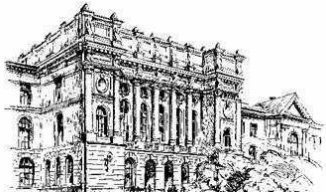
Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий



ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Разработка модуля рекомендаций онлайн-курсов для систем поддержки образовательного процесса

Студент гр. 3530901/60202 Е.С. Шумеева

Санкт-Петербург
2020

Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Работа допущена к защите
директор ВШИСиСТ
_____ В.М. Ицыксон
«____» _____ 2020 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Разработка модуля рекомендаций онлайн-курсов для систем поддержки образовательного процесса

по направлению 09.03.01 «Информатика и вычислительная техника»
по образовательной программе
09.03.01_02 «Технологии разработки программного обеспечения»

Выполнил студент гр. 3530901/60202 _____ Е.С. Шумеева
Научный руководитель,
ст. преподаватель _____ А.В. Мяснов
Консультант по нормоконтролю,
к. т. н., доц. _____ А.Г. Новопашенный

Санкт-Петербург
2020

Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

УТВЕРЖДАЮ
директор ВШИСиСТ
_____ В. М. Ицыксон
«____» _____ 2020 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы студенту Шумеевой Екатерине Сергеевне, группа 3530901/60202

1. **Тема работы:** Разработка модуля рекомендаций онлайн-курсов для систем поддержки образовательного процесса.
2. **Срок сдачи студентом законченной работы:** 25 июня 2020 года
3. **Исходные данные к проекту (работе):**
   - Фрагмент базы данных вуза с описанием образовательных программ.
4. **Содержание работы (перечень подлежащих разработке вопросов):**
   - Анализ существующих методов построения рекомендательных систем.
   - Определение основных функций разрабатываемого модуля, формирование требований.
   - Обзор и выбор возможных подходов для реализации контентной фильтрации.

- Разработка модуля рекомендаций онлайн-курсов с помощью различных подходов.
- Апробация разработанных подходов на наборе тестовых данных.
- Анализ полученных результатов, выявление проблем и определение траектории дальнейшего развития разработки.

5. **Консультанты по работе:** Нестеров С. А.

6. **Дата выдачи задания:** 3 февраля 2020 года

Руководитель ВКР _____ А. В. Мяснов

Задание принял к исполнению «03» февраля 2020 года

Студент _____ Е. С. Шумеева

РЕФЕРАТ

На 52 с., 1 рисунок, 6 таблиц, 2 приложений

РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ, КОНТЕНТНАЯ ФИЛЬТРАЦИЯ, МООК-ПЛАТФОРМА, ЛЕММАТИЗАЦИЯ, TF-IDF, LDA

Выпускная квалификационная работа посвящена разработке модуля рекомендаций онлайн-курсов для систем поддержки образовательного процесса.

Рекомендательные системы — программы, которые предсказывают, какие объекты будут интересны пользователю, исходя из информации о его профиле и объектах. Тема рекомендательных систем активно исследуется последние десятилетия [1]. Она широко применима на практике, в том числе в коммерции, что значительно стимулирует ее развитие, как в плане теоретических исследований, так и в виде практических задач.

В ходе работы был выполнен анализ методов построения рекомендательных систем и существующих решений в этой сфере. При разработке системы были проанализированы два различных подхода в области контентных рекомендательных систем. Результатом выполнения работы является разработанный модуль, подбирающий рекомендации исходя из анализа образовательных программ

THE ABSTRACT

52 pages, 1 pictures, 6 tables, 2 appendicies

RECOMMENDER SYSTEM, CONTENT-BASED FILTERING, MOOC PLATFORMS, LEMMATIZATION, TF-IDF, LDA

Bachelor's thesis is dedicated to the development of online-course reccomendation module for the educational support systems.

Reccomendation system is a program capable of predicting user's preference objects, basing it's predictions on his profile and objects. The issue of reccomendation systems has been widely investigated last decades [1], because recommendation systems can be applied very often. Possibility of using them in commerce significantly stimulates their development both in theoretical researches and in practical tasks.

During the work recommendation system constructing methods and already existing solutions in this question were analyzed. Two different approaches in the field of content recommendation systems were analyzed when developing the system. Developed module, selecting recommendations on the basis of educational program analysis, is the result of the work.

СОДЕРЖАНИЕ

СПИСОК ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

API	Application Programming Interface
LDA	Latent Dirichlet Allocation
TF-IDF	TF — term frequency, IDF — inverse document frequency
URL	Uniform Resource Locator
БД	База данных
MOOK	Массовые открытые онлайн-курсы

ВВЕДЕНИЕ

Онлайн-обучение или e-learning — обучение с применением мультимедиа и интернет-технологий. В современном мире данный формат образования становится все более популярным. При этом самым быстро развивающимся на сегодня сектором онлайн-обучения являются массовые открытые онлайн-курсы (или МООК), разрабатываемые частными лицами или учебными организациями для специализированных МООК-платформ.

Большинство МООК на крупных образовательных платформах не уступают по качеству традиционному оффлайн-обучению. За счет совместного использования словесных, наглядных и практических методов организации учебной деятельности улучшается качество восприятия информации студентами.

В нынешних реалиях от хорошего специалиста требуются основательные, глубокие познания в области своей деятельности. Для соответствия этим требованиям необходимо заниматься самообразованием. Одним из результативных способов является прохождение МООК. Согласно опросу 2014 г., проведенному среди пользователей прошедших какой-либо курс на платформе Coursera, 72 % респондентов получили определенный карьерный рост после окончания курса [4].

Поскольку студенты Политехнического университета являются будущими молодыми специалистами, расширение знаний является важной задачей и для нас. Однако в настоящий момент количество онлайн-курсов настолько велико, что выбрать оптимальный вариант исходя из своих потребностей является весьма сложной задачей. Решением этой проблемы может выступать рекомендательная система, выдающая релевантные рекомендации онлайн-курсов для конкретного студента.

1. МЕТОДЫ ПОСТРОЕНИЯ РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМ

В данном разделе рассматриваются три наиболее распространенных типа рекомендательных систем: контентно ориентированный подход (content-based), коллаборативная фильтрация (collaborative filtering) и гибридные системы.

1.1. Контентная фильтрация

Контентно ориентированные системы строятся на основе анализа описания объектов рекомендаций. Для использования данного типа рекомендательных систем необходимо иметь подробную информацию об объектах: чем больше информации мы имеем, тем точнее будет результат.

Основная идея таких рекомендательных систем заключается в том, что если человеку понравился объект, то вероятнее всего его заинтересуют и объекты схожие с ним.

В рамках контентной фильтрации выделяют два подхода:

- сопоставление непосредственно описаний объектов;
- сопоставление предварительно составленных профилей пользователя и объектов. Профиль пользователя формируется на основе атрибутов ранее просмотренных объектов.

При реализации контентной фильтрации выделяют три этапа:

1. предварительная обработка описаний;
2. приведение их к числовому виду;
3. фильтрация и получение рекомендаций.

1.2. Коллаборативная фильтрация

Коллаборативная фильтрация или же совместная фильтрация строится на основе анализа взаимодействия пользователей с объектами (матрицы полезности). Матрица полезности представляет собой таблицу, где строкам соответствуют пользователи, а столбцам - объекты. На пересечении находятся оценки, выставленные пользователями данным объектам.

Основная идея заключается в том, что если два пользователя имели одинаковые интересы в прошлом, то можно сделать предположение, что в будущем их предпочтения также будут совпадать. В отличие от контентной фильтрации, в данном подходе при построении рекомендаций анализируются лишь оценки объектов. Выделяют несколько подходов коллаборативной фильтрации:

- memory-based. В данном подходе для предсказания новой оценки выполняется попарное сравнение всех векторов статистически накопленных данных. В зависимости от предмета сравнения выделяют два подвида: подход основанный на пользователях (user-based) и на объектах (item-based). В user-based для предсказания новой оценки сравниваются вектора пользователей, в item-based - вектора объектов.

