МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

РЕКОМЕНДОВАНО К ЗАЩИТЕ В ГЭК
Заведующий кафедрой, к.т.н, доцент
_____М. С. Воробьева
02.07. 2021 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
магистерская диссертация

СОЗДАНИЕ СЕРВИСА ДЛЯ ПОИСКА ЭЛЕКТИВНЫХ КУРСОВ И АНАЛИЗА ИХ СОДЕРЖАНИЯ С ПОМОЩЬЮ МЕТОДОВ ИССЛЕДОВАНИЯ ТЕКСТОВ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

02.04.03 Математическое обеспечение и администрирование информационных систем
Магистерская программа «Разработка технологий Интернета вещей и больших данных»

Выполнили работу
(групповой проект)
студенты 2 курса
очной формы обучения

Ханикерян Артем Робертович
Попова Ольга Александровна

Научный руководитель
Заведующий каф. ПО,
к.т.н., доцент

Воробьева Марина Сергеевна

Рецензент
Старший аналитик
ООО «ТС-ИННОТЕХ»

Иваненко Ольга Александровна

Тюмень
2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Текст является основной формой хранения и обмена информации в обществе. Одновременно с этим, текстовая информация по-прежнему занимает значительную долю ресурсов в информационных справочных системах. Поэтому созданию, модернизации и совершенствованию технологий обработки текста уделяют большое внимание на всех этапах развития информационных систем. Наряду с системами хранения и обработки знаний, самыми распространенными системами этой категории являются системы текстового поиска, задача которых заключается в поиске по заданной коллекции документов на естественном языке, удовлетворяющих запросу пользователей [4].

По сравнению с искусственными языками, естественный язык обладает ключевыми отличиями, которые усложняют его обработку в информационных системах. Ключевой особенностью естественного языка является его высокий уровень избыточности, который позволяет облегчить понимание смыслов между участниками коммуникации в условиях неограниченного времени. Негативный эффект избыточности естественного языка состоит в больших объемах передаваемого сообщения, которые складываются из множественного дублирования передаваемой информации, и традиционно используемых лексических связок, и конструкций [21].

Информационный поиск является актуальной задачей анализа текстов на естественном языке. Задача информационного поиска определяется как процесс поиска неструктурированной документальной информации, удовлетворяющей информационные потребности. Изучение данной проблемы важно, поскольку в настоящее время происходит модернизация образовательных программ университетов, имплементация концепции развития человека путем обучения на протяжении всей его жизни, а также появление большого количества качественного, актуального, открытого образовательного контента в сети

Интернет. Всё это приводит к появлению огромного числа образовательных программ [1].

Тюменский Государственный Университет в образовательном процессе использует новую образовательную модель индивидуальных образовательных траекторий для студентов, позволяющую, в дополнении к основным дисциплинам, выбрать дополнительные дисциплины для изучения, тем самым позволяя сформировать уникальный набор компетенций для будущего специалиста.

Индивидуальная образовательная траектория формируется студентом за счет выбора элективных курсов в общеобразовательном и профессиональном блоке дисциплин, а также возможности освоения дополнительного профиля [8].

В ТюмГУ разработано множество элективных курсов по нескольким областям знаний: естественные науки, искусство, математика и информатика, науки об обществе и человеке, социальные коммуникации. Благодаря широкому выбору дисциплин для обучения и обеспечивается возможность формирования студентами индивидуального учебного плана.

Оценив их содержание и свои интересы, студенты формируют свое индивидуальное расписание и получают возможность более глубоко погрузиться в освоение дисциплин профессиональной сферы или получить дополнительные компетенции в других областях [8].

В условиях индивидуальных образовательных траекторий по уровню подготовки студента и требований рынка труда формируется список рекомендаций для разработки элективных курсов [3].

В электронной системе Modeus находится большое количество образовательных курсов с похожим содержанием в области конкретной дисциплины, что ставит проблему выбора наиболее подходящего электива по запросу пользователя. С другой стороны, для преподавателя возникает проблема в создании электива, который освещает тему с новой стороны, так как ему становится затруднительно ориентироваться в многообразии похожих друг на друга элективов.

Целью данной выпускной квалификационной работы является разработка сервиса для поиска элективных курсов и анализа их содержания с помощью методов исследования текстов на естественном языке.

Для достижения цели были выделены следующие задачи:

1. Изучить научные публикации, рассматривающие поиск текстовой информации.
2. Рассмотреть возможности инструментов и методов, осуществляющих поиск и кластеризацию текстов.
3. Произвести выгрузку данных с сайта Modeus и перенос в БД PostgreSQL и Apache HBase.
4. Произвести выгрузку данных из БД, предобработать данные (очистить и нормализовать).
5. Провести анализ методов и технологий, применимых к выгруженным данным.
6. Подобрать методы кластеризации и классификации по выделенным признакам. Выполнить кластеризацию полученного набора данных.
7. Спроектировать сервис для поиска элективных курсов и анализа их содержания с помощью методов исследования текстов на естественном языке.

Для подготовки и защиты выпускной квалификационной работы использовались поиск, анализ информации, системный подход для решения поставленных задач; приемы критического анализа проблемных ситуаций, а также средства и методы саморазвития и самореализации; методики межкультурного взаимодействия; умение расставлять приоритеты собственной деятельности при работе в общем проекте в соответствии с командной стратегией для достижения поставленной цели.

Формулирование выводов по итогам проведенной работы осуществлялись с учетом применения современных коммуникативных технологий (в том числе на иностранном языке) для представления результатов на академических, профессиональных, экспертных ИТ-мероприятиях.

ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ ИССЛЕДОВАНИЙ В ОБЛАСТИ ВЕКТОРИЗАЦИИ И КЛАСТЕРИЗАЦИИ ТЕКСТОВ

1.1. ОБЗОР МЕТОДОВ ВЕКТОРИЗАЦИИ ДОКУМЕНТОВ

Наиболее распространенным представлением текстового документа на естественном языке с применением алгоритмов машинного обучения является числовой вектор. Рассмотрим несколько методов для перевода текстов в числовые тензоры.

На основе набора документов строится словарь из всех встречающихся в нем n-грамм, где n меньше или равно какому-то заранее заданному значению. Документ представляется набором признаков, каждому из которых соответствует одна n-грамма из словаря. Существует множество методов кодирования наборов признаков документа. В данной работе будут рассмотрены следующие методы:

- BinaryBOW
- Bag of words
- TF-IDF

1.1.1.МЕТОД BINARYBOW

Наиболее простой способ при бинарном представлении признака, при котором признак принимает единичное значение, если в документе встречается соответствующая n-грамма, и ноль, если нет. Это простой вариант преобразования токенов в тензоры. Основным недостатком прямого кодирования является размер итогового представления документа: длина получаемого тензора равна длине словаря признаков, что приводит к разреженным данным, работа с которыми требует больших затрат памяти и вычислительных мощностей [6].

Рассмотрим пример использования BinaryBOW.

Возьмем для примера запрос пользователя при выборе учебного курса: «Открытый онлайн курс по дисциплине веб-дизайна. Студенческий семинар по основам программирования». В качестве исходных данных используется

список слов из 9 элементов, на выходе получается квадратная матрица размерностью 9x9 с одним ненулевым элементом в каждой строке (Листинг 1).

```
import sklearn.preprocessing as preprocessing
import numpy as np
import pandas as pd

targets = np.array(['открытый онлайнкурс', 'онлайнкурс', 'открытый
семинар', 'курс', 'направление дисциплин', 'программирован',
'студент', 'основ', 'вебдизайн'
])
labelEnc = preprocessing.LabelEncoder()
new_target = labelEnc.fit_transform(targets)
onehotEnc = preprocessing.OneHotEncoder()
onehotEnc.fit(new_target.reshape(-1, 1))
targets_trans = onehotEnc.transform(new_target.reshape(-1, 1))
print("Исходные данные")
print(targets)
print("Преобразованные данные с помощью OneHotEncoder")
print(targets_trans.toarray())

Вывод:
 Исходные данные
['открытый онлайнкурс' 'онлайнкурс' 'открытый семинар' 'курс'
 'направление дисциплин' 'программирован' 'студент' 'основ'
'вебдизайн']
Преобразованные данные с помощью OneHotEncoder
[[0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Листинг 1. Тестирование метода BinaryBOW

Как видно из примера, для небольшого по размерам словаря получается большая разреженная квадратная матрица с единственным ненулевым значением для каждого слова в каждой строке, это говорит о том, что чем больше слов в словаре, тем больше будет матрица.

Применение метода BinaryBOW на целом корпусе текстов приведет к созданию еще больших матриц**,** размерностью в несколько тысяч. Хранение и

обработка подобных данных требует больших вычислительных мощностей, вследствие чего было решено отказаться от данного метода. представление BinaryBOW является сопоставлением слова с вектором длины выбранного документа, в котором проставлена единица на позиции слова в словаре. Такой подход абсолютно не сохраняет никакой семантической близости между словами. При таком кодировании степень близости между любыми двумя словами будет равна нулю.

1.1.2. МЕТОД BAG OF WORDS

Отличительной особенностью метода Bag of words от метода BinaryBOW является то, что производится преобразование в вектор всего документа, и каждый элемент кодируется единицей по порядку следования слов в словаре. Это устраняет проблему размерности по первой оси, а количество строк задается количеством документов, однако в этом методе также есть недостаток: он не учитывает важность токенов, ведь одно слово может повторяться в документе.

Рассмотрим пример кодирования Bag of words на примере списка из n-грамм с повторяющимися словами. (Листинг 2).

```
from keras.preprocessing.text import Tokenizer
text = ['Открытый онлайн курс по дисциплине вебдизайна',
'Студенческий семинар по основам программирования']
# используем токенизатор
rep = model.texts_to_matrix(text, mode='count')
print(rep)
Вывод:
 Исходные данные
['открытый онлайнкурс' 'онлайнкурс' 'открытый семинар' 'курс'
 'направление дисциплин' 'программирован' 'студент' 'основ'
'вебдизайн']
Преобразованные данные с помощью Bag of words
[[0. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 1.]]
```

Листинг 2. Тестирование метода Bag of words

При реализации метода Bag of words вектору выделяется весь документ, и каждый элемент кодируется 1 по порядку следования слов в словаре. Применение этого метода решает проблему размерности матриц, но не учитывает вес слова в документе, ведь одно и то же слово может повторяться по несколько раз. Что влияет на синтаксический состав документа. И не подходит для работы с узкоспециализированными тематиками текстов. При таком подходе мы теряем всю информацию о взаимном расположении слов внутри текста, но, несмотря на это, закодированные таким образом текста уже можно сравнивать

1.1.3. МЕТОД TF-IDF

Широко применяемый и наиболее успешный в реализации при работе с текстами больших объемов метод TF-IDF. Его реализация основывается на том, что значимость n-граммы прямо пропорциональна частоте ее появления в документе и обратно пропорциональна доле документов в наборе, в которых эта n-грамма встречается.

Наибольший вес получает n-грамма, часто встречающаяся в одном документе, но не встречающаяся в остальных, а значит — отличающая этот документ от остальных. Признаки документов в этом подходе представляют собой произведение двух величин, частоты n-граммы (1.1) и обратной частоты документа (1.2):

$$TF_{token_i} = \frac{n_i}{N_i}, \quad (1.1)$$

$$IDF_{token} = log\,\frac{p}{P} \quad (1.2)$$

n_i– сколько раз встречается токен в i-ом документе,

N - общее количество токенов в i-ом документе,

p – количество документов, в которых встречается токен,

P – общее количеств документов.

TF-IDF – это произведение TF на IDF:

$$TF - IDF = TF \times IDF \quad (1.3)$$

Схема векторизация по методу TF-IDF хорошо работает с большими объемами текстовой информации. При дальнейшей обработке текста редкие слова и слова, которые встречаются во всех документах, несут мало информации и являются лишними при обработке текста, поэтому их удаляют. (Листинг 3).

```
# Векторизация Библиотекой Tfidf
from
  sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(p)
V = vectorizer.get_feature_names()
print(X.shape)
print(X)
print(V)

