МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)
01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)
Системное программирование и компьютерные технологии
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему Разработка новостного агрегатора с кластеризацией новостей на основе векторных моделей

Студент Р.Н. Муртазин _____
(И.О. Фамилия) (личная подпись)
Руководитель В.С. Климов _____
(И.О. Фамилия) (личная подпись)
Консультанты М.А. Четаева _____
(И.О. Фамилия) (личная подпись)

Допустить к защите
Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____20_____г.

Тольятти 2018

АННОТАЦИЯ

Название дипломной работы: «Разработка новостного агрегатора с кластеризацией новостей на основе векторных моделей». Дипломная работа посвящена вопросу применения кластеризации на основе векторных моделей в новостных агрегаторах.

Объект исследования – применение кластеризации для представления пользователю средств навигации по коллекции документов.

Предмет исследования – разбиение новостных документов на темы при помощи кластеризации и векторных моделей.

Цель работы – предоставление пользователю средств навигации по коллекции новостных документов путём разработки новостного агрегатора с кластеризацией новостей на основе векторных моделей.

Во введении даётся описание вопроса. В первой главе работы анализируется предметная область и существующие программные решения, выделяются их достоинства и недостатки. Во второй главе проводится анализ применения кластеризации новостей на основе векторных моделей для устранения недостатков и создания нового программного решения. В третьей главе описывается программная реализация нового решения, описывается архитектура программного решения, используемый язык программирования и используемые библиотеки. В заключении подводятся итоги и делается вывод.

В работе затрагиваются следующие вопросы: кластеризация документов для разбиения коллекции новостей на темы (кластеры), способы векторного представления документов с помощью векторных моделей и сравнение их эффективности, способы наименования кластеров документов для именования тем новостных статей.

Данная работа состоит из пояснительной записки на 49 страниц, включая 24 рисунков, списка 21 источников, в том числе 5 источников на иностранном языке и одного приложения.

ABSTRACT

The title of the bachelor's thesis is Design and implementation of news aggregator using news clustering based on vector space models.

The aim of the work was to provide effective news presentation for exploratory browsing by developing news aggregator using news clustering based on vector space models.

The object of the research was the application of clustering to provide effective news presentation for exploratory browsing.

The subject of the research was clustering news articles using vector space models to divide them into topic groups.

We started with the statement of the problem and then followed through with the possible solution. Firstly, we discussed popular news aggregators. We examined their advantages and disadvantages and analyzed how news clustering might solve problems of static topics approach being used in the discussed existing solutions. We then described k-means clustering algorithm and its k-means++ modification. We used different vector space models to obtain vector representations of news documents and apply k-means++. We reported the results of the experiments conducted to determine the most performant model for our data. Then two ways of cluster naming were compared. Finally, we designed and implemented a news aggregator that collected news documents from preset news sources, parsed them, clusterized and gave a name to each cluster.

The dynamic topics approach has confirmed that it is much more flexible compared to static topics approach. It can be concluded that the implemented news aggregator with dynamic topics approach using clustering and vector space models solved the inflexibility problem of existing news aggregators, but it is less accurate and reliable due to the nature of machine learning algorithms.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

С развитием интернета актуальным встал вопрос о навигации по большому количеству быстрорастущего контента. Решением этого вопроса занимается информационный поиск. Хотя данная область существует довольно долго, сильное развитие она получила совсем недавно за счёт набирающих популярность областей машинного обучения и больших данных. Машинное обучение – класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме.

В настоящее время существующие новостные агрегаторы для разделения новостей по темам либо используют ручной подход, либо ориентируются на то, к какой теме принадлежит новость в оригинальном источнике (что в свою очередь предъявляет требование к источникам новостей). В обоих случаях темы являются довольно широкими и заранее заданными. Поэтому перспективным с точки зрения исследования являются способы автоматического разделения коллекции новостей на узкие для данной коллекции темы, в этом заключается актуальность работы.

Объект исследования – применение кластеризации для представления пользователю средств навигации по коллекции документов.

Предмет исследования – разбиение новостных документов на темы при помощи кластеризации и векторных моделей.