- model-based. При данном походе на начальном этапе выполняется построение модели данных с помощью машинного обучения, матричного разложения или вероятностных моделей, а затем на основании полученной модели формируются рекомендации.

1.3. Гибридные системы

Суть гибридных систем заключается в использовании нескольких различных подходов вместе. К примеру, совместное применение алгоритмов коллаборативной и контентной фильтрации. Благодаря этому устраняется большая часть недостатков, присущих каждому подходу в отдельности,

однако разработка такой системы имеет гораздо большую сложность. Существует несколько принципов сочетания различных подходов:

- коммутация. В зависимости от заданных критериев приоритет отдается тому или иному подходу;
- наделение весовыми коэффициентами. В вычислении результатов принимают участие все подходы, однако оказываемое влияние пропорционально их весу. При первом запуске все подходы наделяются одинаковыми весовыми коэффициентами, одна в дальнейшем они корректируются исходя из оценки прогнозируемых и реальных результатов;
- смешивание. Если пользователю можно демонстрировать достаточно большое число рекомендаций, то результаты различных подходов объединяются вместе, обеспечивая тем самым большее раскрытие темы.

Гибридные системы на сегодняшний день получили наибольшее распространение.

1.4. Резюме

По результатам исследования выделим основные особенности различных методов построения рекомендательных систем.

Коллаборативная фильтрация используется в системах с большим числом пользователей и большим объемом информации о взаимодействии пользователей с объектами рекомендаций. Данный подход позволяет получать качественные разносторонние рекомендации, однако имеется проблема при построении рекомендаций для новых пользователей или объектов.

Контентная фильтрация применяется в системах, где объекты имеют неструктурированные описания. В данном подходе отсутствует проблема "холодного старта" , однако полезность полученных рекомендаций ограничена.

Гибридные системы решают большинство проблем, присущих каждому подходу по отдельности, однако имеют большую сложность при построении.

Исходя из изученных материалов для построения нашей рекомендательной системы был выбран метод контентной фильтрации. Это объясняется рядом факторов:

1. в качестве исходных данных об объектах мы имеем неструктурированные данные;

2. описание интересов пользователя формируется исходя из анализа изучаемых предметов;

3. отсутствует информация о взаимодействии пользователей непосредственно с объектами рекомендаций;

Поскольку подбор рекомендаций будет выполняться относительно специальности студента, что уже представляет собой ограниченную предметную область, основной недостаток контентной фильтрации не является для нас существенной проблемой.

2. АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ

В данном разделе рассмотрим существующие решения для построения контентных рекомендательных систем. Проведем обзор функциональности, а также проанализируем их соответствие вводимым критериям.

2.1. Критерии сравнения

Для обзора существующих решений и их сравнения введем некие критерии, которые позволят нам это осуществить. В данном случае для анализа существующих решений реализации контентных рекомендательных систем целесообразно рассмотреть в качестве критериев следующие условия:

- построение на основе явного задания интересов пользователя;
- разграничение информации на данные для построения интересов; пользователя и собственно объектов рекомендаций;
- взаимодействие с внешними системами;
- локальное расположение сервиса.

Наиболее важным критерием является возможность системы генерировать рекомендации на основе статической информации об интересах пользователя. Действия пользователя имеют меньшую значимость, поскольку мы можем учитывать лишь переход пользователя по ссылке. Оценить другие действия пользователя не представляется возможным, поскольку они происходят уже на стороннем ресурсе. Переход по ссылке также является слабым параметром, поскольку относится к методам неявной обратной связи. Методы неявной обратной связи характеризуются неопределенностью в том, положительно или отрицательно повлиял конкретный акт обратной связи на степень предпочтения. Иными словами, по переходу

пользователя по ссылке невозможно сделать однозначное предположение о том, понравился ему рекомендованный курс или нет.

Также необходимо, чтобы в системе имелась возможность разграничивать данные. Профиль пользователя строится на основе анализа его образовательных программ, однако мы не должны их использовать в качестве объектов рекомендаций.

Необходимо, чтобы система имела возможность взаимодействия с внешними системами, поскольку объектами рекомендаций являются курсы со сторонних сайтов

2.2. Библиотека easyrec

Easyrec – опенсорсное веб-приложение, позволяющее интегрировать персональные рекомендации в веб-сайт. Доступ к easyrec обеспечивается через веб-сервисы, что обеспечивает простую и быструю интеграцию.

Одним из 4 поддерживающихся алгоритмов является калькулятор мер подобия профилей товаров. Для работы данного алгоритма требуются профили товаров с заданными атрибутами, предоставленные в строго заданном XML формате. Вычисление мер подобия атрибутов выполняется с использованием компараторов, а затем из индивидуальных мер подобия свойств выводится общая мера подобия товаров.

2.3. Recombee

Recombee позиционируется как мощный рекомендательный механизм на основе искусственного интеллекта. Данный сервис комбинирует в себе методы глубокого обучения, коллаборативной и контентной фильтрации. Поддерживается работа в режиме реального времени: после каждого взаимодействия пользователя с системой происходит корректировка модели обучения. Recombee имеет интуитивно понятный RESTful API и SDK для многих языков программирования, что упрощает интеграцию в проекты.

2.4. Плагин YARPP

YARPP является одним из самых популярных плагинов похожих постов для WordPress. В данном плагине имеется возможность настраивать параметры, которые будут учитываться при определении релевантности подбираемых постов. Также имеется возможность указать минимальный порог похожести: чем выше этот порог, тем более релевантными окажутся рекомендации. Имеются готовые шаблоны по оформлению отображаемого списка рекомендации.

2.5. Резюме

Среди рассмотренных готовых решений контентных рекомендательных систем нет ни одного, которое полностью удовлетворяло бы введенным критериям.

Алгоритм, реализованный в библиотеке easyrec, ориентирован на работу с четко формализированными и короткими описаниями атрибутов объектов.

В сервисе Recombee отсутствует возможность разграничения данных, в связи с чем образовательные программы также будут рассматриваться как перспективные объекты рекомендаций. Также данный сервис является платным.

Плагин YARPP внедряется непосредственно в сайт и не способен анализировать информацию со сторонних ресурсов. Кроме того, использование плагина YARPP возможно только в системе управления контентом WordPress.

В связи с этим можно сделать вывод, что для решения нашей задачи необходима разработка новой системы рекомендаций.

3. ПОСТАНОВКА ЗАДАЧИ

В данном разделе выдвигаются требования к разрабатываемому модулю, производится описание основных функций и накладываемых ограничений. Также выполняется анализ подходов к решению поставленной задачи.

3.1. Формулировка требований

В рамках выполнения работы требуется разработать модуль рекомендаций для систем поддержки образовательного процесса вуза. Рекомендации должны быть построены на основе анализа изучаемых студентом образовательных программ. Персонализация рекомендаций обеспечивается за счет введения вектора интересов пользователей. Критерии формирования вектора интересов пользователя выбираются на предусмотрение администратора системы. В качестве начальной установки было предложено подбирать рекомендации относительно предметов, по которым студент показал наилучшую успеваемость. Данный выбор объясняется тем, что высокая успеваемость студента в большинстве случаев свидетельствует о заинтересованности данной предметной областью.

3.2. Основные функции разрабатываемого модуля

В рамках реализации модуля рекомендаций необходимо решить следующие подзадачи:

- сбор данных с MOOK-платформ;
- очистка полученных данных;
- приведение описаний в некоторую числовую форму;
- сопоставление и отбор лучших N результатов;

3.3. Накладываемые ограничения

Контентные рекомендательные системы имеют сильную зависимость от проектов, в которых они используются, а также от объема доступных для анализа данных. В нашем случае можно выделить следующие факторы, накладывающие ограничения на точность подбора рекомендаций:

- объекты рекомендаций представляют собой ресурсы со сторонних сервисов. В связи с этим отсутствует возможность проследить действия пользователя после перехода по ссылке и дать явную оценку реакции пользователя на предоставленную рекомендацию.