Вывод:
(9, 10)
  (0, 4)    0.7071067811865475
  (0, 6)    0.7071067811865475
  (1, 4)    1.0
  (2, 8)    0.7639653363561462
  (2, 6)    0.6452572857754033
  (3, 2)    1.0
  (4, 1)    0.7071067811865476
  (4, 3)    0.7071067811865476
  (5, 7)    1.0
  (6, 9)    1.0
  (7, 5)    1.0
  (8, 0)    1.0
['вебдизайн', 'дисциплин', 'курс', 'направление', 'онлайнкурс',
  'основ', 'открытый', 'программирован', 'семинар', 'студент']
```

Листинг 3. Тестирование метода TF-IDF

При реализации метод TF-IDF работает с значимостью n-граммы, которая прямо пропорциональна частоте ее появления в документе и обратно пропорциональна доле документов в наборе. Не создает большеразмерных матриц и не искажая синтаксической значимости текста.

При реализации 3 методов были получены следующие результаты:

- Метод BinaryBOW не подошел, так как показал большую размерность. Каждое предложение состояло из 10 и более слов, что в итоге приводило к большим матрицам для каждого документа. Количество строк регулируется словарем, поэтому чем больше слов в словаре, тем больше будет матрица. В реализации нашего проекта мы используем большие текстовые данные.
- Метод BinaryBOW не подошел, так как метод не учитывает важность того или иного токена, ведь одно слово может повторяться по несколько раз. Что влияет на синтаксический состав документа. Что очень важно при работе с узкоспециализированными текстами научных тематик.
- Метод TF-IDF. Его реализация основывается на том, что значимость n-граммы прямо пропорциональна частоте ее появления в документе и обратно пропорциональна доле документов в наборе, в которых эта n-грамма встречается. Поэтому, наибольший вес получает n-грамма, часто встречающаяся в одном документе, но не встречающаяся в остальных. Хорошо работает с текстами больших объемов, не создавая большеразмерных матриц и не искажая синтаксической значимости текста.

При выборе метода основная цель была чтобы алгоритм справлялся с большими объемами текстовых данных узкоспециализированной тематики. Таким образом, из результатов сравнения методов нам подошел метод TF-IDF.

1.2. ОБЗОР МЕТОДОВ КЛАСТЕРИЗАЦИИ ДОКУМЕНТОВ

Обширной областью работы с текстами является интеллектуальный анализ, целью которого является получение информации из коллекций текстовых документов, основываясь на применении эффективных методов машинного обучения и обработки естественного языка [2].

Особое внимание следует уделить поиску тематических текстов, так как в большинстве своем при запросе пользователь не имеет четкого понимания предметной области или самой информационной потребности. Вероятнее

решение заключается в навигации на основе кластеров и других методов исследовательского поиска, в которых часто используется кластеризация как один из этапов.

Кластеризация текстовых документов широко используется в информационном поиске, определении спама. В настоящее время остается актуальным изучение влияния специфики текстовой информации, в частности, принадлежность документов к одной предметной области или динамика изменения качества кластеризации в зависимости от использования целого текста, или его фрагментов [14].

Кластеризация текстовых документов, то есть разбиение множества документов на близкие по смыслу подмножества, является фундаментальной задачей поиска и обработки текстов. Ее результаты используются как для непосредственного анализа исходного множества документов, так и для информационного поиска, определения спама, помощи в проведении судебно-медицинских экспертиз и социологических исследований [4].

В большинстве научных статей, посвященных методам кластеризации текстов, документы представляются как векторы, в виде мешка слов (bag of words, bow). Таким образом, кластеризация документов рассматривается именно как кластеризация bow векторов.

Широко распространены и исследуются методы, основанные на мешке слов, извлечении терминологии, тематическом моделировании, а также векторном представлении слов и документов, полученном с помощью искусственных нейронных сетей, например, Word2Vec, Paragraph2Vec.

Andrews N. и Fox E. в своей работе [5] показали способы представления набора документов в виде векторной модели, а также некоторые способы предобработки текстов, и алгоритмы их кластеризации (модификации метода k-средних, EM-алгоритм и спектральная кластеризация). В результате чего были выявлены некоторые недостатки представления документов в виде мешка слов

- высокая размерность и разреженность полученных векторов. Авторы работы, показали методы понижения размерности векторного пространства.

В статье описаны шесть различных мер близости, между которыми проведено экспериментальное сравнение на алгоритме k-средних; лучшие результаты по метрикам чистоты и энтропии показала кластеризация, использующая в качестве меры близости коэффициент Жаккара и коэффициент корреляции Пирсона [5]. Авторы Sathiyakumari K. Manimekalai G. и др. [6] аналогично рассматривают кластеризацию документов только применительно к их представлению в виде мешка слов и выделяют четыре группы методов кластеризации: разделительная кластеризация, иерархическая кластеризация, k-средних и EM-алгоритм. Таким образом, кластеризация документов обычно сводится к кластеризации их векторных представлений в виде мешка слов.

Кластеризации текстовых документов обычно состоит из двух основных этапов. На первом этапе текстовые представления документов переводят в векторные представления, а на втором этапе к полученным векторам применяют методы кластеризации, основанные на расстоянии между векторами.

Кластеризация — это способ обработки данных, при котором производится разбиение множества объектов на группы, называемые кластерами. Внутри каждого кластера должны оказаться «похожие» объекты, а объекты разных групп должны максимально отличаться.

По способу разбиения на кластеры алгоритмы бывают двух типов: иерархические и неиерархические. Классические иерархические алгоритмы работают только с категорийными атрибутами, когда строится полное дерево вложенных кластеров. Распространены агломеративные методы построения иерархий кластеров – в них производится последовательное объединение исходных объектов и соответствующее уменьшение числа кластеров. Иерархические алгоритмы обеспечивают сравнительно высокое качество кластеризации и не требуют предварительного задания количества кластеров [7].

Другая группа - это неиерархические алгоритмы, определяющие оптимальное разбиение множества объектов на кластеры, число которых может основываться на предыдущих исследованиях, теоретических данных или каких-то фактах о кластеризуемых данных. В этой группе популярны алгоритмы семейства k-средних, которые в качестве целевой функции используют сумму квадратов взвешенных отклонений координат объектов от центров искомых кластеров. Некоторые алгоритмы кластеризации, например, k-means и LDA требуют, чтобы был указан или угадан кластер.

Кластер часто представляет собой область плотности в области признаков, где примеры области (наблюдения или строки данных) ближе к кластеру, чем другие кластеры. Кластер может иметь центр (центроид), который является выборкой или пространством точечных объектов, и может иметь границу или экстент.

Таким образом, кластерный анализ представляет собой итеративный процесс, в котором субъективная оценка процесса идентифицированных кластеров учитывается при изменении конфигурации до тех пор, пока не будет достигнут желаемый или подходящий результат.

Список из 10 наиболее популярных алгоритмов кластеризации представлен в таблице 1.

Таблица 1

Алгоритмы кластеризации

Название алгоритма	Параметры	Масштабируемость	Вариант использования	Геометрия (используется метрическая система)
К-средних	Количество кластеров	Очень большой n_samples, средний n_clusters с кодом MiniBatch	Универсальный, даже размер кластера, плоская геометрия, не слишком много кластеров, индуктивный	Расстояния между точками
Распространение сродства	Демпфирование, предпочтение образца	Не масштабируется с помощью n_samples	Множество кластеров, неравномерный размер кластера, неплоская геометрия, индуктивная	Расстояние графа (например, граф ближайшего соседа)
Средний сдвиг	Пропускная способность	Не масштабируется с помощью n_samples	Множество кластеров, неравномерный размер кластера, неплоская геометрия, индуктивная	Расстояния между точками
Иерархическая кластеризация отделения	Количество кластеров или порог расстояния	Большой n_samples и n_clusters	Многие кластеры, ограничения связи, трансдуктивные	Расстояния между точками
Агломеративная кластеризация	Количество кластеров или порог расстояния, тип связи, расстояние	Большой n_samples и n_clusters	Множество кластеров, возможно, ограничения связности, неевклидовы расстояния, трансдуктивные	Любое попарное расстояние

Продолжение таблицы 1

DBSCAN	Размер района	Очень большой n_samples, средний n_clusters	Неплоская геометрия, неодинаковые размеры кластеров, трансдуктивный	Расстояния между ближайшими точками
ОПТИКА	Минимальное членство в кластере	Очень большой n_samples, большой n_clusters	Неплоская геометрия, неравномерный размер кластеров, переменная плотность кластеров, трансдуктивный	Расстояния между точками
Иерархическая кластеризация отделения	Количество кластеров или порог расстояния	Большой n_samples и n_clusters	Многие кластеры, ограничения связи, трансдуктивные	Расстояния между точками
Гауссовы смеси	Многие	Не масштабируется	Плоская геометрия, подходит для оценки плотности, индуктивная	Махаланобис расстояния до центров
БЕРЕЗА	Фактор ветвления, порог, необязательный глобальный кластеризатор	Большой n_clusters и n_samples	Большой набор данных, удаление выбросов, сокращение данных, индуктивный	Евклидово расстояние между точками

ГЛАВА 2. ИНСТРУМЕНТАЛЬНЫЕ И ПРОГРАММНЫЕ СРЕДСТВА

2.1. ИНСТРУМЕНТЫ ДЛЯ РАЗРАБОТКИ СЕРВИСА

Для разработки сервиса был выбран язык программирования Python, так как он очень компактен, расширяем, читабелен и имеет множество открытых библиотек и модулей для работы с данными. Особенно стоит подчеркнуть библиотеки для Data Science: Scikit-learn, Matplotlib, Seaborn, Pandas, TensorFlow, nltk. Библиотеки включают различные алгоритмы Machine Learning, статистические инструменты, методы визуализации, интеллектуального анализа данных и методы обработки текстов на естественном языке.

Стек библиотек и методов ML, используемых в выпускной квалификационной работе представлен в таблице 2.

Таблица 2

Библиотеки и методы ML

Название библиотеки	Описание	Название метода	Описание метода
bs4	Библиотека Python для извлечения данных из файлов HTML и XML	find_all()	Находит все элементы, которые соответствуют заданным критериям. (Например, все теги 'li')
nltk	NLTK - это библиотека Python, используемая для обработки естественного языка человека	word_tokenize()	Разбивает абзац на отдельные слова.
		Stopwords()	Удаляет список стоп-слов с заранее определенным списком стоп-слов.

Продолжение таблицы 2

		Snowball Stemmer()	Извлекает основную форму слов путем удаления из них аффиксов.
sklearn.feature_extraction.text	Собирает данные для построения векторов признаков из текстовых документов.	TfidfVectorizer()	Преобразует набор необработанных документов в матрицу функций TF-IDF
sklearn.cluster	Собирает популярные алгоритмы кластеризации	KMeans()	Объединяет данные с похожими характеристиками вместе с помощью евклидова расстояния
sklearn.naive_bayes	Собирает контролируемые методы обучения, основанные на применении теоремы Байеса с предположениями о независимости функций.	ComplementNB	Используется для дополнения каждого класса для вычисления весов модели. Он особенно подходит для несбалансированных наборов данных.

Для написания программного кода приложения была выбрана среда разработки PyCharm. PyCharm — это интегрированная среда разработки для языка программирования Python, она предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django, в IDE есть поддержка проектов и системы управления версиями.

