение названий документов, даты появления на Google Drive, статусов обработки.
2. Сохранение пар семантически близких документов, их косинусного расстояния.
3. Проверка существования в базе имени документа.
4. Сохранение данных о запусках проекта.
5. Подключение к базе данных и разрыв соединения.
6. Очистка таблицы.

Далее приводится перечень таблиц, а также их целевое назначение:

- Tm_current – перечень документов, входящих в обучающую коллекцию тематической модели. Обновляется после запуска администратором переобучения.
- Document – перечень документов находящихся в файловой системе и дата их сохранения.
- Doc_processing – информация о статусе докумнта во время предобработки и при использовании моделью.

- Doc_similarity – таблица пар сходих документов.
- Run_logs – содержит историю запусков программного средства.

Данная структура позволяет успешно интегрировать дополнительные модели векторизации текста в существующий комплекс.

2.6. Пример результата работы программного средства

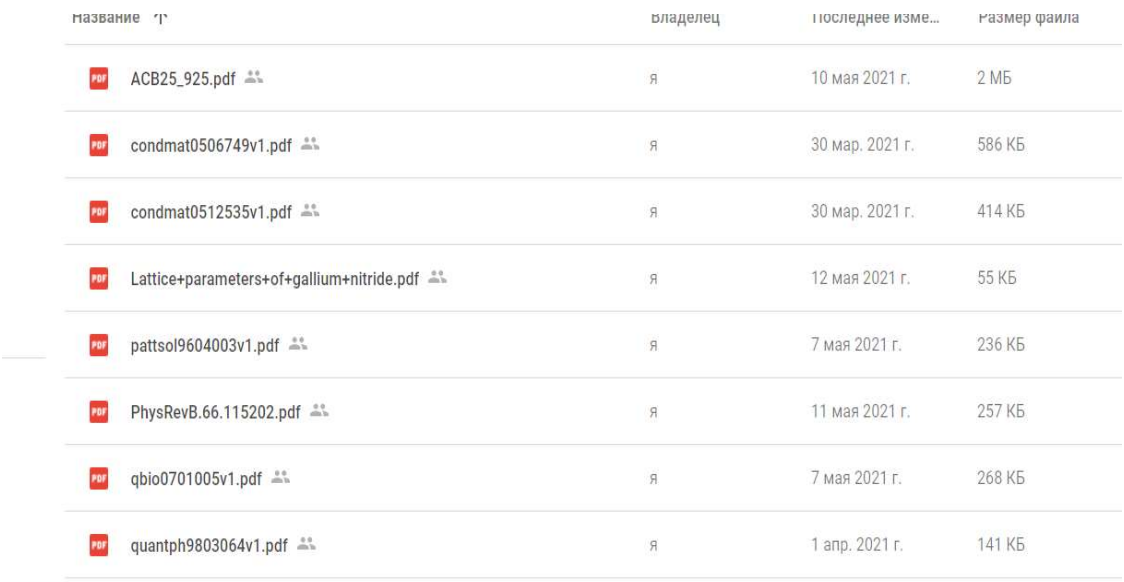

Рис. 2.10 Сохранения данных на Google Drive

На (Рис. 2.10) изображен пример сохранения научных статей на диск в формате PDF.

	algo_type character varying	doc_id_1 character varying	doc_id_2 character varying	similarity double precision
1	TM	Lattice+parameters+of...	151102305v1.pdf	0.600046790947024
2	TM	151102305v1.pdf	Lattice+parameters+of...	0.600046790947024
3	TM	Lattice+parameters+of...	condmat0507587v1.pdf	0.6007119810525342
4	TM	condmat0507587v1.pdf	Lattice+parameters+of...	0.6007119810525342
5	TM	Lattice+parameters+of...	07103088v1.pdf	0.6012732308911711
6	TM	07103088v1.pdf	Lattice+parameters+of...	0.6012732308911711
7	TM	Lattice+parameters+of...	condmat0601035v2.pdf	0.6014413961099255
8	TM	condmat0601035v2.pdf	Lattice+parameters+of...	0.6014413961099255
9	TM	Lattice+parameters+of...	07104542v2.pdf	0.601738375757237
10	TM	07104542v2.pdf	Lattice+parameters+of...	0.601738375757237

Рис. 2.11 Пары схожих документов

В качестве результата, приведен пример таблицы Doc_similarity (Рис. 2.11), в которой представлены пары наиболее схожих документов. Прототип итогового программного комплекса выходит за рамки данной работы, но тем не менее будет представлен на защите проекта. Он заключается в реализации Web-интерфейса, в котором предусматриваются следующие компоненты:

- Скачивание рекомендаций
- Градиентная окраска в замисимоти от значение косинусной меры
- Показ названий файлов рекомендаций к конкретномудокменту

ЗАКЛЮЧЕНИЕ

В рамках данной работы были выполнены следующие задачи:

- Собрана коллекция документов с помощью реализованного программного модуля
- Изучены и протестированы модели векторизации текстовых документов
- Реализован модуль для работы с моделями векторизации документов
- Реализована база данных для сохранения документов и мониторинга процессов
- Реализован модуль по работе с Google Drive
- Реализованы стратегии и методы отказоустойчивости проекта
- Произведена интеграция с существующим программным комплексом Crystal-search
- По теме диссертации приняты к публикации тезисы в сборнике трудов Международной научно-технологической конференции студентов и молодых ученых «Молодежь. Инновации. Технологии»

Стоит также выделить перспективы развития данного программного средства:

- Реализация модулей в виде микросервисов.
- Добавление возможности поиска по предложенным темам, полученных в результате обучения модели.
- Добавление возможности удаления файлов из модели.
- Интеграция других моделей векторизации текста.

Исходный код программного средства можно найти по ссылке: https://github.com/baumanproject/Topic-Modeling-Recommendation-System

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina (11 October 2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". arXiv:1810.04805v2

[2] Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space // In Proceedings of Workshop at ICLR. — 2013a

[3] Bengio Y., Ducharme R., Vincent P. A neural probabilistic language model // In Journal of Machine Learning Research. — 2003.

[4] Vorontsov K. V., Frei O. I., Apishev M. A., Romov P. A., Suvorova M. A. BigARTM: Open Source Library for Regularized Multimodal Topic Modeling of Large Collections // AIST'2015, Analysis of Images, Social Networks and Texts. Springer International Publishing Switzerland, 2015. Communications in Computer and Information Science (CCIS), pp. 330–395.

[5] Anastasia Ianina, Lev Golitsyn, and Konstantin Vorontsov. Multi-objective topic modeling for exploratory search in tech news. In Communications in Computer and Information Science, pages 181–193. Springer International Publishing, nov 2017.

[6] Google Drive, официальный сайт с документацией по реализации [Электронный ресурс], - режим доступа: https://developers.google.com/drive

[7] Python, официальный сайт [Электронный ресурс], - режим доступа: https://www.python.org

[8] Dan McCreary, Ann Kelly. Making Sense of NoSQL: A guide for managers and the rest of us. — Manning Publications, 2013. — 312 p.

[9] PostgreSql, официальный сайт [Электронный ресурс], - режим доступа: https://www.postgresql.org/

[10] ElasticSearch, официальный сайт [Электронный ресурс], - режим доступа: https://www.elastic.co/