- отсутствие возможности свободного выбора пользователями курсов. Иными словами, пользователь всегда имеет доступ только к ограниченному числу курсов, предложенными нашей рекомендательной системой. Это также обуславливается расположением курсов на сторонних сервисах и уменьшает значимость обратной связи.

- отсутствие детальной информации о содержании курсов. Посредством предоставляемого API возможно получить только информацию, представленную на стартовой странице курса. В ходе анализа было выявлено, что средняя длина описаний курсов после выполнения обработки составляет порядка 150 слов.

3.4. Анализ подходов к решению поставленной задачи

Как было сказано в главе 1, при построении контентных рекомендательных систем выделяют два подхода: сопоставление описаний объектов и сопоставление предварительно составленных профилей. Проанализируем данные подходы с точки зрения ограничений, изложенных в разделе 3.3.

Поскольку в нашей системе присутствуют лишь методы неявной обратной связи, а именно переход пользователя по ссылке, и значимость этого

вида связи ограничена, необходимо деликатно подойти к оценке взаимодействия пользователей с объектами рекомендаций. Можно выделить два варианта решения данной ситуации:

- не учитывать переходы пользователя по ссылкам и подбирать рекомендации исходя из начального задания вектора интересов;
- перестраивать вектор интересов пользователя после каждого просмотра рекомендации с предусмотрением возможности вернуться к начальному. состоянию

Для реализации первого подхода достаточно использовать сопоставление описаний объектов. В данном случае выполняется подбор наиболее близких курсов к каждому предмету в рамках учебного плана, а затем в зависимости от предметов, составляющих вектор интересов пользователя, формируется подбор рекомендаций для каждого студента. Однако в данном случае снижается гибкость настройки системы. Вектор интересов будет изменен либо при явном задании новых данных, либо при изменении критерия формирования.

Создание профилей пользователей и объектов позволяет реализовать второй подход к оценке обратной связи. В таком случае в начале разрабатывается некоторая система выделения атрибутов онлайн-курсов и учебных программ. Чаще всего при этом используется тематическое моделирование. При пропуске описания объекта через данную модель на выходе мы получаем распределение тем в объекте. Профиль пользователя в этом случае формируется как оценка заинтересованности выделенными темами. Дальнейший подбор рекомендаций выполняется исходя из сопоставления полученных профилей. Мы не можем судить об удовлетворенности пользователя рекомендацией исходя из одного перехода, однако проанализировав несколько последних взаимодействий, можно сделать вывод о преобладании переходов по рекомендациям с одинаковой доминирующей темой. Соответственно при такой ситуации значимость данной темы в профиле пользователя будет увеличена. Если же на протяжении нескольких следу-

ющих демонстраций пользователь не проявил интерес к данной теме, ее вес будет декрементирован. В данном случае вследствие учета обратной связи рекомендации будут более персонализированными. Однако качество тематической модели имеет сильную зависимость от объема и качества анализируемой информации.

3.5. Резюме

В данном разделе были сформулированы основные требования, предъявляемые к разрабатываемому модулю. Также проведен анализ подходов к построению выбранного типа рекомендательных систем. Поскольку каждый из подходов обладает определенными преимуществами и недостатками, было принято решение реализовать оба подхода и сравнить полученные результаты.

4. ОБЗОР ИСПОЛЬЗУЕМЫХ АЛГОРИТМОВ

В данном разделе рассматриваются два используемых алгоритма преобразования текстовых описаний к числовым последовательностям.

4.1. Алгоритм TF-IDF

TF-IDF — статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса [6]. Вес каждого слова определяется путем умножения двух метрик: сколько раз слово встречается в контексте данного документа и частота слова во всем наборе документов.

- TF - отношение числа вхождения некоторого слова к общему количеству слов документа (4.1).

$$TF = \frac{n_i}{\sum_{i=j}^{k} n_j} \tag{4.1}$$

- IDF — инверсия частоты, с которой некоторое слово встречается в документах коллекции (4.2)..

$$IDF = \log \frac{|D|}{|(d_i \supset t_i)|} \tag{4.2}$$

Умножение этих двух чисел приводит к оценке слова TF-IDF в документе (4.3). Больший вес получат слова с высокой частотой в пределах конкретного документа и с низкой частотой употреблений в других документах.

$$w_i = TF * IDF \tag{4.3}$$

4.2. LDA

Другим популярным способом обработки текстовой информации является Латентное распределение Дирихле (LDA).

LDA – это генеративная, вероятностная, иерархическая Байесовская модель, которая индуцирует темы из коллекции текстовых документов в три шага:

- Каждый документ в коллекции распределяется по темам, которые отбираются для этого документа на основе распределения Дирихле.
- Каждое слово в документе, связано с одной из тем на основе распределения Дирихле.
- Каждая тема представлена в виде полиномиального распределения над словами, которые назначены в выборку темы.

Для построения LDA модели необходимо предварительно преобразовать корпус в терм-документную матрицу.

Терм-документная матрица — это матрица, строки которой соответствуют количеству документов в корпусе, а столбцы — количеству уникальных слов, представленных в данном корпусе. На основе анализа встречаемости слов в контексте документов LDA строит два распределения:

- распределение тем по текстам;
- распределение слов по темам.

Для построения LDA модели нет необходимости в размеченной обучающей выборке, поскольку данный подход относится к форме машинного обучения без учителя. Однако необходимо выполнить задание количества тем. Для определения оптимального количества в данном случае была использована метрика оценки косинусного сходства [5]. При таком подходе тема рассматривается как семантический кластер и выполняется расчет косинусного сходства между векторами распределения слов по темам. Минимум такой меры соответствует оптимальному числу тем.

5. РАЗРАБОТКА СИСТЕМЫ

Разработка модуля рекомендаций онлайн-курсов для систем поддержки образовательного процесса выполняется на языке Python, поскольку данный язык содержит большое число специализированных библиотек для анализа данных.

5.1. Сбор данных

Первым этапом при построении нашей рекомендательной системы является сбор информации об объектах с различных платформ. В качестве платформ курсов были рассмотрены Stepik и Открытое образование. Получение информации с данных платформ было реализовано посредством предоставленного открытого API. Информация передается в json формате.

5.1.1. Платформа Открытое образование

Для получения необходимой информации о курсах с данной платформы использовались два типа запросов: для получения общего перечня курсов и для получения детальной информации о каждом курсе. Пример запроса к конкретному курсу представлен в приложении 1.

Каждый повторный запуск курса передается в данном случае как отдельный объект, однако по сути они представляют собой один и тот же ресурс. В связи с этим перед добавлением нового курса в нашу БД выполняется предварительная проверка для предотвращения дублирования.

Описание курса является объединением следующих тегов:

- “description” – краткое описание курса;
- “results” – полученные знания по итогу завершения курса;
- “competences” – формируемые компетенции;

- “content” – описание программы курса.

Так как в базе вместе содержится информация об актуальных и устаревших курсах, выполняется проверка доступности по URL. Также анализируется значение поля “language”, поскольку в рамках данной системы рассматриваются только курсы на русском языке.

5.1.2. Платформа Stepik

При формировании описания курсов с данной платформы использовались три типа запросов: получение перечня курсов с общей информацией, получение наименований разделов и наименований уроков. Пример получаемых данных представлен в приложении 2.

Данная платформа предоставляет возможность более гибкой настройки параметров запроса, что сокращает количество обращений. Так по умолчанию были отобраны только курсы на русском языке, которые при этом являются бесплатными.