Django – это кросс-платформенная среда разработки, которая совместима с Windows, macOS, Linux. PyCharm Community Edition (бесплатная версия) находится под лицензией Apache License. IDE включает в себя инструменты разработки: встроенный терминал, интеграцию с системами управления версий и встроенными инструментами баз данных, возможность удаленной разработки

с удаленными интерпретаторами, интегрированный терминал ssh, интеграция с Docker и Vagrant [10].

2.2. СРЕДСТВА ОРГАНИЗАЦИИ ХРАНЕНИЯ ДАННЫХ

Для хранения с постоянным обновлением и извлечения данных в режиме реального времени было выбрано хранилище HBase [13].

HBase предоставляет произвольный доступ для чтения / записи в реальном времени к большим данным. В нее входит размещение очень больших таблиц - миллиардов строк X миллионов столбцов - на кластерах обычного оборудования. HBase - это распределенная нереляционная база данных с открытым исходным кодом, созданная по образцу Bigtable от Google: распределенная система хранения структурированных данных. Подобно тому, как Bigtable использует распределенное хранилище данных, предоставляемое файловой системой Google, HBase предоставляет возможности, подобные Bigtable, поверх Hadoop и HDFS.

Цель использования такого хранилища для текстов элективов - хранение версионности. Это необходимо для того, чтобы отслеживать наличия изменений. HBase также предоставляет линейную и модульную масштабируемость, строго согласованные чтение и запись, автоматическое и настраиваемое сегментирование таблиц, поддержку автоматического переключения при отказе между RegionServers, удобные базовые классы для поддержки заданий Hadoop MapReduce с таблицами HBase, простой в использовании Java API для клиентского доступа, блокировку кэша и фильтры Блума для запросов в реальном времени, передачу предиката запроса через фильтры на стороне сервера, экономичный шлюз и веб-сервис REST API с поддержкой XML, Protobuf и параметры кодирования двоичных данных, расширяемая оболочка на основе jruby (JIRB), поддержку экспорта метрик через подсистему метрик Hadoop в файлы или Ganglia; или через JMX.

Распределенное хранилище данных HBase - колоночно-ориентированная система, хранение данных в которой осуществляется по принципу

распределенной файловой системы Hadoop (HDFS), что положительно влияет на производительность операции чтения данных. Хранилище рассчитано на параллельную работу множества узлов [13].

Для хранения метаданных (текстов элективов) была выбрана PostgreSQL [14]. Данная СУБД бесплатна, часто используется для ведения баз данных веб-сайтов. Позволяет пользователям управлять как структурированными, так и неструктурированными данными. PostgreSQL может быть использован на большинстве основных платформ, включая Linux. СУБД справляется с задачами импорта информации из других типов баз данных с помощью собственного инструментария.

Ядро БД может быть размещено в ряде сред, в том числе виртуальных, физических и облачных. Самая свежая версия, PostgreSQL 9.5, предлагает обработку больших объемов данных и увеличение числа одновременно работающих пользователей. Безопасность была улучшена благодаря поддержке DBMS_SESSION. Основные достоинства СУБД - это масштабируемость и способность обрабатывать терабайты данных, поддержка формата json и множество предопределенных функций. Для управления СУБД было использовано веб-приложение pgAdmin 4 v4.0, которое позволяет через браузер осуществлять администрирование сервера, просматривать содержимое таблиц и баз данных, а также запускать команды SQL.

Для связи с PostgreSQL СУБД был использован модуль psycopg2 [20]. Для загрузки данных пользователь входит в WEB-интерфейс, вводит запрос поиска необходимого курса, после чего система подключается к ИС Modeus и начинается парсинг курсов на текущую дату. Данные загружаются и распределяются по необходимым ячейкам в памяти БД. В базе данных хранятся следующие таблицы: таблица с данными о курсах, таблица с данными о занятиях, таблица с данными о продолжительности занятия, таблица с данными авторов курсов, таблица с данными о кафедры курсов, таблица с данными технического оснащения для проведения учебного курса (см. Таблицы 3–8).

Таблица 3

Описание таблицы courses

№	Поле	Тип данных	Описание
1	id	uuid (PK)	id курса
2	catalog_code	VARCHAR(255)	код EL в каталоге
3	catalog_name	VARCHAR(255)	название EL
4	catalog_short_name	VARCHAR(255)	сокращенное название EL
5	lessons_count	INT	количество встреч
6	lessons_hours	INT	количество ак. часов
7	lesson_hours_units_code	VARCHAR(255)	код часов урока
8	full_description	TEXT	полное описание курса
9	description	TEXT	описание курса
10	status	VARCHAR(255)	статус EL (опубликован)
11	date_created	TIMESTAMP	дата создания
12	date_published	TIMESTAMP	дата публикации
13	date_archived	TIMESTAMP DEFAULT null	дата архивации
14	date_agreement	TIMESTAMP DEFAULT null	дата согласования
15	annotation	TEXT DEFAULT null	аннотации
16	throughput	INT DEFAULT null	max кол-во обучающихся
17	educational_result	TEXT DEFAULT null	результат обучения

Продолжение таблицы 3

18	knowledge_id	INT DEFAULT null	id знания
19	knowledge_parent_field_id	INT DEFAULT null	шкала оценивания
20	knowledge_title	VARCHAR(255) DEFAULT null	название области знаний
21	curator_person_id	UUID DEFAULT null	id куратора
22	author_person_ids	TEXT DEFAULT null	id автора (несколько)
23	requirement_ids	TEXT DEFAULT null	id требования (несколько)

Таблица 4

Описание таблицы courses_lessons

№	Поле	Тип данных	Описание
1	course_id	UUID (FK)	внеш.ключ из таб.courses(id)
2	id	UUID (PK)	id занятия
3	name	VARCHAR(255)	название занятия
4	type_code	VARCHAR(12)	код типа занятий
5	form_type_code	VARCHAR(255)	код типа формы занятий
6	form_type	VARCHAR(255)	форма занятия
7	description	TEXT	описание занятия
8	duration	INT	продолжительность пары
9	duration_units_code	VARCHAR(255)	Код (пара или ак. час)

10	throughput	INT	max кол-во обучающихся

Таблица 5

Описание таблицы courses_lessons_hours

№	Поле	Тип данных	Описание
1	course_id	UUID (FK)	внеш. ключ из таб. courses (id)
2	type_code	VARCHAR(12)	код типа занятий
3	type_label	VARCHAR(255)	учебные встречи
4	count	INT	кол. встреч
5	hours	INT	кол. часов
6	hours_units_code	VARCHAR(255)	Код (пара или ак. час)

Таблица 6

Описание таблицы courses_persons

№	Поле	Тип данных	Описание
1	id	UUID (PK)	id
2	course_id	UUID (FK)	внеш.ключ таб. courses(id)
3	name	VARCHAR(255)	имя автора
4	surname	VARCHAR(255)	фамилия
5	middle_name	VARCHAR(255)	отчество
6	birth_date	TIMESTAMP DEFAULT null	возраст

Продолжение таблицы 6

7	gender	VARCHAR(255) DEFAULT null	пол
8	description	VARCHAR(255) DEFAULT null	описание
9	phone	VARCHAR(255) DEFAULT null	телефон
10	email	VARCHAR(255) DEFAULT null	адрес электронной почты
11	other_contacts	VARCHAR(255) DEFAULT null	другие контакты

Таблица 7

Описание таблицы courses_persons_groups

№	Поле	Тип данных	Описание
1	id	UUID (PK)	id кафедры
2	course_id	UUID (FK)	внеш. ключ таб. courses(id)
3	name	VARCHAR(255)	кафедра

Таблица 8

Описание таблицы courses_requirements

№	Поле	Тип данных	Описание
1	requirement_id	UUID (PK)	id требования
2	lesson_id	UUID (FK)	внеш. ключ courses_lessons(id)
3	at_type	VARCHAR(255)	тип оборудования

4	type	VARCHAR(255) DEFAULT null	тип

Продолжение таблицы 8

5	description	TEXT DEFAULT null	описание
6	category_code	VARCHAR(255) DEFAULT null	код категории
7	classroom_type_code	VARCHAR(255) DEFAULT null	вид аудитории
8	role_name	VARCHAR(255) DEFAULT null	вид занятия
9	teachers_number	INT DEFAULT null	количество преподавателей
10	students_group_size	INT DEFAULT null	размер группы студентов на занятии
11	provision_quantity	INT DEFAULT null	резерв
12	min_capacity	INT DEFAULT null	минимальная вместимость
13	rooms_quantity	INT DEFAULT null	количество аудиторий
14	preparation_time	VARCHAR(255) DEFAULT null	время обучения
15	preparation_type_code	VARCHAR(255) DEFAULT null	код вида обучения
16	lesson_relation_type	VARCHAR(255) DEFAULT null	тип обучения уроку
17	lesson_relation_type_code	VARCHAR(255) DEFAULT null	код типа обучения уроку
18	spec_lesson_id	VARCHAR(255) DEFAULT null	id спец урока
19	prev_lesson_type	VARCHAR(255) DEFAULT null	предыдущий тип урока(лекция/семинар)
20	min_interval_in_days	INT DEFAULT null	мин. интервал в днях

Схема базы данных отображена на рисунке 1.

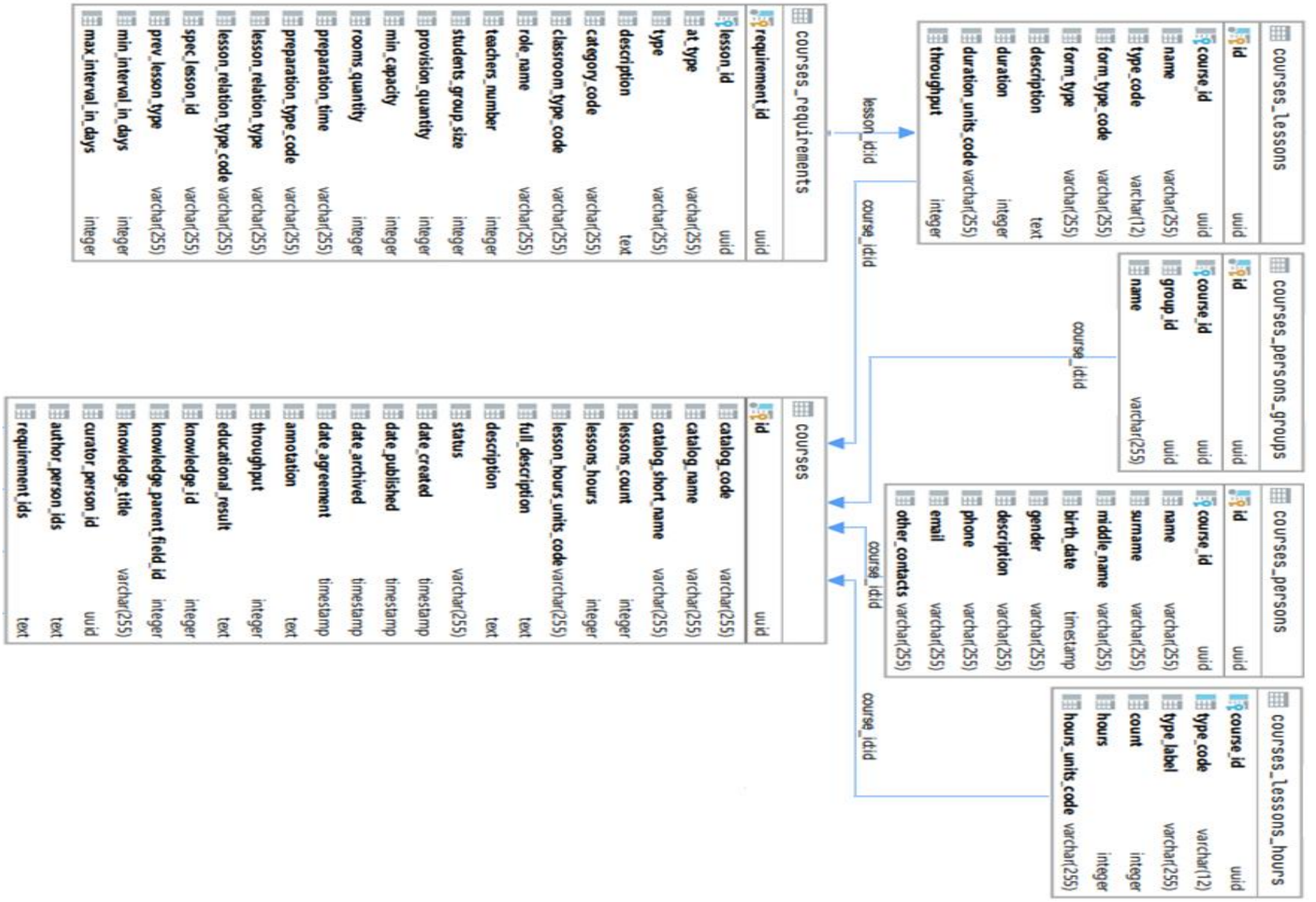

Рис. 1. Схема базы данных PostgreSQL для хранения элективов

ГЛАВА 3. РАБОТА С ДАННЫМИ

В работе было реализовано два подхода для работы с данными учебных курсов:

- Первый подход включает в себя тестирование и методов кластеризации и классификации данных (пункт 3.1.).
- Второй подход включает в себя анализ данных, их очистку, предобработку текста и кластеризацию элективов разными методами (пункт 3.2.).

В качестве данных в работе использовали наборы текстов учебных курсов из ИС Modeus.

Состав данных:

- Формат представления – файлы расширения json;
- Название курса, краткая аннотация, полная аннотация, автор, академические часы, область знаний, дата создания;
- Объем датасета: 921 электив (12.03.2021).

3.1. ВЫБОР АЛГОРИТМОВ И МЕТОДОВ КЛАСТЕРИЗАЦИИ И КЛАССИФИКАЦИИ

3.1.1. ВЫБОР МЕТОДА КЛАСТЕРИЗАЦИИ

В результате исследований из 10 рассмотренных методов кластеризации было выбрано 3 метода, наиболее удовлетворяющих входным параметрам данных.

Были выбраны методы, которые на вход получают разреженную матрицу данных, так как большое количество признаков (тексты элективов) и не нуждаются в указании количества кластеров.

Такими методами являются: DBSCAN, Affinity propagation, BIRCH.

Метод DBSCAN начинает выполнение на случайном объекте выборки: он определяет, есть ли в окрестности радиуса этого объекта количество объектов, не меньшее заранее заданного параметра, и, если есть, определяет

эту окрестность как кластер; далее все объекты, лежащие в окрестности кластера, присваиваются этому кластеру. Это повторяется до тех пор, пока есть не посещенные объекты. Если в итоге объект оказывается не принадлежащим ни одному кластеру, он помечается как шум.

После предварительных экспериментов, было решено не проводить исследование алгоритма DBSCAN, так как он продемонстрировал слишком высокую чувствительность к выбору параметров (min_samples и eps); кроме того, DBSCAN достаточно много объектов не присваивает ни к одному из кластеров, помечая их как шум, что затрудняет его сравнение с другими алгоритмами кластеризации.

Метод Affinity propagation принимает в качестве входных данных меры сходства между парами точек данных и одновременно рассматривает все точки данных как потенциальные образцы. Между точками данных происходит обмен сообщениями с действительными значениями до тех пор, пока постепенно не появится высококачественный набор образцов и соответствующих кластеров.

В качестве входных данных алгоритм требует предоставления двух наборов данных:

Данные сходства между точками данных, показывающие, насколько хорошо одна точка может служить примером для другой. Если между двумя точками нет сходства, так как они не могут принадлежать одному кластеру, это сходство можно опустить или установить на -Infinity в зависимости от реализации.

Данные сходства и предпочтения часто представлены в виде единой матрицы, где значения на главной диагонали представляют предпочтения. Матричное представление подходит для плотных наборов данных. Если связи между точками редки, практичнее не хранить всю матрицу nxn в памяти, а вместо этого хранить список сходств с подключенными точками.

С помощью данного метода было произведено разбиение на 200 кластеров по всем признакам набора данных. Количество элективов в каждом

кластере распределено равномерно: нет резких перекосов в сторону одного кластера или принадлежности ни к одному из кластеров (Рисунок 2).

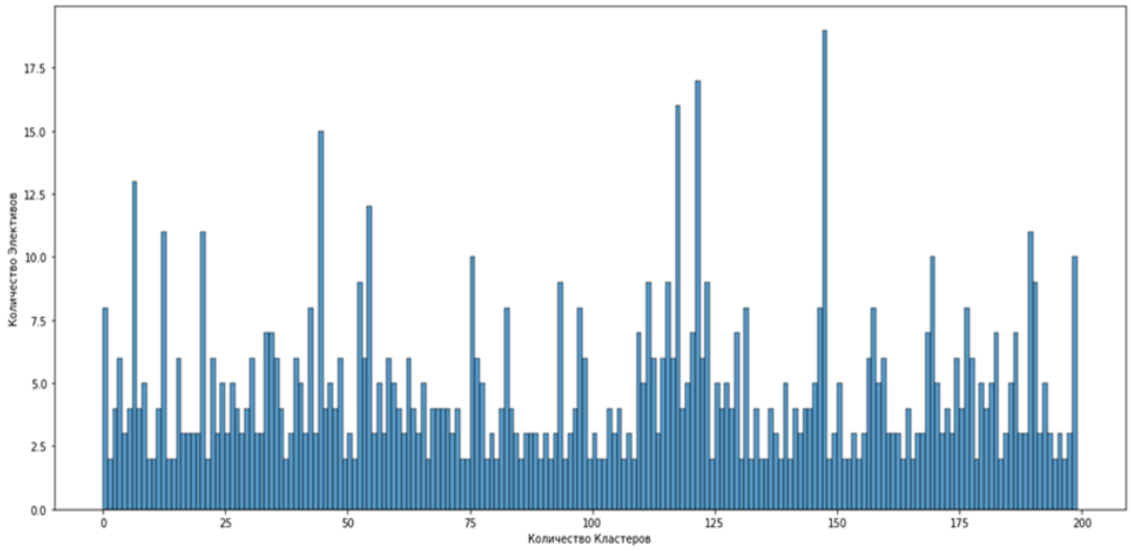

Рис. 2. Метод кластеризации Affinity propagation

При реализации метод Affinity propagation показал равномерность распределения по кластерам, схожесть элективов в одном кластере. Метод был исключен, так как среднее количество элективов в кластере, не превышало пяти, что привело к перенасыщению кластерами и не показывало полный объем курсов, удовлетворяющих пользовательскому запросу.