Цель работы – предоставление пользователю средств навигации по коллекции новостных документов путём разработки новостного агрегатора с кластеризацией новостей на основе векторных моделей.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ ПРОГРАММНЫХ РЕШЕНИЙ

1.1 Анализ предметной области

Представление средств навигации по коллекции документов является задачей информационного поиска. Информационный поиск до некоторого времени оставался скромной научной и прикладной областью, в которой работало небольшое количество учёных. За последние годы именно развитие интернет-технологий являлось основным двигателем инноваций в сфере информационных технологий [1]. Оно и привело к сильному развитию области информационного поиска, так как интернет является огромным источником информации. Без возможности извлекать необходимую информацию из богатого контентом интернета, он не был бы так полезен. Современный информационный поиск – это миллионы пользователей, огромные объёмы данных, мощные вычислительные системы, изощрённые алгоритмы [1].

К информационному поиску относятся и такие задачи, как навигация пользователей по коллекции документов и фильтрация документов, а также дальнейшая обработка найденных документов. Для решения задачи поиска информации привлекаются методы машинного обучения, анализа мультимедийной информации, компьютерная лингвистика, геоинформационные сервисы, исследуется психология пользователей и их социальные связи, удобство интерфейсов и т.д [1]. Отдельно стоит выделить машинное обучение. Именно благодаря методам машинного обучения были созданы существующие популярные поисковые системы, такие как Google, Yandex, Bing и т.д. Также машинное обучение может применяться на производстве Например, для управления и диагностики сварки [2-4]. Схема информационного поиска представлена на рисунке 1.1.

Рисунок 1.1 – Схема информационного поиска

Раньше информационным поиском занимались лишь определённые лица (например, библиотекари). Сейчас же информационным поиска занимается практически каждый человек, пользующийся компьютером и интернетом [1]. Примером этого является выполнение поиска писем в электронной почте. На данный момент информационный поиск становится более популярным, чем какой-либо формальный поиск (например, поиск заказа по его номеру).

Так как информационный поиск применяется на больших данных, многие его алгоритмы масштабируемы и распределяемы (например, алгоритм построение инвертированного индекса по большой текстовой коллекции). По аналогичной причине изучаются методы сжатия словарей и инвертированных индексов, с целью уменьшить размер хранения [1]. Именно благодаря этому в поисковых системах, работающих на огромных данных, пользователь получает результат поиска меньше чем за секунду.

1.2 Анализ существующих решений

Новость на новостном веб-сайте представляет собой веб-страницу с контентом, а, следовательно, может обрабатываться поисковыми системами так же, как обрабатываются другие веб-страницы. Поэтому поиск новостей можно выполнять через обычный поиск веб-страниц, что есть главная функция поисковых систем. Однако такой поиск не будет учитывать желание

пользователя искать только новостные статьи, поэтому его нельзя считать за применение информационного поиска для обработки новостей.

Конкретным примером новостного агрегатора является Google News. Интерфейс навигации по разделам представлен на рисунке 1.2. Интерфейс Google News представлен на рисунке 1.3. Он обладает следующими функциями:

- поиск новостей по ключевым словам, аналогично обычному поиску веб-страниц;
- подборка последних новостей;
- учёт оригинальности и цитируемости источников;
- персонализация подборки и результатов поиска на основе информации о взаимодействии пользователя с поисковой системой Google;
- добавление собственных категорий;
- добавление своих интересов вручную, что позволяет улучшить персонализацию;
- вывод ключевых тем, которые описаны в новостной статье, с целью сделать навигацию по новостям более удобной;
- поддержка множество языков (в отличии от специализированных на русский сегмент интернета поисковых систем).

Список категорий, которые поддерживаются в Google News:

- главные новости;
- в мире;
- Россия (либо любая страна, в зависимости от выбранного языка);
- бизнес;
- наука и техника;
- культура;
- спорт;
- здоровье.