Для составления описания решено было взять следующие ключи:

- “description” – краткое описание курса;
- “results” – полученные знания по итогу завершения курса;
- “competences” – формируемые компетенции;
- “content” – описание программы курса.

В данном случае также выполняется проверка доступности по URL.

5.1.3. Резюме

При первом импорте было обнаружено 7300 курсов. После просмотра собранной базы данных было решено внести несколько дополнительных условий отбора курсов:

1. Игнорировать курсы, длина которых составляет менее 50 символов из-за их неинформативности. После принятия данного утверждения, было отброшено порядка 3 тысяч курсов. Такое большое количество обусловлено тем, что на платформе Stepik размещаются курсы не только учебных организаций, но и частных лиц. В большинстве случаев такие курсы направлены на узкую группу лиц, в связи с чем отсутствует детальное описание.

2. Анализировать наименования курсов и целевую аудиторию на наличие следующих паттернов: 'ЕГЭ', 'лет', 'детей', 'школьников', 'старшеклассников', 'ОГЭ', '*_класс'. Таким образом предпринимается попытка отбросить курсы, предназначенные для обучающихся в школе и детей.

После принятия данных условий количество курсов сократилось до 3096.

5.2. Предварительная обработка текста

Получаемые исходные данные имеют плохое качество, в связи с чем необходимо выполнить их предварительную обработку.

Предварительная обработка данных состоит из следующих этапов [3]:

1. преобразование всех слов в нижний регистр;

2. удаление html тегов;

3. удаление одиночных символов;

4. удаление пунктуационных знаков;

5. удаление стоп-слов;

6. приведение слов в базовую форму;

Удаление тегов, одиночных символов и пунктуационных знаков было реализовано с использованием регулярных выражений.

Для удаления стоп-слов использовался корпус из библиотеки Python nltk. Данный корпус содержит основные союзы, местоимения и предлоги, однако имеет достаточно ограниченный набор слов, всего 151. В связи с чем было принято решение расширить список стоп-слов вручную. В ходе анализа было найдены еще порядка 100 новых слов.

Приведение слов к базовой форме решено было реализовывать с помощью лемматизации. При обработке русскоязычных текстов выделяют анализаторы Mystem и pystem2. Оба морфологических анализатора получили широкое распространение и показывают высокие результаты с точки зрения качества анализа данных [2]. Однако по времени обработки mystem существенно проигрывает. В связи с этим в качестве морфологического анализатора был выбран pystem2.

После первичной очистки данных был выполнен повторный анализ на предмет выявления слов, не несущих смысловой нагрузки в рамках нашей задачи. К примеру, к таким словам были отнесены "курс" , "тест" , "экзамен". Данные слова также были добавлены в список стоп-слов.

После выполнения предварительной обработки был проведен анализ изменения длины описаний курсов 5.1.

Таблица 5.1

Сравнение длины курсов до и после выполнения обработки

	Исходные данные	**После обработки**
Меньше 50 слов	0	413
От 50 до 100 слов	727	925
От 100 до 150 слов	584	600
Более 150 слов	1785	1158
Среднее число слов	**227.0419**	**151.5801**

5.3. Построение рекомендаций

5.3.1. Реализация подхода на основе TF-IDF

Построение TF-IDF оценки было выполнено с помощью готовой библиотеки scikit-learn. При создании модели были заданы дополнительные условия:

1. Не учитывать слова встречающиеся реже, чем в 5 документах. Таким образом из рассмотрения удаляются слишком редкие слова.

2. Не учитывать слова встречающиеся более, чем в 50 процентах от общего числа курсов.

3. Выполнить построение би- и триграмм. N-граммами называется последовательность слов из n элементов. Данное дополнение позволяет учитывать не только отдельные слова, но и определенные устойчивые словосочетания. К примеру, таким образом удается выделить словосочетание 'база данных'

После построения TF-IDF матрицы на основе описаний курсов, выполняется последовательный импорт учебных планов из базы данных вуза. Для каждого учебного плана выполняется аналогичное преобразование описаний рабочих программ в числовую форму, однако уже на основе ранее составленного словаря.

Последним этапом с помощью косинусного сходства определяется близость векторов каждой рабочей программы с исследуемым перечнем курсов. Если близость векторов составила более 0.2, делается предположение, что данный курс можно порекомендовать для данной учебной программы. Пороговое значение было определено путем многократного повторения экспериментов. Полученные рекомендации сохраняются в предварительно созданной таблице БД в представленном ниже формате 5.2.

Таблица 5.2

Формат хранения рекомендаций

studyplan_programblock_id	courses
1	[{"name":"a" ,"link":"https://openedu.ru"}, ...]
2	[{"name":"b" ,"link":"https://stepik.ru"}, ...]

5.3.2. Реализация подхода на основе тематического моделирования

Для выполнения тематического моделирования была использована готовая реализация модели LDA из библиотеки gensim.

В первую очередь требовалось привести данные в правильный формат. Описания курсов были разделены на отдельные слова, а затем с помощью библиотеки Phraser добавлены би- и три-граммы аналогично предыдущему подходу. Из полученных данных был сформирован словарь с применением тех же условий отбрасывания слов: не менее, чем в 5 документах и не чаще, чем в 50 процентах от общего размера корпуса. На основе полученного словаря для каждого курса составляется разреженное представление в виде списка кортежей - (id_токена, частота).

Следующим этапом был проведен анализ оптимального числа тем. Для этого использовался подход, описанный в разделе 4.3. Поскольку невозможно сделать более узкое априорное предположение о необходимом количестве тем, был проведен анализ более широкого диапазона. Тестирование LDA модели было проведено в диапазоне от 10 до 250 тем с шагом 10. Результаты представлены на рис 5.1.

Исходя из полученного графика было принято решение, что оптимальным параметром является 70 тем. Остальные параметры LDA модели были подобраны экспериментальным путем.

Как и в предыдущем подходе, вычисление близости профиля пользователя и объектов реализовано с помощью косинусного сходства.

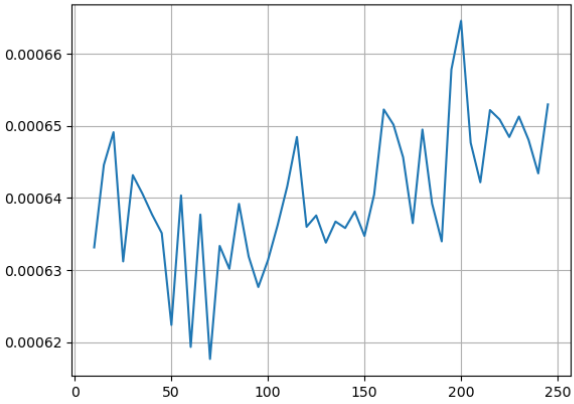


Рис.5.1. Выбор оптимального количества тем

5.4. Резюме

В данном разделе были разобраны реализации двух выбранных подходов построения рекомендаций. Были описаны используемые в работе инструменты и подробно разобраны все этапы построения рекомендаций. Исходный код для некоторых элементов приведен в приложении 2.

6. ТЕСТИРОВАНИЕ

В данном разделе проводится сравнение реализованных подходов на основе A/B тестирования. В качестве метрики использовалась экспертная оценка.

6.1. Тестовая выборка

В качестве тестовой выборки были сгенерированы БД студентов и оценок с привязкой к имеющейся БД образовательных программ. Затем были построены векторы интересов пользователей на основе критерия, высказанного в главе 3.1

6.2. Задача тестирования

Задачей тестирования является сравнение рекомендаций, полученных на основе двух реализованных подходов и выявление лучшего.

6.3. Ход тестирования

Рассмотрим несколько примеров рекомендаций полученных с применением различных подходов. В качестве первого примера возьмем рекомендации для пользователя с следующими интересами.