Метод BIRCH строит древовидную структуру данных с центроидами кластера в виде листовых узлов.

Алгоритм BIRCH имеет два параметра: порог и коэффициент ветвления. Коэффициент ветвления ограничивает количество подкластеров в узле, а порог ограничивает расстояние между входящей выборкой и существующими подкластерами.

Этот алгоритм можно рассматривать как экземпляр или метод сокращения данных, поскольку он сводит входные данные к набору подкластеров. Эти сокращенные данные могут быть дополнительно обработаны путем подачи их в глобальный кластеризатор.

Новый курс вставляется в корень дерева CF, который является узлом CF. Затем он объединяется с подкластером корня, который имеет наименьший радиус после слияния, ограниченный пороговыми условиями и

условиями фактора ветвления. Если в подкластере есть дочерний узел, то слияние повторяется до тех пор, пока он не достигнет листа. После нахождения ближайшего подкластера в листе свойства этого подкластера и родительских подкластеров рекурсивно обновляются.

Если радиус подкластера, полученного путем слияния новой выборки и ближайшего подкластера, больше квадрата порога и, если количество подкластеров больше, чем коэффициент ветвления, то для этой новой выборки временно выделяется пространство. Берутся два самых дальних подкластера, и подкластеры делятся на две группы на основе расстояния между этими подкластерами. Если у этого разбитого узла есть родительский подкластер и есть место для нового подкластера, то родительский подкластер разбивается на два. Если места нет, то этот узел снова делится на две части, и процесс продолжается рекурсивно, пока не достигнет корня [9].

С помощью данного метода было произведено разбиение исходных данных на 418 кластеров по всем признакам набора данных. Хотя алгоритм показал высокую скорость, возникла сложность с оценкой параметра T (порог, который ограничивает расстояние между входящей выборкой и существующими подкластерами), обеспечивающего получение требуемого количества первичных кластеров. Ошибки в ходе кластеризации были обусловлены тем, что для добавляемой в дерево точки очень часто находился не самый близкий к ней листовой узел, поэтому центром кластера мог быть задан не совсем близкий для всех элективов параметр. Распределение элективов по кластерам оказалось неравномерным: был перекос в сторону одного кластера (339 кластер объединил 84 электива), в котором оказались собраны не совсем близкие элективы (Рисунок 3). Исходя из перекоса в сторону одного кластера, метод BRICH был исключен

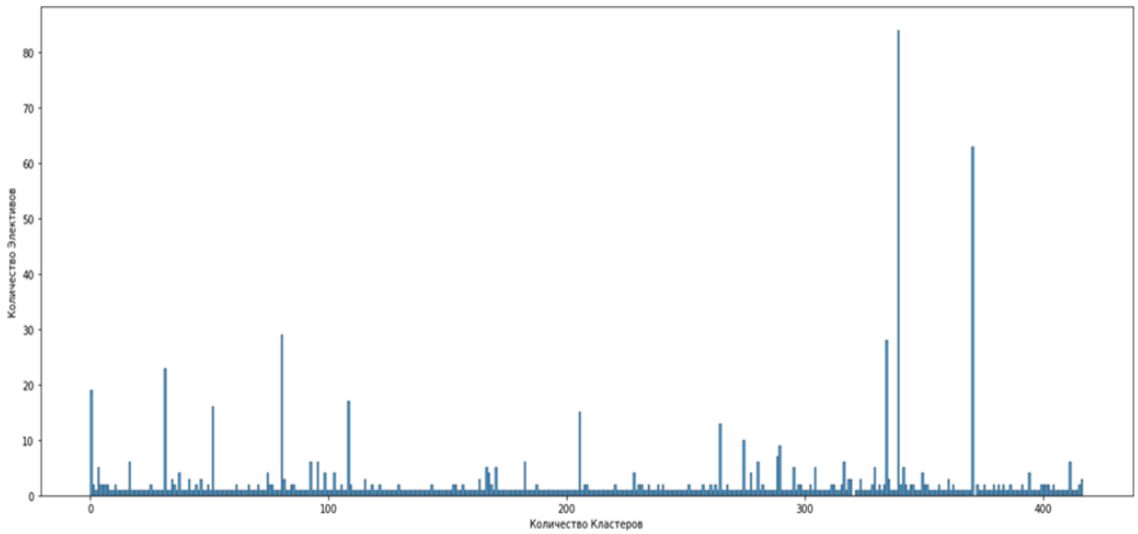

Рис. 3. Метод кластеризации BIRCH

Метод Kmeans разбивает множество элементов векторного пространства на заранее известное число кластеров k. Метод стремится минимизировать среднеквадратичное отклонение на точках каждого кластера. Основная идея заключается в том, что на каждой итерации выполняется вычисление центра масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике. Алгоритм завершается, когда в течении итерации не происходит изменения кластеров.

Этот метод позволяет задать количество кластеров. Данным методом было произведено разбиение по всем признакам набора данных на 10 кластеров (см. рисунок 4, таблица 9), на 20 кластеров (см. рисунок 5, таблица 10) и 25 кластеров (см. рисунок 6, таблица 11).

Таблица 9

Распределение элективов на 10 кластеров методом Kmeans

Номер кластера	Количество элективов
0	50
1	196
2	90
3	125
4	76
5	23
6	104
7	51
8	58
9	148

При разбиении элективов методом Kmeans на 10 кластеров получилось неравномерное распределение в 1, 5, 9 кластерах.

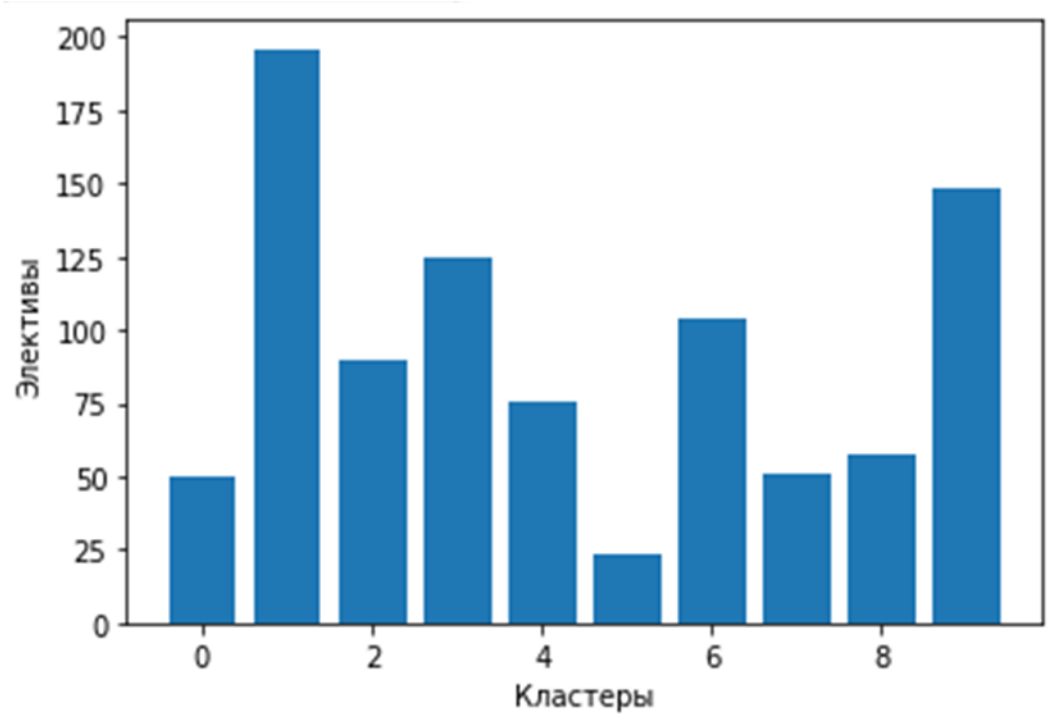

Рис.4. Распределение элективов на 10 кластеров методом кластеризации Kmeans

Таблица 10

Распределение элективов на 20 кластеров методом Kmeans

Номер кластера	Количество элективов
0	21
1	15
2	40
3	33
4	11
5	21
6	73
7	47
8	52
9	139
10	65

Продолжение таблицы 10

11	108
12	10
13	15
14	22
15	32
16	31
17	32
18	11
19	143

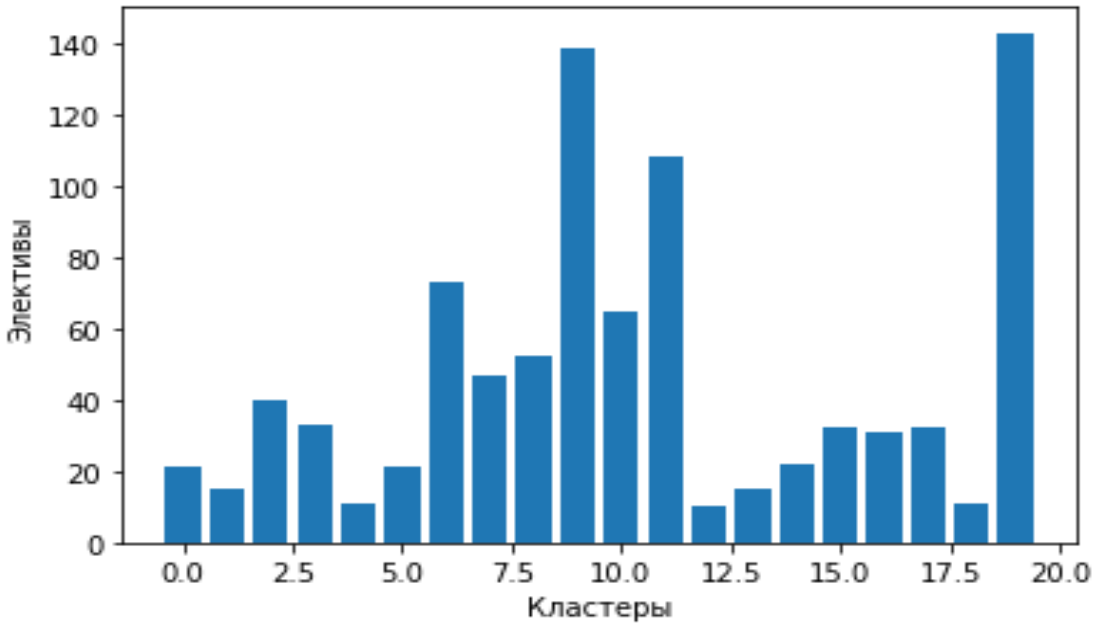

Рис.5. Распределение элективов на 20 кластеров методом кластеризации Kmeans

При разбиении элективов методом Kmeans на 20 кластеров получилось неравномерное распределение курсов в 4, 9, 11, 12, 13, 14,18, 19 кластерах.

Это значит, что алгоритм некорректно сработал и отцентровал в большие кластеры курсы, не являющиеся близкими по содержанию.

Таблица 11

Распределение элективов на 25 кластеров методом Kmeans

Номер кластера	Количество элективов
0	14
1	54
2	60
3	12
4	34
5	18
6	63
7	88
8	63
9	82
10	43
11	53
12	31
13	38
14	13
15	15
16	21
17	10
18	36
19	34

Продолжение таблицы 11

20	19
21	31
22	36

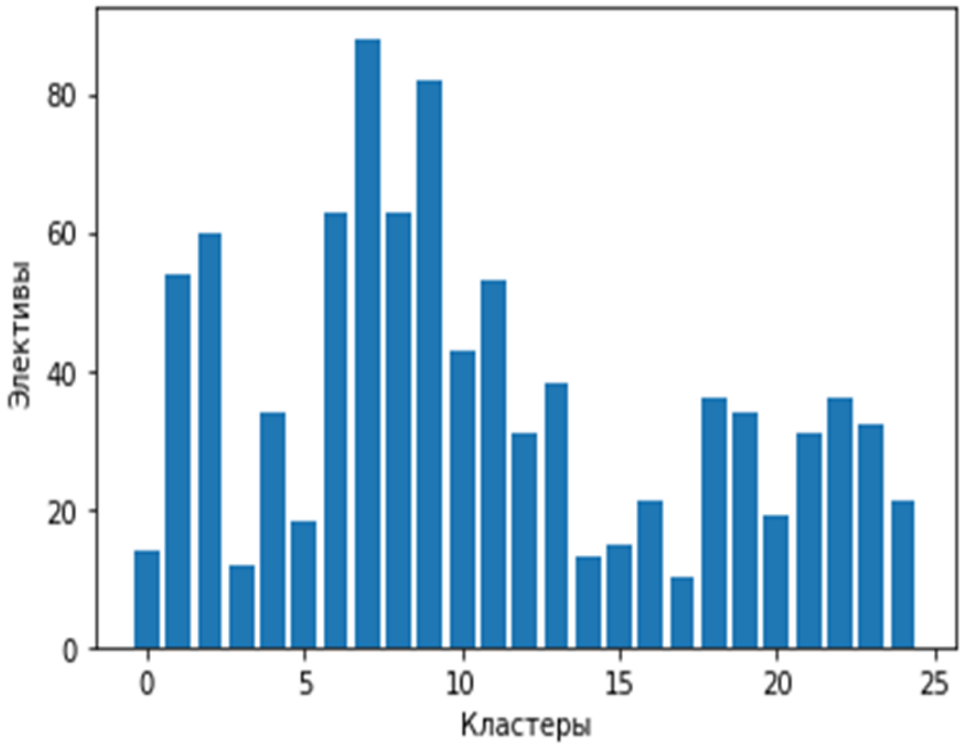

Рис.6. Распределение элективов на 25 кластеров методом кластеризации Kmeans

Из рисунка 6 видно, что наиболее равномерное распределение показало разбиение на 25 кластеров. Метод показал хорошие результаты по схожести элективов в одном кластере. Среднее количество в одном кластере не превышает 35.

3.1.2. ВЫБОР МЕТОДА КЛАССИФИКАЦИИ

Для реализации поиска по запросу пользователя было решено выбрать классификатор. Для сравнения было выбрано 3 классификатора: RandomForestClassifier, ComplementNB, SVC [15,16,18].

Тестирование проводилось на тестовой выборке (разбиение на выборки было в соотношении 30%-тестирующая, 70%-обучающая). Результаты работы классификаторов представлены на рисунках 6, 7, 8. Для оценки качества работы алгоритма на каждом из классов по отдельности использовались метрики precision (точность – доля объектов, классифицированных положительными и такими являющимися) и recall (полнота – доля объектов положительного класса из всех объектов положительного класса обнаружил алгоритм) [Precision-Recall].

- RandomForestClassifier (Случайный лес) – для выполнения классификации производится построение решающих деревьев для случайным образом сгенерированных подвыборок на основе исходной. Полученные решающие деревья классифицируют элемент отдельно друг от друга, затем классифицируемому объекту присваивается тот класс, к которому его отнесло большинство деревьев.

  Характеристики классификатора RandomForestClassifier представлены в таблице 12.
- SVC (Метод опорных векторов) – классификация путем нахождения наиболее правильной линии, или гиперплоскости разделяющую данные на классы.

  Характеристики классификатора SVC представлены в таблице 13.
- ComplementNB (Наивный байесовский метод) – классификатор хорошо подходит для работы с несбалансированными наборами данных. В дополнение к Наивному Байесу, вместо вычисления вероятности принадлежности предмета к определенному классу, вычисляет вероятность принадлежности предмета ко всем классам. Для каждого класса вычислите вероятность того, что данный экземпляр ему не

принадлежит. После расчета для всех классов проверяет все рассчитанные значения и выбирает наименьшее значение (что означает наибольшую вероятность принадлежности к данному классу).

Характеристики классификатора ComplementNB представлены в таблице 14).

Таблица 12

Характеристики метода классификации RandomForestClassifier

Номер кластера	Precision	Recall	F1-score	Support
0	1.00	0.50	0.67	2
1	0.83	1.00	0.91	15
2	0.68	0.94	0.79	16
3	1.00	1.00	1.00	3
4	1.00	0.50	0.67	6
5	1.00	0.22	0.36	9
6	0.45	0.89	0.60	19
7	0.29	0.64	0.40	22
8	0.62	0.76	0.68	17
9	0.70	0.78	0.74	27
10	0.92	0.67	0.77	18
11	0.83	0.29	0.43	17
12	0.86	0.75	0.80	8
13	0.75	0.23	0.35	13
14	1.00	0.67	0.80	3
15	1.00	1.00	1.00	3
16	1.00	0.80	0.89	10
17	1.00	0.33	0.50	3
18	0.80	0.57	0.67	14
19	0.25	0.29	0.27	7
20	1.00	0.14	0.25	7

21	0.86	0.67	0.75	9
22	0.90	0.75	0.82	12

Продолжение таблицы 12

23	1.00	0.45	0.62	11
24	0.83	0.83	0.83	6

Таблица 13

Характеристики классификации SVC

Номер кластера	Precision	Recall	F1-score	support
0	0.00	0.00	0.00	3
1	1.00	1.00	1.00	16
2	0.92	0.96	0.64	23
3	1.00	0.75	0.86	4
4	0.91	0.71	0.80	14
5	1.00	0.50	0.67	4
6	0.82	0.61	0.70	23
7	0.25	0.73	0.37	22
8	0.86	0.86	0.86	22
9	0.68	1.00	0.81	21
10	1.00	0.57	0.73	14
11	0.57	0.44	0.50	18
12	0.90	0.64	0.75	14
13	0.83	0.42	0.56	12
14	1.00	0.83	0.91	6
15	1.00	0.50	0.67	2
16	1.00	1.00	1.00	7
17	1.00	0.33	0.50	3
18	1.00	0.71	0.83	7
19	0.33	0.12	0.18	8

Продолжение таблицы 13

20	1.00	0.20	0.33	5
21	0.80	1.00	0.89	8
22	0.71	0.62	0.67	8
23	1.00	0.50	0.67	6
24	0.83	0.71	0.77	7

Таблица 14

Характеристики классификации ComplementNB

Номер кластера	Precision	Recall	F1-score	Support
0	0.00	0.00	0.00	3
1	0.82	1.00	0.90	18
2	0.72	1.00	0.94	21
3	1.00	1.00	1.00	2
4	1.00	0.86	0.92	14
5	1.00	1.00	1.00	2
6	0.94	0.65	0.77	23
7	0.42	0.52	0.47	21
8	0.58	0.63	0.65	15
9	0.62	0.89	0.73	18
10	0.78	0.82	0.80	17
11	0.64	0.43	0.51	21
12	0.67	0.75	0.71	8
13	0.62	0.71	0.67	7
14	1.00	0.88	0.93	8
15	1.00	0.60	0.75	5
16	0.83	1.00	0.91	5

Продолжение таблицы 14

17	1.00	0.60	0.75	5
18	0.55	1.00	0.71	6
19	0.62	0.56	0.59	9
20	1.00	0.11	0.20	9
21	0.91	0.83	0.87	12
22	0.75	0.90	0.82	10
23	0.89	0.67	0.76	12
24	1.00	0.50	0.67	6

Таблица 15

Сравнительная характеристика методов классификации

	Доля верных предсказаний			Полнота			F-мера		
Характеристики	Случайный лес	Метод опорных векторов	Байесовский классификатор	Случайный лес	Метод опорных векторов	Байесовский классификатор	Случайный лес	Метод опорных векторов	Байесовский классификатор
Точность	___	___	___	___	___	___	0.65	0.70	0.73
Макро - среднее	0.82	0.82	0.77	0.63	0.63	0.72	0.66	0.68	0.72
Среднее взвешенное	0.75	0.79	0.76	0.65	0.70	0.73	0.65	0.71	0.72

По результатам наибольшую точность (0.73) показал классификатор ComplementNB, который почти полностью определил каждый класс, за исключением нулевого, так как этот класс содержит наименьшее число

объектов.

3.1.3. РЕАЛИЗАЦИЯ АЛГОРИТМА ПОИСКОВОГО ЗАПРОСА

После ввода текста запроса данные преобразуются в числовой вектор дальше алгоритм кластеризации разбивает их в кластеры и производит группировку. Схема работы кластеризатора представлена на рисунке 7.

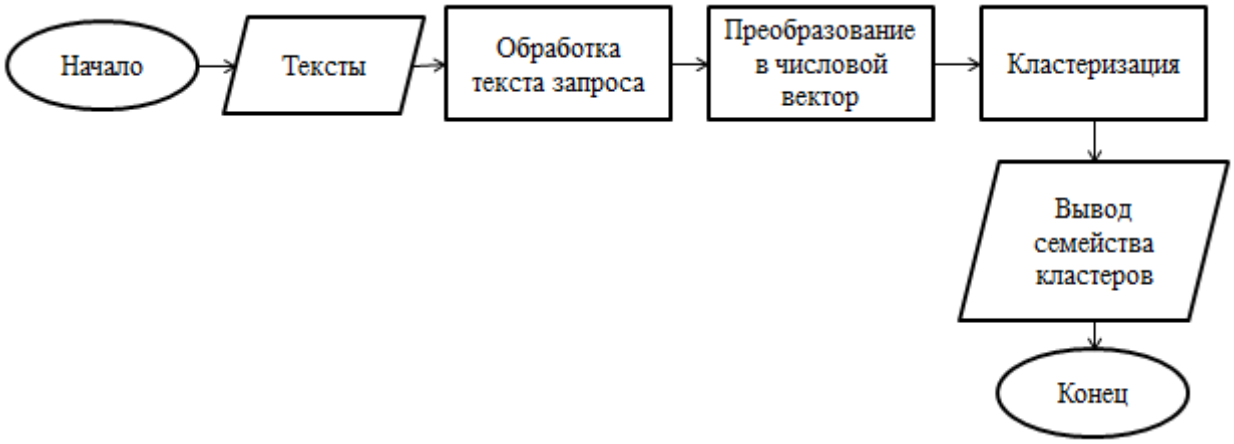

Рис. 7. Схема работы кластеризатора

Классификация нужна при добавления новых текстов курсов и реализуется следующим образом: выделяются признаки из текста и с помощью классификатора, основанного на обучающем наборе данных, определяется, к какому классу относится электив. Схема работы классификатора представлена на рисунке 8.

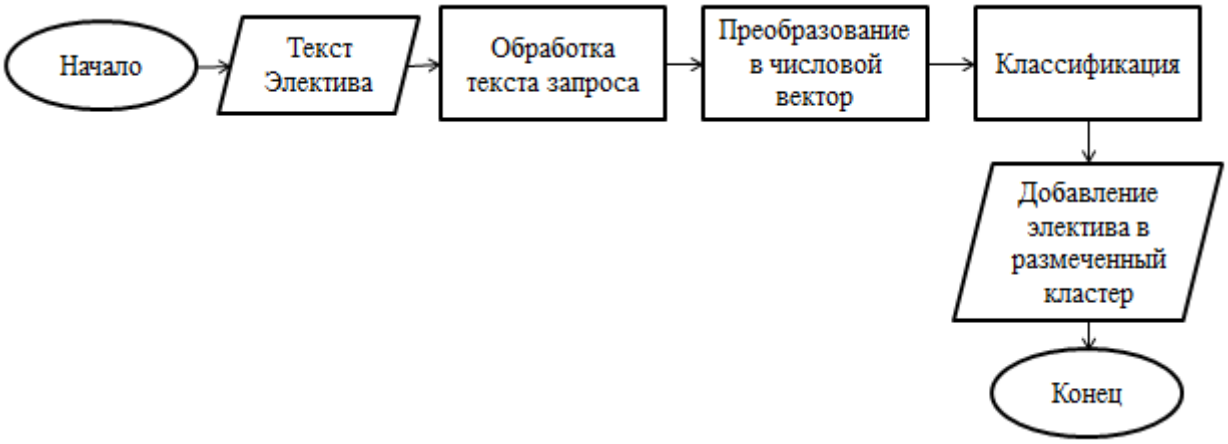

Рис. 8. Схема работы алгоритма поискового запроса первого подхода

3.2. АНАЛИЗ ДАННЫХ И КЛАСТЕРИЗАЦИЯ

Второй подход ориентирован в первую очередь на анализ и обработку данных и только потом на кластеризацию. В ходе выполнения работы были выделены следующие этапы:

- Анализ исходной выборки и выбор компонент для кластеризации
- Очистка выборки
- Предобработка текстовых данных
- Стемминг и лемматизация текста
- Кластеризация обработанных текстов

На рисунке 9 представлена схема процесса подготовки данных.

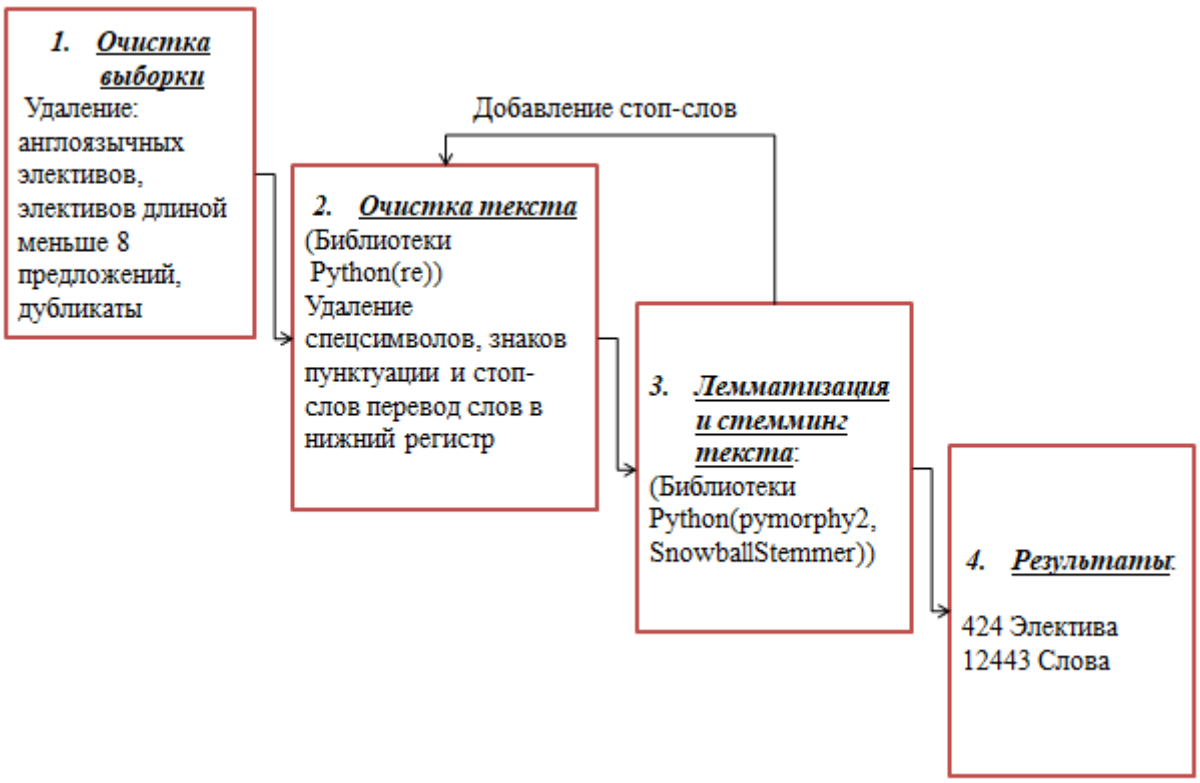

Рис. 9. Схема обработки данных для второго подхода

3.2.1. АНАЛИЗ ИСХОДНОЙ ВЫБОРКИ

Из образовательного портала Modeus была выгружена информация о 921 элективе. Информация об элективах содержит в себе следующий текстовые данные: название электива, аннотация, расширенное описание, образовательный результат, предусловия, авторы курса, ответственный за

курс. Из всех текстовых составляющих для обработки и кластеризации были выбраны название электива, расширенное описание и образовательный результат, объединенные в единый текст, исключение аннотации обусловлено стремлением сокращения пересечения информации в рамках одного электива. При анализе полученных данных было замечено, что в наборе данных присутствуют дубликаты – элективы, взятые данные из которых совпадают полностью.

Из 921 электива было обнаружено 246 дубликатов (26,7% от общего числа элективов). Дальнейшее исследование данных показало, что в наборе присутствуют элементы с небольшим количеством предложений, при этом, вместо предполагаемых описания и образовательного результата, было описание курса в расплывчатой формулировке, и представлены ссылки на внешние ресурсы. В листинге 4 представлен один из подобных элективов.