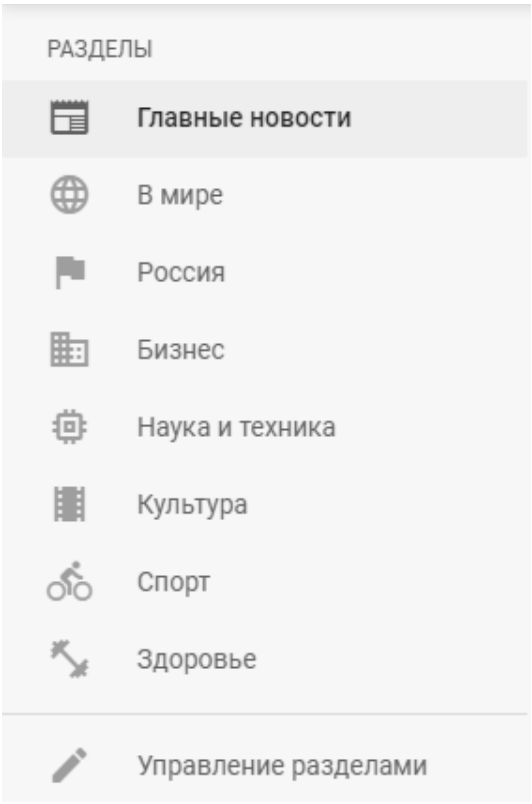

Рисунок 1.2 – Интерфейс навигации по разделам

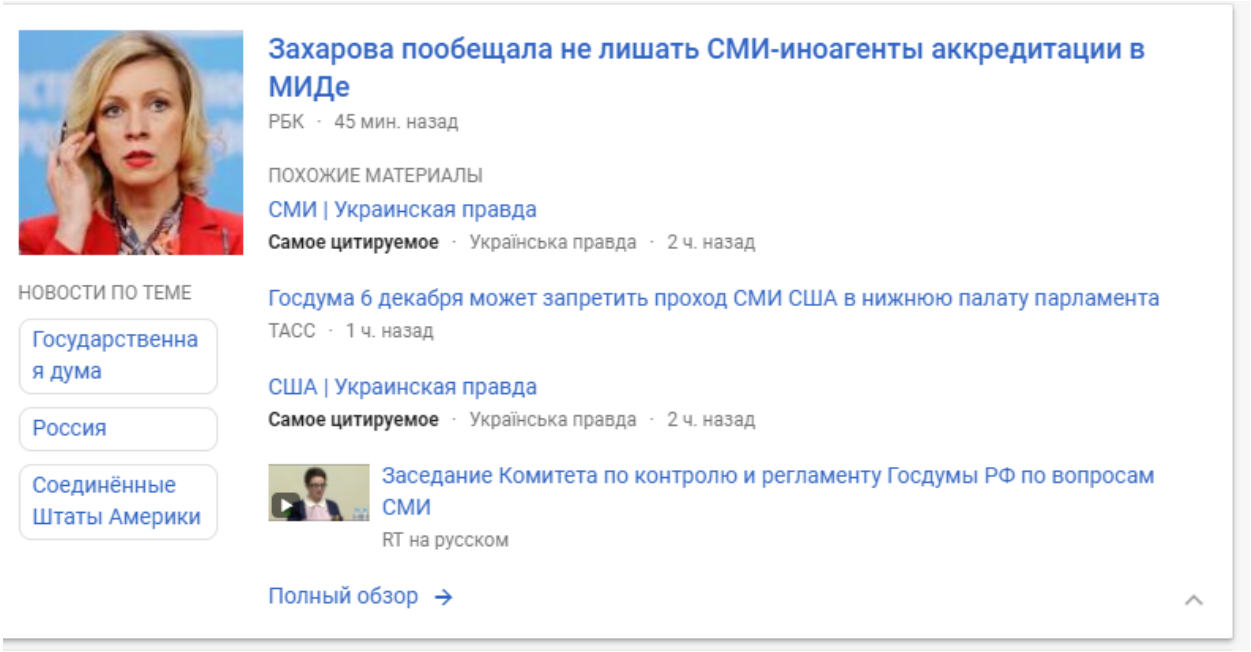

Рисунок 1.3 – Интерфейс Google News

Более простыми примерами являются аналоги от российских разработчиков, такие как «Яндекс Новости» и «Новости Mail.Ru». Они аналогичным образом анализируют источники, оригинальность, цитируемость и позволяют выполнять поиск – то есть всё так же применяют

возможности своих поисковых систем в обработке новостных сайтов. Но они имеют строгий набор категорий, без возможности персонализации. Примеры их работы на рисунках 1.4 и 1.5.