Таблица 6.1

Вектор интересов пользователя 1

Наименование предметов
Алгоритмизация и программирование
История
Базы данных

Теперь проанализируем рекомендации, полученные с помощью наших подходов. Следует уточнить, что в рамках первого подхода соотношения

рекомендаций, выдаваемых для каждого предмета задается вручную. В рамках второго подхода выдаются лучшие N курсов для общего корпуса.

Таблица 6.2

Полученные рекомендации для пользователя 1

Подход 1	**Подход 2**
История	Интерактивный тренажер по SQL
История России	Великая Отечественная война
Всеобщая история. Часть 2	Программирование на языке C++
Введение в базы данных	Введение в программирование
Введение в базы данных	История Древней Греции. Краткий курс.
Базы данных	Отечественная война 1812 года
ИТ-класс Python	Вторая мировая война
Программирование Python	C++
Чемпионат по скоростному кодированию - пробный тур	Инди-курс программирования на Python от egoroff_channel

Исходя из полученных данных можно сделать вывод, что в данном случае обе рекомендательные системы показали хороший результат. Стоит отметить, что второй подход при этом показал большую вариативность.

Рассмотрим еще один пример.

Таблица 6.3

Вектор интересов пользователя 2

Наименование предметов
Проектирование экологической деятельности
Алгоритмы решения нестандартных задач
Техническое регулирование

Таблица 6.4

Полученные рекомендации для пользователя 2

Подход 1	**Подход 2**
Основы метрологии, стандартизация и оценка соответствия	Электрические машины
Документирование и сертификация	Линейные электрические цепи
Государственная культурная политика	Основы цифровой обработки сигналов
Теория решения изобретательских задач (ТРИЗ)	Методы обработки навигационной измерительной информации
Теория решения изобретательских задач	Основы взаимозаменяемости
Управление интеллектуальной собственностью	Методы и алгоритмы теории графов
Экология	Реология
Экология	Биохимия растений
Глобальные экологические проблемы	Биохимия сельскохозяйственной продукции

Исходя из табл. 6.4, в этом рассматриваемом примере более точный результат показал подход 1. Разница рекомендаций между подходами объясняется тем, что в данном случае второй подход соотнес образовательную программу „Техническое регулирование“к техническим предметам. Данное предположение не является абсолютно неверным, поскольку в описании образовательной действительно можно выделить две лидирующие темы: „документация“и „техника“. Однако в реальности более значимой является тема „документации“.

6.4. Анализ результатов

По итогам исследования точнее оказались рекомендации, подобранные на основе сравнения непосредственно описаний объектов. Проигрыш тематического моделирования в данном случае можно объяснить следующими факторами:

- изначальная неравномерность распределения курсов по темам. Большая часть анализируемых курсов, была связанна с областью программирования и информационных технологий. В связи с чем такие темы были выделены хорошо, однако моделирование более узких областей оказалось не точным;

- недостаточность описаний. Как было подчеркнуто ранее, после очистки среднее описание курсов составляет всего порядка 150 слов.

6.5. Резюме

В данном разделе был выполнен анализ работоспособности реализованных подходов. Было выявлено, что на данном этапе лучший результат удается получить при сопоставлении описаний объектов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были исследованы различные существующие методы построения рекомендательных систем. По сформулированным требованиям и задачам было принято решение реализовывать модуль на базе контентной фильтрации. При разработке модуля были реализованы два различных подхода: построение рекомендаций на основе прямого сопоставления описаний и предварительное формирование профилей пользователей и объектов.

В главе 2 был выполнен анализ готовых решений в области контентных рекомендательных систем. Глава 3 была посвящена формированию требований, определению основных функций и анализу накладываемых ограничений и возможных путей их обхода.

В главе 4 было представлено краткое описание алгоритмов, используемых при реализации выбранных подходов.

По итогам тестирования более точные результаты были получены с помощью первого подхода. Однако он не учитывает взаимодействия пользователей, что существенно ограничивает гибкость системы.

В связи с этим дальнейшее развитие работы заключается в повышении точности и эффективности разработанного модуля.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Anand Sarabjot Singh, Mobasher Bamshad. Intelligent Techniques for Web Personalization // Proceedings of the 2003 International Conference on Intelligent Techniques for Web Personalization. — ITWP'03. — Berlin, Heidelberg : Springer-Verlag, 2003. — P. 1–36. — URL: https://doi.org/10.1007/11577935_1.
2. Asiryan Aleksandr Kamoevich. Сравнение инструментов морфологической разметки // Научный взгляд в будущее. — 2017. — № 07-01. — С. 27–34.
3. Text Cleaning Methods for Natural Language Processing [Электронный ресурс], Towards Data Science. — URL: https://towardsdatascience.com/text-cleaning-methods-for-natural-language-processing-f2fc1796e8c7 (дата обращения: 13.05.2020).
4. Who's Benefiting from MOOCs, and Why [Электронный ресурс], Harvard Business Review. — URL: https://hbr.org/2015/09/whos-benefiting-from-moocs-and-why (дата обращения: 15.04.2020).
5. A density-based method for adaptive LDA model selection / Juan Cao, Tian Xia, Jintao Li et al. // Neurocomputing. — 2009. — Vol. 72, no. 7. — P. 1775 – 1781. — Advances in Machine Learning and Computational Intelligence. URL: http://www.sciencedirect.com/science/article/pii/S092523120800372X.
6. Шахзад Кайзер Рамша Али. Text Mining: использование TF-IDF для изучения релевантности слов для документов // Международный журнал компьютерных приложений. — 2018. — № 07-01. — С. 25–29.

ПРИЛОЖЕНИЕ 1
ПРИМЕР ВЫГРУЖАЕМЫХ ДАННЫХ

Листинг 1.1. Выгружаемые данные с платформы Открытое образование


```
{
    "hours": null,
    "competences": null,
    "external_url": "https://openedu.ru/course/mipt/ADV_GRAPHS/",
    "title": "Продвинутые графы",
    "transfers": [],
    "finished_at": "2020-06-30",
    "started_at": "2020-02-17",
    "subtitles_lang": "",
    "cert": true,
    "duration": {
        "value": 8,
        "code": "week"
    },
    "teachers": [
        {
            "display_name": "Дайняк Александр Борисович",
            "image": "https://cdn.openedu.ru/f1367c/instructor/48
                a056140832c2ae0c0c5288b1d9fc056c965f2d.jpg",
            "description": "Преподаватель"
        }
    ],
    "estimation_tools": "",
    "promo_url": "http://s18874.cdn.ngenix.net/video/x/i/xip02v1gjn/hd.
        mp4",
    "visitors": 4205,
    "results": null,
    "hours_per_week": null,
    "proctoring_service": "",
    "content": null,
    "lectures": 8,
    "enrollment_finished_at": "2020-06-30",
    "accreditated": "",
    "direction": [
        "01.00.00",
        "03.00.00",
        "09.00.00",
        "09.03.01",
        "10.00.00",
        "16.00.00"
```

```
    ],
    "description": "<p>Ключевые вопросы, затрагиваемые в курсе:</p>\n\n<
        ul>\n\t<li>фундаментальные понятия, законы, теории случайных гра
        фов;</li>\n\t<li>современные проблемы соответствующих разделов с
        лучайных графов;</li>\n\t<li>понятия, аксиомы, методы доказатель
        ств и доказательства основных теорем в разделах, входящих в базо
        вую часть цикл;</li>\n\t<li>основные свойства соответствующих ма
        тематических объектов;</li>\n\t<li>аналитические и численные под
        ходы и методы для решения типовых прикладных задач случайных гра
        фов</li>\n</ul>",
    "requirements": "",
    "id": "74789a6b19064aaba886de625feb2a3e",
    "language": "ru",
    "sessionid": "course-v1:mipt+ADV_GRAPHS+spring_2020",
    "institution": "84dcc16be487434c9539565e939971f0",
    "business_version": 3,
    "promo_lang": "ru",
    "image": "https://cdn.openedu.ru/f1367c/cover/
        e313b933f14baf02c16ca198f278f29cf352509a.png"
}
```