```
for i in data:

    if len((i[0]+i[1]+i[2]).split('.'))<8:

        badEl.append(i)

print("Name:",badEl[0][0],"Description:",badEl[0][2],"Educa
tionalResult:",badEl[0][2])



Name: Ecology.Techology. Anime/Экология. Технологии. Аниме
Description: Дисциплина проводится в формате открытого
онлайн-курса на платформе "Лекториум"
(https://www.lektorium.tv/anime) EducationalResult: Знать
Уметь Владеть
```

Листинг 4. Пример электива с малым количеством предложений

Из данных по подобным элективам информацию о нем указывает разве что название. Подобные малоинформативные элективы не удастся успешно кластеризовать вместе с обычными, в которых присутствует описание и образовательный результат.

3.2.2. ОЧИСТКА ВЫБОРКИ

Для повышения качества кластеризации было решено произвести работу над выборкой. Так как кластеризация выполняется для элективов на русском языке, было решено удалить элективы на других языках. Также удалению подверглись повторяющиеся элективы.

Малоинформативные элективы не получится анализировать методами анализа. Замечено, что у малоинформативных элективов есть одно общее качество: сумма предложений их текстовых составляющих была заметно ниже таковой у обычных элективов. В связи с этим была выдвинута гипотеза о том, что удаление элективов с малым количеством предложений положительно скажется на результатах кластеризации.

В качестве критерия качества кластеризации было решено взять среднее арифметическое между расстояниями запросов и центроидами ближайших кластеров. Удаление элективов с малым количеством предложений показало, уменьшение среднего расстояния, поэтому было принято решение убрать из выборки все элективы, длина которых меньше порогового значения.

Динамика изменения характеристик выборки представлена в таблице 16.

Таблица 16

Изменение выборки при удалении элективов

Минимальная длина информации об элективе в предложениях	Количество неподходящих элективов /количество повторов	Количество подходящих элективов	Среднее расстояние до ближайшего кластера
5	143/208	570	1.713
6	196/198	527	1.825
7	269/172	480	1.287
8	355/142	424	0.929

Из таблицы видно, что удаление элективов с малым количеством предложений положительно сказалось на динамике уменьшения расстояния до центров кластеров, поэтому было принято решение об исключении из выборки всех элективов, длина текстовой информации о которых была меньше восьми предложений.

3.2.3. ПРЕДОБРАБОТКА ТЕКСТОВЫХ ДАННЫХ

Для дальнейшего повышения качества кластеризации была предпринята предобработка текста. Текст элективов представляет из себя не только набор слов, но еще и знаки препинания, разметки, теги и ссылки. Подобные элементы используются для оформления текста, при этом сами элементы никакой смысловой нагрузки в себе не несут. Их присутствие в кластеризируемых данных влечет за собой снижение эффективности кластеризации. Отдельно стоит упомянуть слова, которые сами по себе не несут никакой смысловой нагрузки, так называемые «стоп-слова». В большинстве своем стоп-слова в русском языке представлены служебными частями речи, которые без контекста не имеют смысла. По итогу, для успешной кластеризации были совершены следующие шаги по предобработке текста:

- Удаление спецсимволов и разметки текста
- Удаление ссылок
- Удаление знаков пунктуации
- Перевод текста в нижний регистр
- Удаление стоп-слов
- Стемминг и лемматизация слов

3.2.4. КЛАСТЕРИЗАЦИЯ ОБРАБОТАННЫХ ТЕКСТОВ

В рамках исследовательской работы была произведена кластеризация данных с помощью двух разных методов: Doc2Vec и TF-IDF+K-means. Это популярные методы для кластеризации текстов, при этом каждый из них использует отличное от остальных представление текста и механизм работы

с ним, что позволяет посмотреть на особенности данных с разных сторон. В качестве критериев определения успешности кластеризации были выбраны расстояния запросов до ближайших элективов и экспертная проверка соответствия темы запроса и предложенных элективов. В случае, если расстояние до пяти ближайших предложенных элективов не превышает полутора единиц, а также эксперт подтвердит, что темы элективов и запросов совпадают, кластеризацию можно считать успешной [11,12].

Метод Doc2Vec учитывает семантическую близость слов. Алгоритм основывается на предположении, что в одинаковом контексте встречаются похожие по значению слова. Для этого в алгоритме присутствуют два метода обхода по тексту: CBOW (непрерывный мешок слов) и Skip-gram. Метод CBOW определяет вероятность слова по контексту, в то время как метод Skip-gram пытается определить контекст по слову. При обходе всего набора данных для каждого слова формируется вектор значений (Рисунок 10).

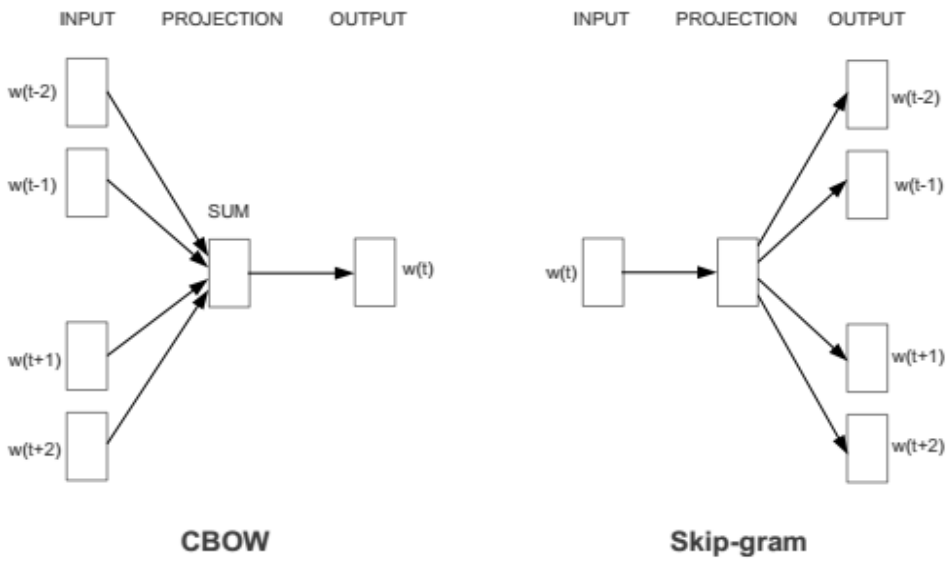

Рис. 10. Схема работы методов CBOW и Skip-gram [15]

Рассмотрим результаты кластеризации методом Doc2Vec для запроса «Машинное обучение» (Рисунок 11). В качестве аргументов функции

передается строка запроса. По окончании поиска выводятся названия, расстояние и сокращенный текст найденных ближайших элективов.

```
Метод: модель Doc2Vec
Запрос
Машинное обучение
Расстояние: 3.413 Название электива: Тайм-менеджмент  Электив:  электив разработан направления менеджмент изуче
ния дисциплины достижение студентом понимания проблем управления рабочим временем руководителя планированием ко
нтролем. задачи формирование студента представления роли работе менеджера организации форми

Расстояние: 3.768 Название электива: Тайм-менеджмент  Электив:  электив разработан направления менеджмент изуче
ния дисциплины достижение студентом понимания проблем управления рабочим временем руководителя планированием ко
нтролем. задачи формирование студента представления роли работе менеджера организации форми

Расстояние: 3.800 Название электива: Фото- и видеоискусство Электив:  данная дисциплина ориентирована практику
занятия проходят обязательным выполнением творческих заданий. предполагаются занятие аудитории съемки пленере с
тудии мероприятии. большую роль играет знакомство принципами работы известных зарубежных местных

Расстояние: 4.005 Название электива: Аргументация и логическая прагматика Электив: цель курса познакомить студе
нтов проблематикой основными идеями логической прагматики теории аргументации. задачи формирование знаний умени
й навыков непосредственно связанных аргументацией. развитие навыков самостоятельного рационального анализа проб
```

Рис. 11 Результат запроса кластеризации Doc2vec

Как видно из рисунка, расстояние от запроса до найденного электива заметно выше принятого порогового значения, а экспертная проверка показала несоответствие темы запроса и электива между собой. На основании полученных данных от метода Doc2Vec пришлось отказаться.

Метод TF-IDF+K-means основывается на представлении каждого текста как вектора, компонентами которого являются значение произведения характеристик TF (частота слова) и IDF (обратная частота документа) для каждого слова во всех текстах. Применение данной меры отражает условную «ценность» слова: малоупотребительные слова, такие как термины, имеют большее значение характеристики, чем распространенные слова, например, числительные и общие слова. Полученные векторы кластеризируются по алгоритму K-means. Алгоритм заключается в следующих действиях:

1. Случайная генерация центроидов кластеров
2. Определение элементов к ближайшему кластеру
3. Пересчет центроидов кластеров как среднее его элементов
4. Повторение шагов 2 и 3 до тех пор, пока состав кластеров не перестанет изменяться.

В ходе тестирования алгоритма было обнаружено эффективное разбиение элективов на 10 кластеров. Полученные кластеры содержали в себе элементы преимущественно одной тематики, что позволило охарактеризовать кластер понятным для человека способом – темой. Так же для каждого кластера были автоматически определены ключевые слова, которые, по экспертной оценке, совпадали с темой кластера.

Дополнительно был произведен подсчет авторов элективов, кластера, для просмотра зависимости количества экспертов, освещающих данную тематику в виде элективов от темы.

Рассмотрим результаты запроса для кластеризации TF-IDF+K-means для типовых запросов «Физическая культура» и «Уголовное право» (Рисунки 12, 13).

Для выполнения метода передается строка запроса, по окончании работы метода выводятся название кластера, а также ближайшие к запросу элективы в кластере, представленные расстоянием до запроса, названием электива и сокращенным текстом электива.

```
Метод: TF-IDF + k-means
Запрос: Физическая культура
Кластер: Физическая культура и здоровье
Расстояние: 0.875 Название электива: Аэробные направления современного фитнеса Электив: аэробные направления со
временного фитнеса задачи формирование студентов мотивационно ценностного отношения физической культуре установ
ки здоровый стиль жизни физическое самосовершенствование самовоспитание потребности регулярных занятиях физичес
кими

Расстояние: 0.877 Название электива: Оздоровительные виды гимнастики Электив: оздоровительные виды гимнастики з
адачи формирование студентов мотивационно ценностного отношения физической культуре установки здоровый стиль жи
зни физическое самосовершенствование самовоспитание потребности регулярных занятиях физическими упражнения

Расстояние: 1.177 Название электива: Школа современного тренера: этап начальной подготовки ЗФО Электив: школа с
овременного тренера этап начальной подготовки зфо результате освоения дисциплины школа современного тренера тре
нировочные этапы подготовки бакалавр офо зфо должен обладать общепрофессиональными опк профессиональными пко ко
мпетенциями. опк спос

Расстояние: 1.199 Название электива: Игры народов России Электив: игры народов россии цель освоения курса форми
ровать физическую культуру личности способность направленному использованию разнообразных игр народов россии со
хранения укрепления здоровья психофизической подготовки самоподготовки будущей жизни профессио
```

Рис. 12. Пример работы с запросом «Физическая культура»

```
Метод: TF-IDF + k-means
Запрос: Уголовное право
Кластер: Юриспруденция
Расстояние: 0.842 Название электива: Тюрьмоведение Электив: тюрьмоведение предмет тюрьмоведение входит професси
ональный цикл дисциплины выбору. курсе тюрьмоведения студенты изучают ключевые вопросы уголовно исполнительного
права. являясь одной отраслей называемого криминального цикла уголовно исполнительное п

Расстояние: 0.985 Название электива: Стратегии защиты частных прав Электив: стратегии защиты частных прав знать
законодательство формах способах защиты нарушенных частных прав теорию правоотношения вещного обязательственног
о права деликты кондикционные обязательства способы защиты нарушенных вещных обязательственных корпорат

Расстояние: 0.987 Название электива: Конкурентное право Электив: конкурентное право целью освоения муп конкурен
тное право является приобретение бакалавром набора компетенций включающих знание понимание навыки области конку
рентного права также способности творческому самостоятельному осмыслению практическому примен