Категории в Яндекс Новости:

- Главное;
- Самара (или другой регион пользователя);
- Политика;
- Общество;
- Экономика;
- В мире;
- Спорт;
- Происшествия;
- Культура;
- Технологии и наука;
- Авто.

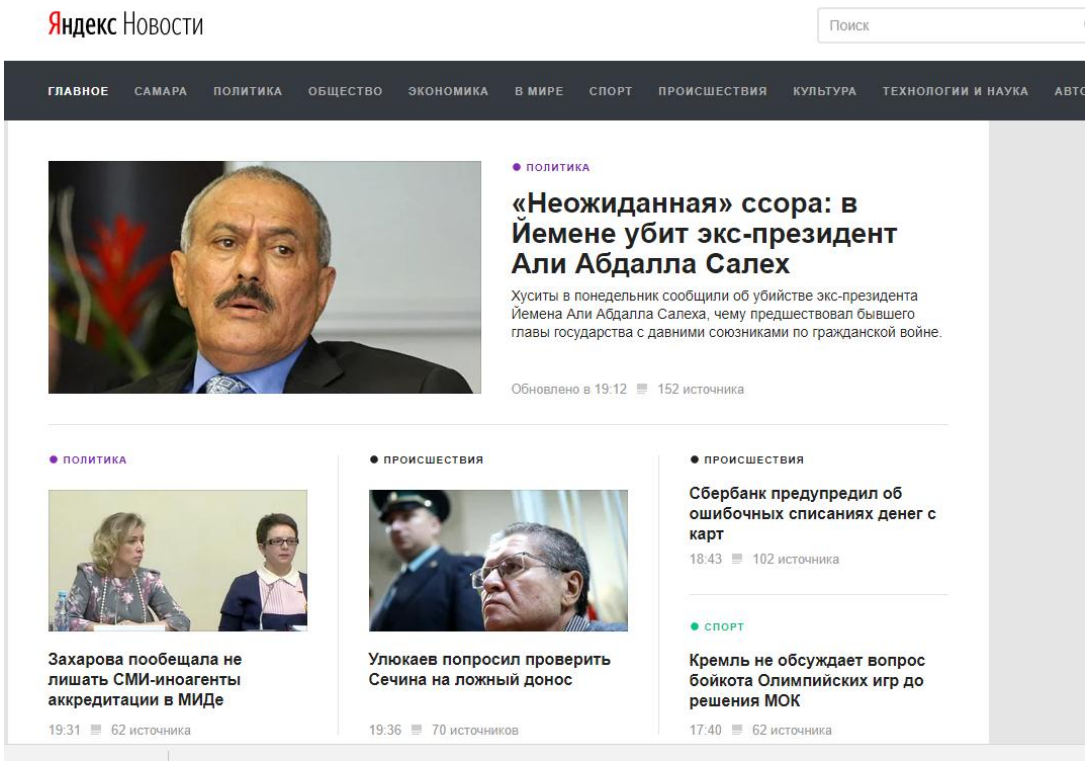

Рисунок 1.4 – Интерфейс Яндекс Новости

Категории в Новости Mail Ru:

- картина дня;
- Самарская обл. (зависит от региона пользователя);
- экономика;
- общество;
- события;
- спорт;
- курсы;
- погода.