Листинг 1.2. Выгружаемые данные с платформы Stepik

```
{
  "meta": {
    "page": 1,
    "has_next": true,
    "has_previous": false
  },
  "courses": [
    {
      "id": 73942,
      "summary": "Выполнение домашних заданий Examis",
      "workload": "",
      "cover": "/media/cache/images/courses/57527/cover_mltDFRV/
          ccdd206a28931486f4890e3162de53ec.jpg",
      "intro": "",
      "course_format": "",
      "target_audience": "",
      "certificate_footer": null,
      "certificate_cover_org": null,
      "is_certificate_issued": false,
      "is_certificate_auto_issued": false,
      "certificate_regular_threshold": null,
      "certificate_distinction_threshold": null,
      "instructors": [
```

```
        73829122
      ],
      "certificate": "",
      "requirements": "",
      "description": "",
      "sections": [
        138222,
        138223,
        138224,
        138225,
        138226,
        138227,
        138228,
        138229,
        138230,
        138231,
        138232,
        138233
      ],
      "total_units": 21,
      ...
      "is_certificate_with_score": true,
      "owner": 15017732,
      "language": "ru",
      "is_featured": false,
      "is_public": true,
      "title": "Copy of Русский язык | Тест + сочинение",
      "slug": "Copy-of-Русский-язык-Тест-+-сочинение-73942",
      "begin_date": null,
    ...
    }
  ],
  "enrollments": []
}
```

ПРИЛОЖЕНИЕ 2
ЛИСТИНГИ

Листинг 2.1. Импорт данных с платформы Открытое образование

```
import requests
import json
import numpy as np
import re
from langdetect import detect

def import_courses(coursesList):
    next_page = "https://courses.openedu.ru/api/courses/v1/courses/"
    course_url = "https://openedu.ru/api/courses/export/"
    index = 0

    while next_page is not None:

        courses_json = requests.get(next_page).json()
        next_page = courses_json['pagination']['next']
        courses_json = courses_json['results']

        # add courses from one page
        for i in range(0, len(courses_json)):

            # remove duplicates of different years
            if np.all(coursesList['parameters'] != (courses_json[i]['org'
                ] + courses_json[i]['number'])):
                try:
                    name = courses_json[i]['name'].lower()
                    name_split = re.search(r'\b\d-\d\b|\d+.*класс\w*|\b\d
                        {1,2}\s\w\b', name)
                    stop_words = ['ЕГЭ', 'егэ', 'лет', 'детей', 'школьник
                        ов', 'старшеклассников', 'ОГЭ', 'огэ']
                    mask = [courses_json[i]['name'].find(j) != -1 for j
                        in stop_words]
                    if name_split is not None or np.any(mask):
                        continue
                    course = requests.get((course_url + courses_json[i]['
                        org'] + '/'
                                           + courses_json[i]['number'] +
                                               '?format=json')).json()
                    if 'external_url' not in course:
                        continue
                    accessibility = requests.head(course['external_url'])
```

```
35                      if accessibility.status_code == 404:
36                          continue
37                      if course['language'] != 'ru':
38                          continue
39
40                      description = ''
41
42                      if course['description'] is not None:
43                          description += course['description']
44                      if course['results'] is not None:
45                          description += (course['results'])
46                      if course['competences'] is not None:
47                          description += course['competences']
48                      if course['content'] is not None:
49                          description += course['content']
50
51                      len_words = description.split()
52
53                      if len(len_words) < 50:
54                          continue
55
56                      language = detect(description)
57
58                      if language != 'ru':
59                          continue
60
61                      coursesList.loc[index, 'link'] = course['external_url
                            ']
62                      coursesList.loc[index, 'parameters'] = courses_json[i
                            ]['org'] + courses_json[i]['number']
63                      coursesList.loc[index, 'name'] = courses_json[i]['
                            name']
64                      coursesList.loc[index, 'description'] = description
65                      coursesList.loc[index, 'year'] = courses_json[i]['
                            start']
66
67                      index += 1
68                  except json.decoder.JSONDecodeError:
69                      continue
70              else:
71                  j = coursesList.parameters[
72                      coursesList.parameters == courses_json[i]['org'] +
                            courses_json[i]['number']].index.tolist()[0]
73                  if coursesList.loc[j, 'year'] < courses_json[i]['start']:
74                      coursesList.loc[j, 'year'] = courses_json[i]['start']
75                      coursesList.loc[j, 'name'] = courses_json[i]['name']
```

```
76      return coursesList
```

Листинг 2.2. Импорт данных с платформы Stepik

```
1  import requests
2  import json
3  import re
4  import numpy as np
5  from langdetect import detect
6
7
8  def import_courses(coursesList):
9      index = 0
10     if coursesList.last_valid_index() is not None:
11         index = 1 + coursesList.last_valid_index()
12     has_next_page = True
13     page = 1
14
15     while has_next_page:
16
17         courses_json = requests.get("https://stepic.org/api/courses",
               params={'page': page, 'language': 'ru', 'is_paid': False}).
               json()
18         has_next_page = courses_json['meta']['has_next']
19         page += 1
20         courses_json = courses_json['courses']
21
22         for i in range(0, len(courses_json)):
23             try:
24                 link = 'https://stepic.org/course/' + str(courses_json[i
                       ]['id'])
25                 accessibility = requests.head(link)
26                 if accessibility.status_code == 404:
27                     continue
28                 name = courses_json[i]['title'].lower()
29                 name_split = re.search(r'\b\d-\d\b|\d+.*класс\w*|\b\d
                       {1,2}\s\w\b', name)
30                 stop_words = ['ЕГЭ', 'егэ', 'лет', 'детей', 'школьников',
                        'старшеклассников', 'ОГЭ', 'огэ']
31                 mask = [courses_json[i]['title'].find(j) != -1 for j in
                       stop_words]
32                 if name_split is not None or np.any(mask):
33                     continue
34
35                 description = ''
36
```

```
37                  if courses_json[i]['summary'] is not None:
38                      description += courses_json[i]['summary']
39                  if courses_json[i]['requirements'] is not None:
40                      description += courses_json[i]['requirements']
41                  if courses_json[i]['target_audience'] is not None:
42                      description += courses_json[i]['target_audience']
43                  if courses_json[i]['description'] is not None:
44                      description += courses_json[i]['description']
45                  sections_url = 'https://stepik.org/api/sections?{}'.
                        format(
46                      '&'.join('ids[]={}'.format(obj_id) for obj_id in
                            courses_json[i]['sections']))
47                  sections_json = requests.get(sections_url).json()
48                  sections_json = sections_json['sections']
49                  for j in range(0, len(sections_json)):
50                      description += sections_json[j]['title'] + '. '
51                  if courses_json[i]['total_units'] != 0:
52                      lessons = requests.get('https://stepic.org:443/api/
                            lessons',
53                                             params={'language': 'ru', '
                                                   course': courses_json[i][
                                                   'id']}).json()
54                      lessons = lessons['lessons']
55                      for k in range(0, len(lessons)):
56                          description += lessons[k]['title'] + '. '
57
58                  len_words = description.split()
59
60                  if len(len_words) < 50:
61                      continue
62
63                  language = detect(description)
64
65                  if language != 'ru':
66                      continue
67
68                  target_aud = courses_json[i]['target_audience'].lower()
69                  target_aud_split = re.search(r'\d+.*класс\w*|\b\d{1,2}\s\
                        w\b', target_aud)
70                  mask = [courses_json[i]['target_audience'].find(j) != -1
                        for j in stop_words]
71                  if target_aud_split is not None or np.any(mask):
72                      continue
73
74                  coursesList.loc[index, 'name'] = courses_json[i]['title']
75                  coursesList.loc[index, 'parameters'] = str(courses_json[i
```

```
                    ]['id'])
76                  coursesList.loc[index, 'link'] = link
77                  coursesList.loc[index, 'description'] = description
78
79                  index += 1
80              except json.decoder.JSONDecodeError:
81                  continue
82
83      return coursesList
```