Расстояние: 1.044 Название электива: Конкурентное право Электив: конкурентное право целью освоения дисциплины к
онкурентное право является приобретение магистрантом набора компетенций включающих знание понимание навыки обла
сти конкурентного права также способности творческому самостоятельному осмыслению практическо
```

Рис.13. Пример работы с запросом «Уголовное право»

На изображениях видно, что предлагаемые по элективы имеют малое расстояние до запроса, а их темы совпадают. В таблице 17 представлены характеристики полученных кластеров.

Таблица 17

Характеристики кластеров

Название кластера	Количество элементов	Количество авторов	Ключевые слова
Психология и проектный менеджмент	88	22	Организация, управление, бизнес, экономика,
Изучение языков	65	13	Язык, общение, речь, устный
Химия и экология	61	16	Среда, охрана, человек, окружающий
Науки о культуре	61	18	Культура, искусство, рок, мышление
Математические дисциплины	36	9	Математика, решение, теория, анализ
Юриспруденция	35	15	Право, защита, регулирование, организация
История и философия	33	14	Политический, история, регион, мир

Продолжение таблицы 17

Физическая культура и здоровье	24	13	Спорт, здоровье, физкультурный, организм
Науки о данных	17	7	Данные, анализ, обработка, программирование
Бухгалтерский учет	4	4	Учет, бухгалтерия, отчетность, предпринимательство

Исходя из результатов полученной кластеризации, было решено реализовать подход TF-IDF как один из вариантов поиска похожего электива.

ГЛАВА 4. ОПИСАНИЕ СЕРВИСА И ЕГО ЭЛЕМЕНТОВ

Разработка сервиса для прогноза уникальности электива была реализована на языке Python в виде веб-сервиса. Пользовательский интерфейс написан на html и JavaScript, открывается в браузере, программно-аппаратная часть сервиса написана на Python 3.7. Описание методов и функционала сервиса для поиска элективных курсов и анализа их содержания с помощью методов исследования текстов на естественном языке приведено в таблице 18.

Таблица 18

Методы сервиса

Разработанные методы	Функционал
Сервис	
1. def create_config()	Метод создания конфигурации и привязки к номеру порта 446
2. async def html_form()	Асинхронный метод запроса к форме
3. async def update_courses_route()	Асинхронный метод запроса обновления курсов
4. async def check_courses()	Асинхронный метод запроса проверки курсов
5. async def update_cluster_route()	Асинхронный метод запроса обновления кластера
6. async def predict_result_route()	Асинхронный метод запроса предсказания результата
Парсер	
1. def requirement_clrm()	Метод получения всех данных для курса по аудитории
2. def requirement_prov()	Метод получения всех данных для курса по количеству
3. def requirement_sifp()	Метод получения всех данных для курса по интервалам
4. def requirement_tchr()	Метод получения всех данных для курса по преподавателям
5. def requirement_prep()	Метод получения всех данных для курса по времени подготовки

6. def get_course_params()	Метод получения параметров для полей таблицы курсы

Продолжение таблицы 18

7. async def create_hbase_table()	Асинхронный метод создание таблицы в Hbase
8. async def send_to_hbase()	Метод соединения сервера с БД Hbase
9. def store_courses()	Метод сохранения курсов в БД
10. def get_token()	Метод получения токена
Кластер	
1. def lemma()	Метод предобработки текста (нормализация, удаление стоп-слов, стемминг)
2. def clear_courses()	Метод очистки от регулярных выражений
3. def update_cluster()	Метод векторизации и кластеризации текста

4.1. РАЗРАБОТКА ПРОТОТИПА СЕРВИСА

Разработка сервиса для прогноза уникальности электива была реализована на языке Python в виде веб-сервиса. Пользовательский интерфейс написан на html и JavaScript, с использованием CSS. Программно-аппаратная часть сервиса (back-end) на Python 3.7.

При разработке по типу клиент-сервер, где клиентом является браузер пользователя, а сервером – веб-сервер. Данные хранятся на сервере, обмен информацией происходит по сети.

Для хранения данных использована СУБД PostgreSQL и библиотека psycopg2 для связи веб-сервиса с СУБД.

Для вывода сводной информации по курсам и областям знаний для построения диаграмм была использована библиотека Infographic, которая использует технологии HTML5 и поддерживается всеми современными браузерами, включая Microsoft Edge, Google Chrome, Mozilla Firefox, Opera, Safari.

Данная библиотека зарекомендовала себя как быстрый и удобный инструмент для построения диаграмм на чистом JavaScript без использования дополнительных библиотек (см. Листинг 5).

```
    <script src="/js/infographic.min.js"
type="text/javascript"></script>
```

Листинг 5. Пример подключения библиотеки Infographic

4.2. ФУНКЦИОНАЛ СЕРВИСА

Разработанный сервис ориентирован на два типа пользователей: студент и преподаватель. Для пользователя Преподаватель, предусмотрен функционал добавления и обновления элективных курсов.

Для пользователя сервиса доступен поиск элективных курсов по названию, по тематике, по автору. Для отображения предусмотрена сводная таблица найденных элективов по областям знаний и диаграммы. Поиск реализован двумя разными подходами.

Первый подход предполагает поиск похожего электива среди базы элективов, разбитых на 25 тематических кластеров. Данный вариант работает с названиями, полными и краткими описаниями элективов. Подход не поддерживает для поиска запросы, иностранном языке. Второй подход предлагает поиск похожих элективов, разбитых на 10 тематических кластеров. В подходе для работы используются названия, описания и образовательный результат элективов. Второй подход поддерживает запросы с использованием слов и терминов, написанных на иностранном языке.

Для обоих подходов доступна диаграмма и таблица распределения найденных элективов по областям знаний. Так же, для результатов поиска доступны сортировка и фильтрация элективов по названию, автору, числу обучающихся и длительностью электива. Для получения подробной информации по найденным элективам предусмотрена возможность перехода к странице электива на образовательной платформе Modeus.

В случае возникновения у пользователей вопросов предусмотрена форма обратной связи для обращения к разработчикам сервиса.

4.3. ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ

При запуске сервиса открывается главная страница «О сервисе». На данной странице отображаются названия разделов вертикального меню. Список разделов нужного блока меню выбирается нажатием на ссылку с соответствующим названием блока в верхней левой части страницы.

На главной странице можно прочитать информацию о сервисе, выбрать пользователя, перейти на форму обратной связи с разработчиками, по нажатию кнопки «Выбрать учебный курс» приступить к поиску электива. (Рисунок 14).

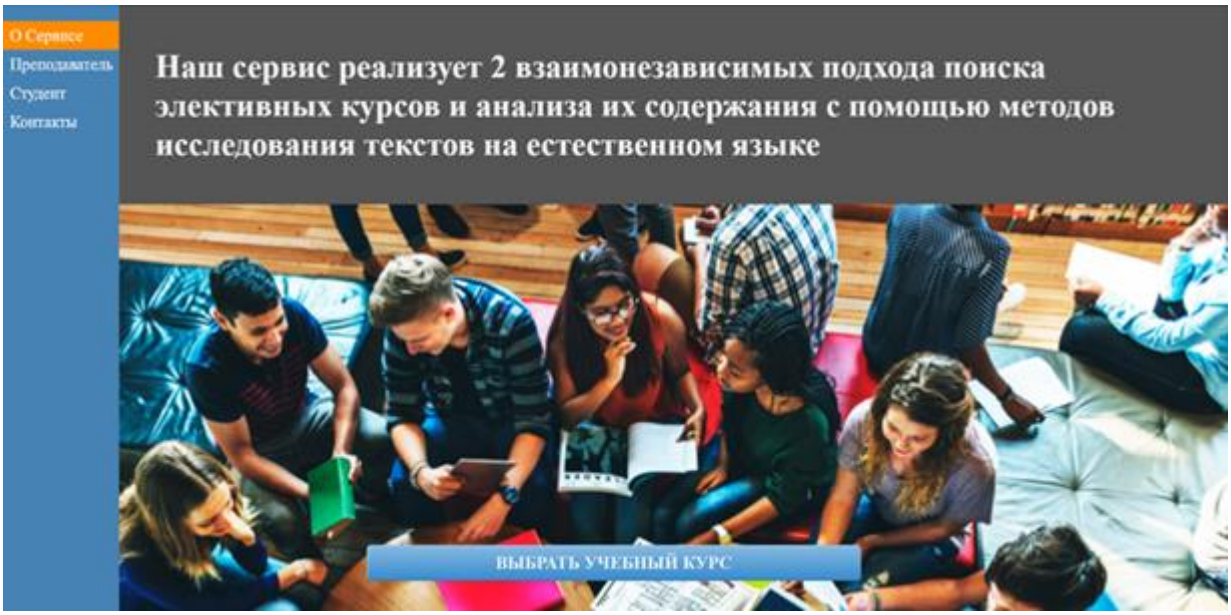

Рис. 14. Главная страница сервиса.

Наличие или отсутствие того или иного пункта в вертикальном меню определяется правами пользователя, установленными Администратором сервиса.

4.3.1. СТРАНИЦА ПОЛЬЗОВАТЕЛЯ

При выборе раздела вертикального выпадающего меню «Преподаватель» или «Студент» осуществляется переход на страницу пользователя с правами которого можно осуществить установленные действия. При переходе на страницу пользователя «Преподаватель» отображается форма с полем ввода поиска учебных курсов.

Для поиска электива необходимо в поле ввода с названием «Текст» ввести свой поисковый запрос и нажать на кнопку с логотипом «Лупа», располагающуюся справа от поля ввода (Рисунок 15).

Рис. 15. Поле ввода поискового запроса

На странице пользователя представлены 2 подхода поиска учебных курсов. По завершении поиска, для каждого подхода выводится общее количество найденных подходящих элективов, информация об элективах выводится в отдельную для каждого подхода таблицу, отображение учебных курсов по областям знаний - на диаграмму слева от таблицы. Примеры поиска с использованием двух подходов на примере запроса «Разработка web сайтов» представлены на рисунках 16,17.

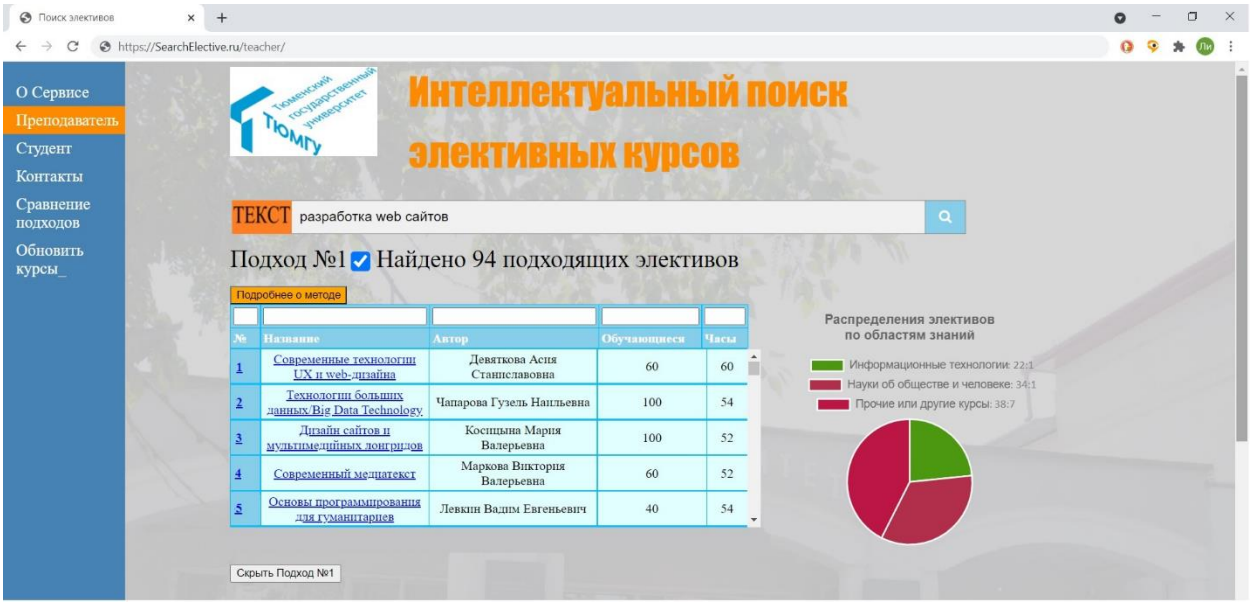

Рис. 16. Результаты поиска по первому подходу

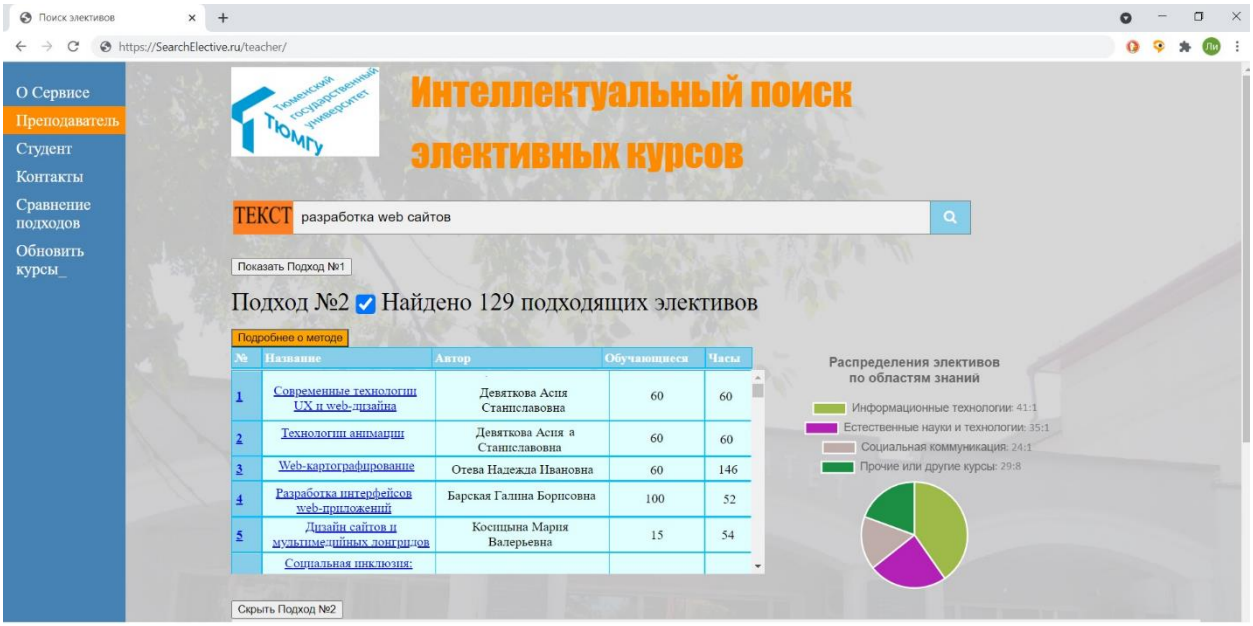

Рис.17. Результаты поиска по второму подходу

Для просмотра краткой справки по реализации каждого подхода над таблицей расположена кнопка «Подробнее о методе». При ее нажатии над таблицей отображается текст с описанием работы каждого подхода.

Чтобы скрыть отображение результатов поиска того или иного подхода под таблицей найденных элективов предусмотрена кнопка «Скрыть/показать подход» Для просмотра всего содержимого таблицы реализована полоса прокрутки. Так же к результатам отдельно по каждому подходу справа от соответствующей таблицы расположена круговая диаграмма. Диаграмма показывает долю областей знаний, к которым принадлежат найденные элективы.

Для пользователя «Преподаватель» доступна кнопка «Обновить курсы», при нажатии на которую происходит обновление списка курсов и запуск алгоритмов кластеризации (обновление кластеров) в соответствии с подходами.

СТРАНИЦА СРАВНЕНИЯ ЭЛЕКТИВОВ

При переходе на раздел вертикального меню «Сравнение подходов» отображается поле ввода с названием «Текст» для введения поискового запроса и нажать на кнопку с логотипом «Лупа». Ниже располагаются две кнопки с отображением подходов. Под кнопками две вкладки: на первой прописано количество элективов (Рисунок 18), на второй таблицы с распределением элективов по областям знаний (Рисунок 19).

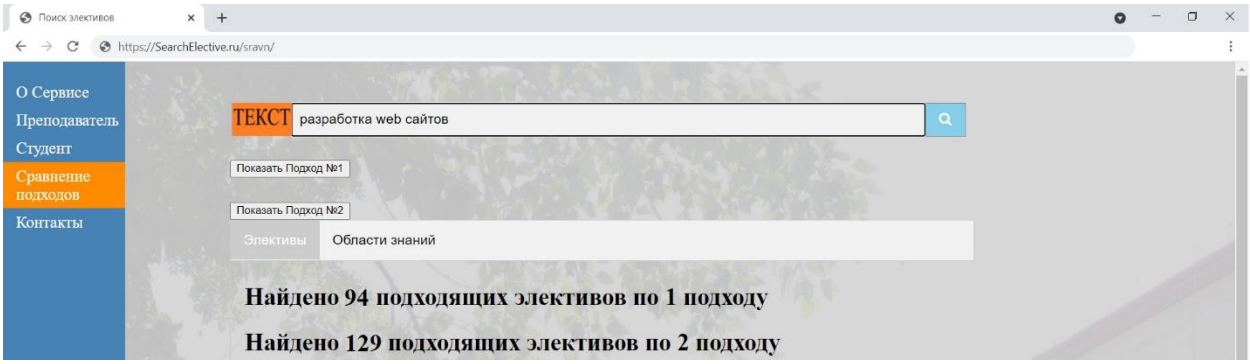

Рис.18. Сравнение количества элективов

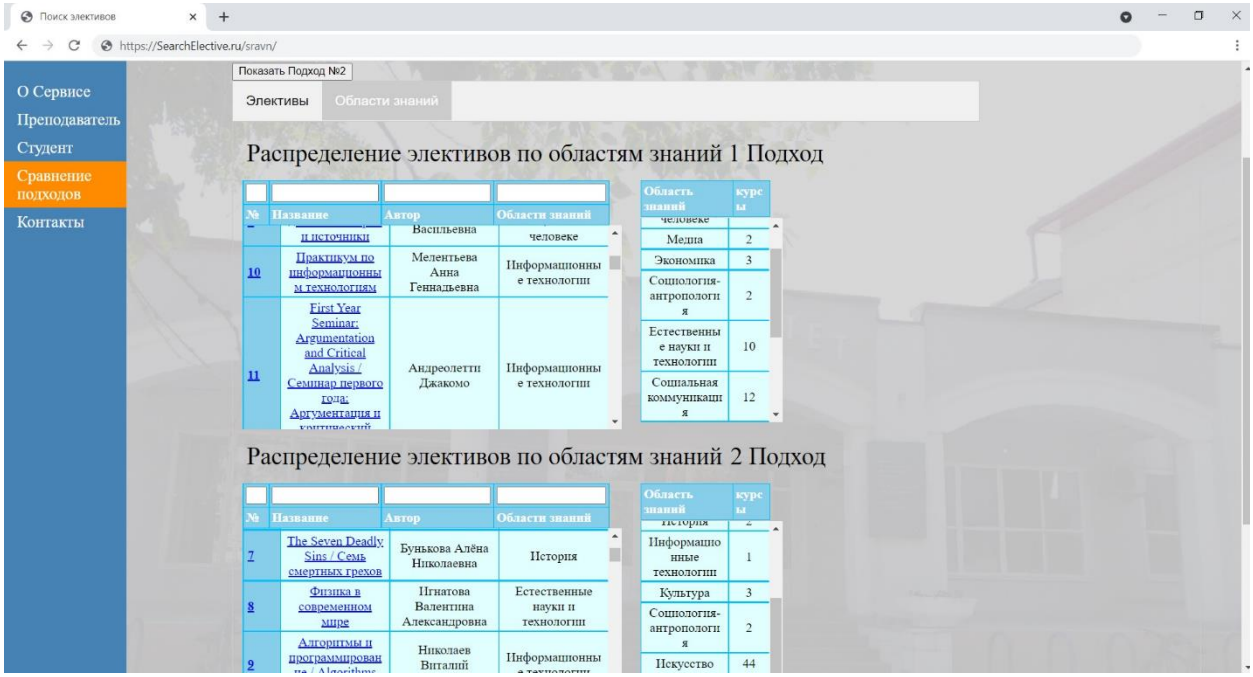

Рис.19. Распределение элективов по областям знаний

4.3.2. СТРАНИЦА КОНТАКТОВ

При переходе на раздел вертикального меню «Контакты» отображается форма ввода контактной информации пользователя.

Форма ввода контактной информации представлена полями:

- Поле «Имя» - вводится имя пользователя
- Поле «Фамилия» - вводится фамилия пользователя
- Поле «Страна» - нужно выбрать свою страну из выпадающего списка
- Поле «Ваше сообщение» - ввести необходимый текст

Для отправки сообщения необходимо нажать на ссылку внизу с названием «Отправить».

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы были изучены технологии разработки поиска подобных текстов по заданному запросу. В изученных статьях были предложены наиболее популярные методы кластеризации успешные их реализации и внедрения.

В результате сравнения и тестирования хорошие результаты показал алгоритм Kmeans. Обработал большой объем данных, равномерно распределил элективы по кластерам. Данный алгоритм кластеризации был использован в двух подходах.

Для реализации сервиса поиска элективных курсов и анализа их содержания с помощью методов исследования текстов на естественном языке были использованы 2 подхода:

- Поход №1: С помощью алгоритма кластеризации Kmeans весь набор текстовых данных разделяется на кластеры. При реализации классификатора каждому классу присваивается метки, и когда появляется новый электив, он уже классифицируется в свой класс. Запрос: выделяются признаки из текста пользовательского запроса, преобразуются в числовой вектор, и с помощью классификатора, основанного на обучающем наборе данных, определяется к какому классу относится введенный текст.
- Подход №2: Был произведен анализ и сокращение данных для успешной кластеризации. Для каждого кластера были получены тема и набор ключевых слов, описывающих его. При выполнении поискового запроса определяется его тема, кластер и ближайшие к нему элективы на основе его числового представления запроса с помощью TF-IDF меры и сравнения представления с элементами кластера и их элементов.

По результатам выпускной квалификационной работы был спроектирован и разработан прототип «Интеллектуальный поиск элективных курсов». Сервис предлагает преподавателям и студентам возможность

поиска похожих элективов двумя разными методами с возможностью сортировки и фильтрации. С его помощью студенты и преподаватели могут производить поиск похожих элективных курсов в двух реализациях.

СПИСОК ЛИТЕРАТУРЫ

1. Богдан А. В., Захарова И. Г. Разработка методов хранения и поиска информации в неструктурированных текстовых документах // elibrary.ru: [сайт]. 2020 URL: https://www.elibrary.ru/item.asp?id=45635462/ (дата обращения: 27.03.2021).
2. Воробьева М. С., Дубаков А. А., Разработка приложения для определения тематики текста с использованием алгоритмов кластеризации // elibrary.ru: [сайт]. 2020 URL: https://www.elibrary.ru/item.asp?id=41380880 (дата обращения: 27.03.2021).
3. Воробьев А.М., Боганюк Ю.В. Рекомендательная система для проектирования образовательных программ в условиях индивидуальных образовательных траекторий // elibrary.ru: [сайт]. 2020 URL: https://www.elibrary.ru/item.asp?id=44034423 (дата обращения: 27.03.2021).
4. Пархоменко П. А., Григорьев А.А., Астраханцев Н.А. Обзор и экспериментальное сравнение методов кластеризации текстов. Труды ИСП РАН, том 29, вып. 2, 2017 стр. 161-200 // URL: https://cyberleninka.ru/article/n/obzor-i-eksperimentalnoe-sravnenie-metodov-klasterizatsii-tekstov/viewer (дата обращения: 29.03.2021).
5. Andrews N. O., Fox E. A. Recent developments in document clustering: Tech. Rep.: Technical report, Computer Science, Virginia Tech, 2007. // URL: https://eprints.cs.vt.edu/archive/00001000/01/docclust.pdf. (дата обращения: 29.03.2021).
6. Sathiyakumari K., Manimekalai G., Preamsudha V. A survey on various approaches in document clustering 2011 // URL: https://www.ijcotjournal.org/volume1/issue-3/IJCOT-V1I3P303.pdf. (дата обращения: 29.03.2021).

7. Ершов К.С., Романова Т.Н. Анализ и классификация алгоритмов кластеризации. 2019// URL: https://cyberleninka.ru/article/n/analiz-i-klassifikatsiya-algoritmov-klasterizatsii (дата обращения: 29.03.2021).
8. Новости ТюмГУ. Элективы, или Что нужно знать о новой модели образования ТюмГУ? // utmn.ru [сайт] 2020 URL: https://news.utmn.ru/news/obrazovanie/554128/ (дата обращения: 30.03.2021).
9. Карпов И.А., Гославский А.И. Применение алгоритма BIRCH к кластеризации текстов. ИПС РАН Переславль-Залеский. 2012. URL: https://www.slideserve.com/faustus-marlas/birch (дата обращения: 30.03.2021).
10. Пархоменко П. А. Обзор и экспериментальное сравнение методов кластеризации текстов. Институт системного программирования РАН. Москва. 2017. // URL: https://www.ispras.ru/proceedings/docs/2017/29/2/isp_29_2017_2_161.pdf (дата обращения: 30.03.2021).
11. Немного про word2vec: полезная теория // nlpx.net [сайт] 2015 URL: http://nlpx.net/archives/179 (Дата обращения 11.04.2021)
12. Бугаев А.А., Гуренко А.В., Нечепоренко М.А. Методы обработки естественного языка // Научно-издательский центр «Мир науки» 2019. URL: http://science-peace.ru/files/INITMP_2019.pdf (Дата обращения 11.04.2021)
13. Big Data от А до Я. Часть 4: Hbase // Habr.com [сайт]. 2016 URL: https://habr.com/ru/company/dca/blog/280700/ (Дата обращения - 15.04.2021).
14. PostgreSQL — объектно-реляционная система управления базами данных // web-creator.ru [сайт]. 2020 URL: https://web-creator.ru/articles/postgresql (Дата обращения - 15.04.2021).
15. Кафтанников И. Л., Парасич А. В. Особенности применения деревьев решений в задачах классификации // Вестник Южно-Уральского

государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. – 2015. – Т. 15. – №. 3.

16. Баев И. О. Использование метода опорных векторов в задачах классификации // Международный журнал информационных технологий и энергоэффективности. – 2017. – Т. 2. – №. 2.
17. Построение систем машинного обучения на языке Python. 2-е издание. / пер. с англ. Слинкин А.А. - М.: ДМК Пресс, 2016.
18. Bhavesh Patel, Jyotindra Dharwa. Selection of Optimal Classification Algorithms in Education Data Mining, 2017 // URL: http://www.imperialjournals.com/index.php/IJIR/article/view/3400 (Дата обращения: 19.04.2021).
19. Основы Data Science и Big data. Python и наука о данных М., 2017 //URL: https://vk.com/wall54530371_140946 (Дата обращения: 20.05.2021).
20. Драйвер Python SQL – psycopg // psycopg.org [сайт] 2017 URL: http://initd.org/psycopg/docs/ (Дата обращения: 20.05.2021).
21. Грудева Е. В. Избыточность языка и избыточность текста: некоторые размышления // Acta Linguistica Petropolitana. Труды института лингвистических исследований РАН. Том VI, часть 2. СПб., 2010.

Приложение 1

Конфигурация сервера