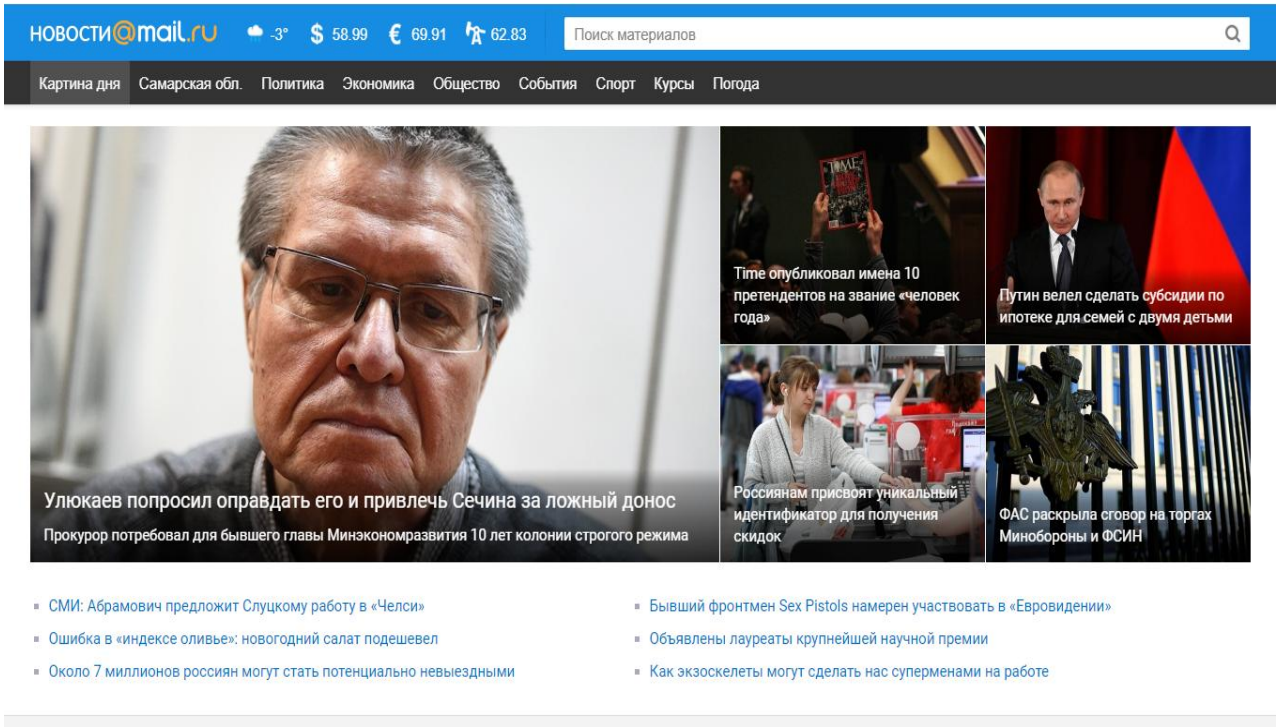

Рисунок 1.5 – Интерфейс Новости Mail Ru

Рассмотренные новостные агрегаторы используют веб-интерфейс. Самым мощным по функциональности из рассмотренных агрегаторов оказался Google News. Все агрегаторы интегрированы в поисковые системы, и обладают благодаря этому своими главными достоинствами:

- автоматическое нахождение источников;

- учёт психологического портрета пользователя, сформированного при взаимодействии с поисковой системой;
- учёт геолокации пользователя.

Однако подобная привязанность к поисковым системам несёт для арегаторов и определёнными минусы:

- закрытость программных продуктов и их неотделимость от поисковых систем;
- заданный набор тематик (хотя у Google News есть возможность добавлять свои, но для этого потребуется знать ключевые слова).

Последний недостаток можно назвать самым главным. Статические тематики отличаются большой негибкостью, так как они не способны подстраиваться под тренды. Например, интересующиеся темой про биткойны найдут её в новостях про финансы или экономику, вместе с остальными новостями на данную тематик. Именно данный недостаток мы попытаемся устранить при помощи методов информационного поиска, а именно кластеризации новостных статей и наименования кластеров.

1.3 Описание требований к программному продукту

Как было решено во второй главе, новостной агрегатор должен генерировать динамические темы для некого набора новостных статей с помощью алгоритма кластеризации k-means++ и назначать каждому кластеру имя, отражающее его содержание за счёт самых популярных векторов центроида. Набор последних за некоторый промежуток времени новостных документов должен браться из различных источников с помощью RSS. Программа должна пройти по ссылкам на новостные статьи в RSS и загрузить их на компьютер для дальнейшей обработки. Так как новость представляет собой HTML-страницу, требуется, чтобы программа умела их парсить с целью достать только текст новостной статьи. HTML-страницы из одного источника имеют одинаковую структуру, поэтому вычленение текста

можно достигнуть, задав правила парсинга для каждого источника, с которым будет работать программа. Получаем следующие задачи:

- загрузка RSS для источника;
- извлечение из RSS ссылок на новостные статьи;
- загрузка новостных статей;
- извлечение текста из HTML-страниц новостных статей;
- преобразование новостных статей в вектора с помощью векторной модели;
- кластеризация документных векторов с помощью алгоритма k-средних++;
- генерация имён для кластеров документов.