Листинг 2.3. Предварительная обработка текста

```
1   import re
2   import Analysis
3   from nltk.corpus import stopwords as nltk_stopwords
4   import pymorphy2
5
6
7   first_block_stop_words = ['который', 'твой', 'сих', 'свой', 'твоя', 'этим
        и', 'слишком', 'нами', 'всему', 'будь', 'саму',
8                             'чаще', 'ваше', 'сами', 'наш', 'затем', 'еще',
                                  'самих', 'наши', 'тут', 'каждое', 'мочь',
9                             'весь', 'этим', 'наша', 'своих', 'оба', 'зато',
                                   'те', 'этих', 'вся', 'ваш', 'такая', 'тем
                                  и',
10                            'ею', 'которая', 'нередко', 'каждая', 'также',
                                  'чему', 'собой', 'самими', 'нем', 'вами',
                                  'ими',
11                            'откуда', 'такие', 'тому', 'та', 'очень', 'сама
                                  ', 'нему', 'оно', 'этому', 'кому', 'тобой'
                                  ,
12                            'таки', 'твоё', 'каждые', 'твои', 'мой', 'нею',
                                   'самим', 'ваши', 'ваша', 'кем', 'мои', 'о
                                  днако',
13                            'сразу', 'свое', 'ними', 'всё', 'неё', 'тех', '
                                  хотя', 'всем', 'тобою', 'тебе', 'одной', '
                                  другие', 'эта',
14                            'само', 'эта', 'буду', 'самой', 'моё', 'своей',
                                   'такое', 'всею', 'будут', 'своего', 'кого
                                  ',
15                            'свои', 'мог', 'нам', 'особенно', 'её', 'самому
                                  ', 'наше', 'кроме', 'вообще', 'вон', 'мною
                                  ',
16                            'никто', 'это', 'либо', 'какой', 'именно', 'нек
                                  оторый', '   ', '-']
17
```

```
18 secondary_block_stop_words = ['понятие', 'задача', 'работа', 'модуль', 'з
       адание', 'тема', 'знание', 'основной', 'лекция',
19                              'решение', 'тест', 'урок', 'вариант', 'друг
                                    ой', 'stepik', 'каждый', 'поэтому', 'д
                                    анный', 'курс',
20                              'course', 'lesson', 'общий', 'org', 'иметь'
                                    , 'например', 'какой', 'часть', 'тест'
                                    , 'test',
21                              'короче', 'вопрос', 'пример', 'примерный',
                                    'билет', 'зачет', 'экзамен', 'зачёт',
                                    'экзаменационный', 'контрольный', '
                                    nbsp']
22
23
24 def word_processing(list_descriptions):
25     stop_words = nltk_stopwords.words('russian') + nltk_stopwords.words('
           english') + first_block_stop_words + secondary_block_stop_words
26
27     morph = pymorphy2.MorphAnalyzer()
28
29     for i in range(0, (list_descriptions.last_valid_index() + 1)):
30
31         # convert string to lowercase
32         list_descriptions.loc[i, 'description'] = list_descriptions.loc[i
               , 'description'].lower()
33
34         # delete html tags
35         list_descriptions.loc[i, 'description'] = re.sub(r'\<[^>]*\>', '
               ', list_descriptions.loc[i, 'description'])
36         list_descriptions.loc[i, 'description'] = re.sub(r'&[a-zA-Z]*', '
                ', list_descriptions.loc[i, 'description'])
37
38         # delete punctuation marks
39         list_descriptions.loc[i, 'description'] = re.sub(r'[\r\t\n\.,
               \/#!\$%\^&\*;:"{}   =_‘~()\?\+]', ' ', list_descriptions.loc
               [i, 'description'])
40
41         # delete number and tags
42         list_descriptions.loc[i, 'description'] = re.sub(r'(?:\s\d+\b|&\w
               +)', ' ', list_descriptions.loc[i, 'description'])
43
44         # delete single characters
45         list_descriptions.loc[i, 'description'] = re.sub(r'\b[a-яA-я]\b',
                ' ', list_descriptions.loc[i, 'description'])
46
47         # lemmetize words
```

```
48          lemmas = (morph.parse(j)[0].normal_form for j in
                list_descriptions.loc[i, 'description'].split())
49
50          # delete stop words
51          list_descriptions.loc[i, 'description'] = ' '.join(j for j in
                lemmas if j not in stop_words)
52
53
54      Analysis.analysis_num_word(list_descriptions, "После первичной чистки
            (удаление стоп-слов и лемматизация)")
55
56      return list_descriptions
```

Листинг 2.4. Реализация подхода на основе TF-IDF

```
1   from sklearn.feature_extraction.text import TfidfVectorizer
2   from sklearn.metrics.pairwise import linear_kernel
3   import pandas as pd
4   import psycopg2
5   import numpy as np
6   import WordProcessing
7
8
9   connection = psycopg2.connect(host='127.0.0.1', user='postgres', password
        ='s2128506', dbname='Test')
10  cursor = connection.cursor()
11
12
13  def main(coursesList):
14      create_tables()
15
16      tf = TfidfVectorizer(analyzer='word', min_df=5, max_df=0.5,
            ngram_range=(1, 3))
17      tfidf_courses = tf.fit_transform(coursesList['description'].values.
            astype('U'))
18
19      cursor.execute("select id from auth_group")
20      id_groups = cursor.fetchall()
21      for i in id_groups:
22          cursor.execute("select distinct studyplan_id from students where
                group_id = %(id)s ", {'id': i[0]})
23          studyplan_id = cursor.fetchall()
24          for j in studyplan_id:
25              subject_list = pd.DataFrame(columns=['id_subject', '
                    description'])
26              subject_list = WordProcessing.word_processing(
```

```
                get_work_program(j[0], subject_list))
            tfidf_priorities = tf.transform(subject_list['description'])
            dense = tfidf_priorities.todense()
            denselist = dense.tolist()
            for l in range(0, len(denselist)):
                cosine_similarities = linear_kernel(dense[l],
                    tfidf_courses).flatten()
                top_courses = np.where(cosine_similarities >= 0.2)[0]
                cursor.execute("INSERT INTO
                    studyplan_programblock_recommend (
                    studyplan_programblock_id, courses) VALUES (%(id)s,
                    %(courses)s)", {'id': subject_list.loc[l, '
                    id_subject'], 'courses': (coursesList.loc[
                    top_courses, 'name':'link']).to_json(orient='records
                    ', force_ascii=False)})
                print(subject_list.loc[l, 'id_subject'])
                print(coursesList.loc[top_courses, 'name':'link'])
            connection.commit()


def create_tables():
    cursor.execute("""DROP TABLE IF EXISTS
        studyplan_programblock_recommend""")
    cursor.execute("""
    CREATE TABLE studyplan_programblock_recommend(
        studyplan_programblock_id INT NOT NULL,
        courses JSON
    );
    ALTER TABLE studyplan_programblock_recommend
        ADD FOREIGN KEY (studyplan_programblock_id) REFERENCES
            studyplan_programblock(id)
    ;""")
    connection.commit()


def get_work_program(id_studyplan, subject_list):
    index = -1
    cursor.execute("""
        with study_plan as(
            select id,name,rpd_id
            from studyplan_programblock
            where study_plan_id = %(id)s and rpd_id is not null
        )
        select distinct study_plan.id, content, number
        from rpd_workprogramsection
        inner join study_plan
```

```
        on rpd_workprogramsection.work_program_id = study_plan.rpd_id
        where number = '1.2' or number = '11.2' or title = 'Содержание и
            результаты обучения'""", {'id': id_studyplan})
    for i in cursor.fetchall():
        if np.all(subject_list['id_subject'] != i[0]):
            if index == -1 or subject_list.loc[index, 'description'] != '
                ':
                index += 1
            subject_list.loc[index, 'id_subject'] = i[0]
            subject_list.loc[index, 'description'] = ''
        if i[2] == '11.2':
            if 'content' in i[1]:
                subject_list.loc[index, 'description'] += i[1]['content']
        else:
            if 'sections' in i[1]:
                for k in i[1]['sections']:
                    subject_list.loc[index, 'description'] += k['name'] +
                        '. '

    return subject_list
```