```
import asyncio
from threading import Thread
from hypercorn import Config
from hypercorn.asyncio import serve
from starlette.applications import Starlette
from starlette.responses import HTMLResponse, JSONResponse
from starlette.routing import Route
from courses_parse import update_courses, check_working
from cluster import update_cluster, predict_result
from db_connection import DBConnection
from queries import GET_COURSES_BY_ID
def create_config():
  config = Config()
  config.bind = ['0.0.0.0:446']
  return config
async def html_form(request):
  html_template = open('templates/form.html', 'r+',
encoding='utf-8').read()
  return HTMLResponse(html_template)
async def update_courses_route(request):
  Thread(target=update_courses).start()
  return JSONResponse({'status': 'working'})
async def check_courses(request):
  return JSONResponse({'status': 'working' if check_working()
else 'waiting'})
async def update_cluster_route(request):
  Thread(target=update_cluster).start()
  return JSONResponse({'status': 'working'})
async def predict_result_route(request):
  json_data = await request.json()
  data = None
```

```
  if isinstance(json_data['search'], str):
    courses = predict_result(json_data['search'])
    data        =        (DBConnection().query(GET_COURSES_BY_ID,
{'course_ids': courses}))
  return JSONResponse({'data': data})
routes = [
  Route('/update_clusters', update_cluster_route),
  Route('/predict_result',                  predict_result_route,
methods=['POST']),
  Route('/update_courses', update_courses_route),
  Route('/check_courses', check_courses),
  Route('/{path:path}', html_form)
]
app = Starlette(routes=routes)
if __name__ == '__main__':
  asyncio.run(serve(app, create_config()))
```

Приложение **2**

Парсер текстов элективов

```
#асинхронный ввод/вывод, обеспечивают
высокопроизводительные сетевые и веб-серверы,
библиотеки подключения к базам данных, распределенные
очереди задач и т.д
import asyncio
#создает потоковый параллельный интерфейс
from threading import Thread
import requests
import json
from aiohappybase import Connection
from selenium import webdriver
import chromedriver_autoinstaller # Adds chromedriver binary to
path
from db_connection import DBConnection
from constants import *
from queries import *
chromedriver_autoinstaller.install()
# Создаем подключение к базе
CONNECTION = DBConnection()
def requirement_clrm(lesson_id, requirement):
  requirement_params = {
    'lesson_id': lesson_id,
    'requirement_id': requirement['id'],
    'description': requirement['requirement']['description'],
    'classroom_type_code':
requirement['requirement']['classroomTypeCode'],
    'min_capacity': requirement['requirement']['minCapacity'],
    'rooms_quantity':
requirement['requirement']['roomsQuantity'],
```

```
    'students_group_size':
requirement['requirement']['studentsGroupSize']
  }
  CONNECTION.execute(CLRM_ADD, requirement_params)
def requirement_prov(lesson_id, requirement):
  requirement_params = {
    'lesson_id': lesson_id,
    'requirement_id': requirement['id'],
    'description': requirement['requirement']['description'],
    'category_code': requirement['requirement']['categoryCode'],
    'at_type': requirement['requirement']['type'],
    'provision_quantity':
requirement['requirement']['provisionQuantity'],
    'students_group_size':
requirement['requirement']['studentsGroupSize']
  }
  CONNECTION.execute(PROV_ADD, requirement_params)
def requirement_sifp(lesson_id, requirement):
  requirement_params = {
    'lesson_id': lesson_id,
    'requirement_id': requirement['id'],
    'description': requirement['requirement']['description'],
    'lesson_relation_type':
requirement['requirement']['lessonRelationType'],
    'lesson_relation_type_code':
requirement['requirement']['lessonRelationTypeCode'],
    'spec_lesson_id':
requirement['requirement']['specLessonId'],
    'prev_lesson_type':
requirement['requirement']['prevLessonType'],
    'min_interval_in_days':
requirement['requirement']['minIntervalInDays'],
    'max_interval_in_days':
requirement['requirement']['maxIntervalInDays']
  }
```

```
  CONNECTION.execute(SIFP_ADD, requirement_params)
def requirement_tchr(lesson_id, requirement):
  requirement_params = {
    'lesson_id': lesson_id,
    'requirement_id': requirement['id'],
    'description': requirement['requirement']['description'],
    'role_name': requirement['requirement']['roleName'],
    'teachers_number':
requirement['requirement']['teachersNumber'],
    'students_group_size':
requirement['requirement']['studentsGroupSize']
  }
  CONNECTION.execute(TCHR_ADD, requirement_params)
def requirement_prep(lesson_id, requirement):
  requirement_params = {
    'lesson_id': lesson_id,
    'requirement_id': requirement['id'],
    'description': requirement['requirement']['description'],
    'preparation_time':
requirement['requirement']['preparationTime'],
    'preparation_type_code':
requirement['requirement']['preparationTypeCode']
  }
  CONNECTION.execute(PREP_ADD, requirement_params)

REQUIREMENT_PARAMS = {
  'CLRM': requirement_clrm,
  'PROV': requirement_prov,
  'SIFP': requirement_sifp,
  'TCHR': requirement_tchr,
  'PREP': requirement_prep
}
def get_course_params(course):
  return {
    'id': course['id'],
```

```
    'catalog_code': course['catalogCode'],
    'catalog_name': course['name'],
    'catalog_short_name': course['shortName'],
    'lessons_count': course['lessonsCount'],
    'lessons_hours': course['lessonsHours'],
    'lesson_hours_units_code': course['lessonHoursUnitsCode'],
    'full_description': course['fullDescription'],
    'description': course['description'],
    'status': course['status'],
    'date_created': course['dateCreated'],
    'date_published': course['datePublished'],
    'date_archived': course['dateArchived'],
    'date_agreement': course['dateAgreement'],
    'annotation': course['annotation'],
    'throughput': course['throughput'],
    'educational_result': course['educationalResult'],
    'knowledge_id': course['fieldOfKnowledge']['id'],
    'knowledge_parent_field_id':
course['fieldOfKnowledge']['parentFieldId'],
    'knowledge_title': course['fieldOfKnowledge']['title'],
    'curator_person_id': course['curator']['person']['id'],
    'author_person_ids': ', '.join(author['id'] for author in
course['courseUnitAuthors']),
    'requirement_ids': ', '.join(author['id'] for author in
course['requiredUnits'])
  }
def get_lesson_type_params(course_id, lesson_type):
  return {
    'course_id': course_id,
    'type_code': lesson_type['lessonTypeCode'],
    'type_label': lesson_type['lessonTypeLabel'],
    'count': lesson_type['lessonsCount'],
    'hours': lesson_type['lessonsHours'],
    'hours_units_code': lesson_type['lessonsHoursUnitsCode']
  }
```

```
def get_lesson_params(course_id, lesson):
  return {
    'course_id': course_id,
    'id': lesson['id'],
    'name': lesson['name'],
    'type_code': lesson['lessonTypeCode'],
    'form_type_code': lesson['lessonFormTypeCode'],
    'form_type': lesson['lessonFormType'],
    'description': lesson['description'],
    'duration': lesson['duration'],
    'duration_units_code': lesson['durationUnitsCode'],
    'throughput': lesson['throughput']
  }
def get_person_params(course):
  return {
    'id': course['curator']['person']['id'],
    'course_id': course['id'],
    'name': course['curator']['person']['name'],
    'surname': course['curator']['person']['surname'],
    'middle_name': course['curator']['person']['middleName'],
    'birth_date': course['curator']['person']['birthDate'],
    'gender': course['curator']['person']['gender'],
    'description': course['curator']['person']['description'],
    'phone':
course['curator']['person']['contactInformation']['phone'],
    'email':
course['curator']['person']['contactInformation']['email'],
    'other_contacts':
course['curator']['person']['contactInformation']['otherContacts
']
  }
def get_group_params(course_id, group):
  return {
    'id': group['id'],
    'course_id': course_id,
```

```
    'name': group['name']
  }
def split_list(lst):
  count = len(lst) // 4 + 1
  for i in range(0, len(lst), count):
    yield lst[i:i + count]
def run_coro(coro):
  asyncio.run(coro)
def create_task(coro):
  while True:
    try:
      run_coro(coro)
      break
    except Exception as e:
      pass
THREADS_WORKING = []
WORKING_STATUS = False
def check_working():
  return WORKING_STATUS or (THREADS_WORKING and
any(thread.is_alive() for thread in THREADS_WORKING))
async def create_hbase_table():
  async with Connection(HBASE_HOST) as hbase:
    if not HBASE_TABLE.encode() in await hbase.tables():
      await hbase.create_table(HBASE_TABLE, {
        'family': {'max_versions': 5}
      })
async def send_to_hbase(course_id, description):
  async with Connection(HBASE_HOST) as hbase:
    table = hbase.table(HBASE_TABLE)
    await table.put(course_id.encode(), {b'family:description':
description.encode()})
def store_courses(token):
  # Создаем хэдеры с нашим токеном
  header = {
    'Authorization': 'Bearer ' + token
```

```
  }
  # Создаем сессию
  session = requests.Session()
  # Задаем номер страницы
  courses_page = 1
  # Делаем запрос и получаем ответ со списком курсов
  response = session.get(COURSES_LINK.format(courses_page),
headers=header)
  # Парсим JSON и достаем из него все курсы
  courses = [chunk for chunk in
split_list(json.loads(response.text)['data'])]
  # Создаем таблицу в hbase
  if HBASE_ENABLE:
    create_task(create_hbase_table())
  for course_chunk in courses:
    thread = Thread(target=thread_store, args=(course_chunk,
token))
    THREADS_WORKING.append(thread)
    thread.start()
def thread_store(courses, token):
  # Создаем хэдеры с нашим токеном
  header = {
    'Authorization': 'Bearer ' + token
  }
  # Создаем сессию
  session = requests.Session()
  # Записываем курсы и уроки в базу
  for course in courses:
    # Для курса получаем его параметры
    course_params = get_course_params(course)
    # Записываем курс в базу
    CONNECTION.execute(COURSE_ADD, params=course_params)
    # Записываем описание в hbase
    if HBASE_ENABLE:
```

```
      create_task(send_to_hbase(course['id'],
course['fullDescription']))
    # Получаем параметры для куратора
    person_params = get_person_params(course)
    # Добавляем его в базу
    CONNECTION.execute(PERSON_ADD, params=person_params)
    for group in course['curator']['groups']:
      # Получаем параметры для группы
      group_params = get_group_params(course['id'],
group['group'])
      # Добавляем ее в базу
      CONNECTION.execute(GROUP_ADD, params=group_params)
    # Получаем уроки для курса
    response = session.get(LESSONS_LINK.format(course['id']),
headers=header)
    lesson_types = json.loads(response.text)['rows']
    for lesson_type in lesson_types:
      # Для типа урока получаем его параметры
      lesson_type_params = get_lesson_type_params(course['id'],
lesson_type)
      # Записываем тип урока в курсе в базу
      CONNECTION.execute(LESSON_TYPE_ADD,
params=lesson_type_params)
      lessons = lesson_type['fields']
      for lesson in lessons:
        if lesson:
          # Получаем параметры урока
          lesson_params = get_lesson_params(course['id'],
lesson)
          # Записываем урок в базу
          CONNECTION.execute(LESSON_ADD, params=lesson_params)
          reqs = lesson['requirements']
          for req in reqs:
            # Разбираем реквайрменты и записываем в базу
```

```
REQUIREMENT_PARAMS[req['requirement']['@type']](lesson['id'],
req)
def get_token():
  # Создаем настройки для Хрома
  options = webdriver.ChromeOptions()
  # Добавляем опцию не включать видимость окна
  options.add_argument('headless')
  # Запускаем хром
  driver = webdriver.Chrome(options=options)
  # Открываем Модеус
  driver.get(MODEUS_LINK)
  # Ждем, пока загрузится сайт с авторизацией
  while 'fs.utmn.ru' not in driver.current_url:
    pass
  # Заполняем данные для авторизации
  username = driver.find_element_by_id('userNameInput')
  password = driver.find_element_by_id('passwordInput')
  username.send_keys(MODEUS_LOGIN)
  password.send_keys(MODEUS_PASSWORD)
  # Нажимаем на кнопку логина
  driver.find_element_by_id('submitButton').click()
  # Ждем пока загрузится Модеус
  while 'utmn.modeus.org/schedule-calendar' not in
driver.current_url:
    pass
  # Получаем токен из объекта sessionStorage браузера
  session_storage = driver.execute_script('return
sessionStorage')
  for item in session_storage.keys():
    if 'oidc.user' in item:
      tokens = json.loads(session_storage[item])
      token = tokens['id_token']
      break
  return token
```

```
def update_courses():
  if check_working():
    return
  THREADS_WORKING.clear()
  global WORKING_STATUS
  WORKING_STATUS = True
  token = get_token()
  store_courses(token)
  WORKING_STATUS = False
if __name__ == '__main__':
  update_courses()
```

Приложение 3

Модуль кластеризации первого подхода

```
import re
import string
import nltk
import pymorphy2
from bs4 import BeautifulSoup
from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.naive_bayes import MultinomialNB, ComplementNB,
BernoulliNB
from db_connection import DBConnection
from queries import GET_COURSE_WITH_DESCRIPTION
nltk.download('stopwords')
nltk.download('punkt')
specs = {ord(i): None for i in string.digits +
string.punctuation + '\n\xa0«»\t—…' + '[\d]'}
russian_stopwords = stopwords.words('russian')
russian_pattern = re.compile('[а-яА-Я]')
all_words = ['дисциплин', 'провод', 'формат', 'открыт']
russian_stemmer = SnowballStemmer('russian')
morph = pymorphy2.MorphAnalyzer(result_type=None)
vectorizer = TfidfVectorizer()
all_courses = []
all_ids = []
groups = {}
# Model
clustering = None
n_samples = None
km = None
```

```
def lemma(text):
  # удаляет указанный набор символов из исходного текста:
  text = text.translate(specs)
  # Токенезация
  text_tokens = word_tokenize(text)
  # Для применения инструментов частотного анализа библиотеки NLTK необходимо список токенов преобразовать к классу Text, который входит в эту библиотеку:
  text_tokens = nltk.Text(text_tokens)
  # удалить стоп-слова из русского и английского текста
  filtered_words = [
    word for word in text_tokens
    if word not in russian_stopwords
  ]
  # выбираем только русские слова
  filtered_words = [w for w in filter(russian_pattern.match, filtered_words)]
  # Лемматизация
  russ = [morph.parse(word)[0][2] for word in filtered_words]
  # СТЕММЕР
  return ' '.join(russian_stemmer.stem(word) for word in russ if word not in all_words)
def clear_courses():
  courses = DBConnection().query(GET_COURSE_WITH_DESCRIPTION)
  all_courses.clear()
  all_ids.clear()
  for row in courses:
    # формируем строку для чистки от регулярных выражений
    text = row['catalog_name'] + ' ' + row['full_description'] + ' ' + row['description']
    text = BeautifulSoup(text, "lxml").get_text(separator=' ', strip=True).lower()
    processed = lemma(text)
    all_courses.append(processed)
    all_ids.append(row['id'])
```

```
def update_cluster(classificator='1'):
  clear_courses()
  global n_samples, km
  # Векторизация Библиотекой Tfidf
  n_samples = vectorizer.fit_transform(all_courses)
  # Пропагайшон кластеризация !!!!
  # На вход разряж матрица (X)
  global clustering
  clustering = KMeans(n_clusters=25,
random_state=1).fit(n_samples)
  for course_id, group_number in zip(all_ids, km):
    ids = groups.get(group_number)
    if ids:
      ids.append(course_id)
    else:
      groups[group_number] = [course_id]
  print('Updated first algorithm')
modes = {
  '1': MultinomialNB,
  '2': ComplementNB,
  '3': BernoulliNB
}
names = [
  'Преподавание немецкого языка',
  'Математика',
  'Основы права',
  'Дискурсивные практики инсотранных языков',
  'Экология',
  'Педагогика',
  'Финансовая грамотность',
  'Современная культура и искусство',
  'Программирование',
  'Экономика',
  'Русская культура',
  'История',
```

```
  'Базовый английский',
  'Социальная психология',
  'Юриспруденция',
  'Банковский менеджмент',
  'Бизнеспланирование',
  'Основы криминалистики',
  'Практические семинары',
  'Профессиональный английский',
  'Естественные науки',
  'Журналистика и основы социологии',
  'Основы политологии',
  'Рискменеджмент',
  'Основы цифровых технологий'
]
def predict_result(search, mode='1'):
  search = lemma(search.lower())
  predict_vector = vectorizer.transform([search])[0]
  if isinstance(clustering, KMeans):
    group = modes[mode]().fit(n_samples,
km).predict(predict_vector)[0]
    if group in groups:
      return names[group], groups[group]
  return []
if __name__ == '__main__':
  update_cluster()
```

Приложение **4**

Модуль кластеризации второго подхода

```
from datetime import datetime
import nltk
import copy
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
import pymorphy2
from sklearn.cluster import KMeans
import scipy as sp
from nltk.corpus import stopwords
from db_connection import DBConnection
from queries import GET_COURSE_WITH_DESCRIPTION
nltk.download('stopwords')
nltk.download('punkt')


def parseText(data): # Описания + Все теги и прочие интерет
плюшки удалены
  for i in range(0, len(data)):
    h = data[i]
    h = re.sub("(https/W*)|(htpp/W*)", " ", h)
    h = re.sub("(</*[^a-я]+>)|( )|(\ufeff)|(\xad)", " ", h)
    h = re.sub("([^A-Za-zA-Яа-я0-
9\.\!\@\"\#\$\%\^\&\*\(\)\№\;\:\?\<\>\,])", " ", h)
    h = h.strip()
    h = h.lower()
    yield h
def parseTextForAnalysis(data): # готовый для анализа(инилиха)
текст
  for h in data:
    h = re.sub("([^A-Za-zA-Яа-я0-
9\.])|([\!\@\#\$\%\^\&\*\(\)\,\:\;\№])", " ", h)
```

```
    h = re.sub("\W*[0-9]+\W+", " ", h)
    h = re.sub("\s+", " ", h)
    h = h.strip()
    h = h.lower()
    yield h
def deleteCopyes(data, name):
  tempData = copy.deepcopy(data)
  tempname = copy.deepcopy(name)
  copycount = 0
  for i in range(0, len(data)):
    if data.count(data[i]) > 1:
      tempData.remove(data[i])
      tempname.remove(name[i])
      copycount += 1
  print("Количество удаленных повторов: ", copycount)
  return tempData, tempname
def deleteLowSentence(data, name, count=8):
  tempData = copy.deepcopy(data)
  tempname = copy.deepcopy(name)
  deletecount = 0
  for i in range(0, len(data)):
    if len(data[i].split(".")) < count:
      tempData.remove(data[i])
      tempname.remove(name[i])
      deletecount += 1
  print("Количество удаленных курсов: ", deletecount)
  return tempData, tempname
stemmer = nltk.stem.SnowballStemmer('russian')
swr=stopwords.words('russian')
swe = stopwords.words('english')
morph = pymorphy2.MorphAnalyzer()
def dropStopWords(data): # список строк, список слов или строка
  temp = []
  if type(data) == str:
    temps = ""
```

```
    for i in data.split():
      if i not in swr and i not in swe:
        temps += i + " "
    return temps # строка
  if type(data) == list:
    if type(data[0]) == str:
      for i in data:
        temps = ""
        j = i.lower()
        j = j.split()
        for k in j:
          if k not in swr and k not in swe:
            temps += k + " "
        temp.append(temps) # список строк
      return temp
    else:
      if type(data[0]) == list:
        for i in data:
          templ = []
          for j in i:
            if j not in swr and j not in swe:
              templ.append(j)
          temp.append(templ)
        return temp # список слов
      print("Ошибка в удалении стоп-слов")
def normalForm(data): # список слов, список строк или строка
  morph = pymorphy2.MorphAnalyzer()
  string = ""
  ListS = []
  if type(data) == str:
    w = data.split()
    for i in w:
      # string+=morph.parse(i)[0].normal_form+" "
      string += stemmer.stem(morph.parse(i)[0].inflect({'sing',
'nomn'}).word) + " "
```

```
    return string # для строки
  else:
    if type(data) == list:
      if type(data[0]) == list:
        for i in data:
          temp = []
          for j in i:
            # temp.append(morph.parse(j)[0].normal_form)

temp.append(stemmer.stem(morph.parse(j)[0].inflect({'sing',
'nomn'}).word))
          ListS.append(temp)
      else:
        if type(data[0]) == str:
          for i in data:
            temp = ""
            for j in i.split():
              temp += morph.parse(j)[0].normal_form + " "
              # try:
              #
temp+=stemmer.stem(morph.parse(j)[0].inflect({'sing',
'nomn'}).word) +" "
              # except AttributeError:
              #
temp+=stemmer.stem(morph.parse(j)[0].normal_form) + " "
            ListS.append(temp)
      return ListS # для списка слов и списка строк
russian_stemmer = nltk.stem.SnowballStemmer('russian')
class StemmedTfidfVectorizer(TfidfVectorizer):
  def build_analyzer(self):
    analyzer = super(TfidfVectorizer, self).build_analyzer()
    return lambda doc: (russian_stemmer.stem(w) for w in
analyzer(doc))
class StemmedCountVectorizer(CountVectorizer):
  def __init__(self, stemmer):
```

```
    super(StemmedCountVectorizer, self).__init__()
    self.stemmer = stemmer
  def build_analyzer(self):
    analyzer = super(StemmedCountVectorizer,
self).build_analyzer()
    return lambda doc: (self.stemmer.stem(w) for w in
analyzer(doc))
def standardClassification(data, num_clusters=20):
  # список текстов, строковое сообщение
  global km, vectorized
  vectorized = vectorizer.fit_transform(data)
  km = KMeans(n_clusters=num_clusters, n_init=1, verbose=0,
random_state=3)
  km.fit(vectorized)
# Section: code
data_raw = []
data_ids = []
km = None
vectorized = None
vectorizer = StemmedTfidfVectorizer(min_df=10, max_df=0.5,
stop_words=swr, decode_error='ignore')
names = [
  'Физическая культура и здоровье',
  'Науки о данных',
  'Изучение языков',
  'История и философия',
  'Бухгалтерский учет',
  'Математические дисциплины',
  'Науки о культуре',
  'Экология',
  'Юриспруденция',
  'Психология и проектный менеджмент'
]
def predict_kmeans_first(message):
  if km is None or vectorized is None:
```

```
    return None
  new_post = message
  new_post_vec = vectorizer.transform([new_post])
  new_post_label = km.predict(new_post_vec)[0] # noqa
  # какой кластер?
  # print('predict_cluster_label: ', new_post_label)
  # номера сообщений из кластера
  index_name = names[new_post_label]
  similar_indices = (km.labels_ == new_post_label).nonzero()[0]
# noqa
  # print(similar_indices)
  similar = []
  for i in similar_indices:
    dist = sp.linalg.norm((new_post_vec -
vectorized[i]).toarray()) # noqa
    similar.append((dist, data_ids[i], i))
  similar = sorted(similar)
  return index_name, similar
def update_kmeans_clusters():
  global data_raw, data_ids
  courses = DBConnection().query(GET_COURSE_WITH_DESCRIPTION)
  data_raw.clear()
  data_ids.clear()


  for row in courses:
    data_raw.append(row['catalog_name'] + ' ' +
row['full_description'] + ' ' + row['description'])
    data_ids.append(row['id'])
  prepdata = []
  for i in parseText(data_raw):
    prepdata.append(i)
  prepdata, data_ids = deleteLowSentence(prepdata, data_ids)
  semilastdata = []
  for i in parseTextForAnalysis(prepdata):
    semilastdata.append(i)
```

```
semilastdata, data_ids = deleteCopyes(semilastdata, data_ids)
finaldata = dropStopWords(semilastdata)
print("Проверка, осталось курсов: ", len(finaldata))
start = datetime.now()
standardClassification(finaldata, 10)
print(datetime.now() - start)
```