Листинг 2.5. Реализация подхода на основе LDA модели

```
from gensim.models import Phrases
from gensim.models.phrases import Phraser
from gensim.corpora.dictionary import Dictionary
from gensim.models.ldamulticore import LdaMulticore as LDA
from gensim.models import CoherenceModel
from sklearn.metrics.pairwise import linear_kernel
import psycopg2
import pyLDAvis.gensim
import warnings
import WordProcessing
import pandas as pd
import config
import numpy as np

connection = psycopg2.connect(host='127.0.0.1', user='postgres', password
    ='s2128506', dbname='Test')
cursor = connection.cursor()


def main(coursesList):

    lda, dictionary, bigrams, trigrams = create_LDA_model(coursesList)
    courses_topic = config.matrix_courses_topic.to_numpy()
```

```
23
24      cursor.execute("select id from auth_group")
25      id_groups = cursor.fetchall()
26      for i in id_groups:
27          cursor.execute("select distinct studyplan_id from students where
                group_id = %(id)s ", {'id': i[0]})
28          studyplan_id = cursor.fetchall()
29          for j in studyplan_id:
30              subject_list = pd.DataFrame(columns=['id_subject', '
                    description'])
31              subject_list = WordProcessing.word_processing(
                    get_work_program(j[0], subject_list))
32              cursor.execute("select id from students where group_id = %(
                    group_id)s and studyplan_id = %(studyplan_id)s",
33                             {'group_id': i[0], 'studyplan_id': j[0]})
34              student_id = cursor.fetchall()
35              for k in student_id:
36                  cursor.execute("select subject from recommend_stud where
                        student_id = %(id)s ", {'id': k[0]})
37                  subject_id = cursor.fetchone()[0]
38                  stud_priorities = pd.DataFrame(columns=['id_subject', '
                        description'])
39                  for l in range(0, len(subject_id)):
40                      stud_priorities.loc[l, :] = subject_list.loc[
41                          subject_list['id_subject'] == subject_id[l]['id'
                                ]].values
42                  token_stud_prog = [program.split(' ') for program in
                        stud_priorities['description']]
43                  token_stud_prog = add_n_grams(token_stud_prog, bigrams,
                        trigrams)
44                  prog_corp = [dictionary.doc2bow(program) for program in
                        token_stud_prog]
45                  topic_prog = lda.get_document_topics(prog_corp)
46                  profile_student = np.zeros(config.num_lda_topic)
47                  for l in range(0, len(topic_prog)):
48                      for m in topic_prog[l]:
49                          profile_student[m[0]] += m[1]
50                  profile_student = profile_student / (len(topic_prog))
51                  mask = np.argsort(linear_kernel(profile_student.reshape
                        (1, -1), courses_topic).flatten())[::-1][:20]
52                  print(coursesList.loc[mask, 'name':'link'])
53
54  def create_LDA_model(coursesList):
55      warnings.filterwarnings('ignore')
56      text_clean = [doc.split(' ') for doc in coursesList['description']]
57      bigrams, trigrams = create_n_grams(text_clean)
```

```
    text_clean = add_n_grams(text_clean, bigrams, trigrams)

    id2word = Dictionary(text_clean)
    id2word.filter_extremes(no_below=5, no_above=0.45)
    corpus = [id2word.doc2bow(text) for text in text_clean]

    num_topics = config.num_lda_topic
    lda_model = LDA(corpus=corpus,
                    id2word=id2word,
                    num_topics=num_topics,
                    random_state=42,
                    alpha='asymmetric',
                    passes=25)
    coherence_model_c_v = CoherenceModel(model=lda_model, texts=
        text_clean, dictionary=id2word, coherence='c_v')
    c_v = coherence_model_c_v.get_coherence()
    term_topic_mat = lda_model.get_topics()
    aver_cosine_similarities = 0
    for i in range(0, (num_topics - 1)):
        cosine_similarities = linear_kernel(term_topic_mat[i].reshape(1,
            -1), term_topic_mat[i + 1:]).flatten()
        aver_cosine_similarities += sum(cosine_similarities)
    if num_topics != 1:
        aver_cosine_similarities = aver_cosine_similarities / (num_topics
            * (num_topics - 1) / 2)
    print(c_v)
    print(aver_cosine_similarities)

    create_vector_topics(lda_model, corpus, id2word, coursesList)

    visual_data = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
    pyLDAvis.save_html(visual_data, 'topics.html')
    return lda_model, id2word, bigrams, trigrams


def create_vector_topics(model, corpus, dictionary, coursesList):
    probability_doc_topics = model.get_document_topics(corpus)
    for i in range(0, len(corpus)):
        best_topic = 0
        prob_best_topic = 0
        for j in probability_doc_topics[i]:
            config.matrix_courses_topic.loc[i, j[0]] = j[1]
            if j[1] > prob_best_topic:
                best_topic = j[0]
                prob_best_topic = j[1]
        coursesList.loc[i, 'topic_id'] = best_topic
```

```
101     config.matrix_courses_topic = config.matrix_courses_topic.fillna(0)
102     config.matrix_courses_topic.to_csv("./source/matrix_doc_top.csv",
            index=False)
103
104
105 def create_n_grams(text):
106     bigram = Phrases(text, min_count=20, threshold=10, delimiter=b' ')
107     bigram_phraser = Phraser(bigram)
108     tokens_bigram = bigram_phraser[text]
109     trigram = Phrases(tokens_bigram, min_count=10, threshold=10,
            delimiter=b' ')
110     trigram_phraser = Phraser(trigram)
111     tokens_trigram = trigram_phraser[tokens_bigram]
112     return tokens_bigram, tokens_trigram
113
114
115 def add_n_grams(text, bigrams, trigrams):
116     for id in range(len(text)):
117         for token in bigrams[id]:
118             if token.count(' ') == 1:
119                 text[id].append(token)
120         for token in trigrams[id]:
121             if token.count(' ') == 2:
122                 text[id].append(token)
123     return text
124
125
126 def get_work_program(id_studyplan, subject_list):
127     index = -1
128     cursor.execute("""
129             with study_plan as(
130                 select id,name,rpd_id
131                 from studyplan_programblock
132                 where study_plan_id = %(id)s and rpd_id is not null
133             )
134             select distinct study_plan.id, content, number
135             from rpd_workprogramsection
136             inner join study_plan
137             on rpd_workprogramsection.work_program_id = study_plan.rpd_id
138             where number = '1.2' or number = '11.2' or title = 'Содержани
                    е и результаты обучения'""",
139                    {'id': id_studyplan})
140     for i in cursor.fetchall():
141         if np.all(subject_list['id_subject'] != i[0]):
142             if index == -1 or subject_list.loc[index, 'description'] != '
                    ':
```

```
143                 index += 1
144             subject_list.loc[index, 'id_subject'] = i[0]
145             subject_list.loc[index, 'description'] = ''
146         if i[2] == '11.2':
147             if 'content' in i[1]:
148                 subject_list.loc[index, 'description'] += i[1]['content']
149         else:
150             if 'sections' in i[1]:
151                 for k in i[1]['sections']:
152                     subject_list.loc[index, 'description'] += k['name'] +
                            '. '
153
154     return subject_list
```