Пользовательский интерфейс программы должен показывать кластеры (имена кластеров) с целью обеспечения навигации пользователя по новостям. Внутри кластера пользователь должен видеть базовую информацию. Для каждой новости нужно показывать:

- заголовок;
- дату публикации;
- удалённость от кластера;
- ссылку на источник.

Чтобы пользователь мог видеть самые новые новости внутри кластера, программа должна поддерживать сортировку новостей по дате публикации. Исходя из того, что кластер собирает в себе новости неидеально, нужно дать возможность сортировать новости внутри кластера по удалённости от центроида, чтобы иметь возможность видеть самые близкие к сути кластера. Для просмотра новостей от интересующего источника, нужна сортировка по источнику. Для упрощения поиска по заголовку – сортировка по заголовку.

На случай, если пользователю хочется посмотреть последние новости независимо от кластера, можно достигнуть это с помощью показа пользователю всех новостей. Тогда последние новости будут получены через

сортировку по дате публикации. Для поиска нужного заголовка полезно будет добавить фильтрацию по заголовку. Итого, получаем следующие требования предоставляемым пользователю функциям:

- показ имён кластеров;
- возможность перейти на любой кластер и посмотреть его новости (заголовки новостей);
- возможность смотреть все новости;
- возможность сортировать новости по заголовку;
- возможность сортировать новости по дате публикации;
- возможность сортировать новости по удалённости от центроида;
- возможность сортировать новости по источникам;
- возможность фильтровать новости по заголовку;
- возможность открыть новость в браузере, кликнул на неё.

После запуска программы, на интерфейсе должны быть отображены кластеры. Причём одним из отображаемых кластеров должен быть кластер «Все новости», открывающий новостные статьи, собранные из всех кластеров. Заметим, что в этом случае не нужно показывать удалённость от центроида.

2 АНАЛИЗ ПРИМЕНЕНИЯ КЛАСТЕРИЗАЦИИ В НОВОСТНОМ АГРЕГАТОРЕ

2.1 Применение кластеризации для разделения на темы

Все выше рассмотренные новостные агрегаторы обладают существенным недостатком – они не могут подстроиться под популярные на определённый момент времени темы, то есть они не обладают необходимой гибкостью. Это из-за применяемой в них статической классификации новостей, в которой есть заданный набор тем и происходит разделение всех новостей между этими темами. Статическая классификация требует ручного вмешательства, так как даже применяя методы машинного обучения, требуется сначала обучить модель на уже проклассифицированных вручную новостных документах. Всё усложняется тем, что подобные модели, как и любые другие модели в информационном поиске, работают с некими терминами – ключевыми словами, содержащимися в документе. Ключевым словом может быть имя политика, спортсмена, название организации и т.п. Со временем эти слова меняются (появляется новое лицо в политике или спорте, старое лицо уходит), и обученная модель становится неспособной так же эффективна классифицировать новости. Поэтому требуется усложнять модель, позволяя ей автоматически дообучаться либо на основе проклассифицированных самостоятельно новостей, либо на основе такой же вручную проклассифицированной выборке.

Для решения данной проблемы можно использовать динамическое разделение документов по темам, что называется кластеризацию. Применение кластеризации на новостные статьи является задачей информационного поиска о применении кластеризации на документы. Смысл кластеризации документов заключается в кластерной гипотезе – гипотезе, что документы из одного кластера являются релевантными к некоторым информационным потребностям [1]. Из этого следует, что кластеризацию новостных статей можно применить для разбиения всей новостной коллекции на некоторое количество тем, и упростить пользователю навигацию. Кластеризация может применяться как на

общий набор документов, так и на поднабор, полученный по определённому поисковому запросу. Как и остальные методы машинного обучения, кластеризация не гарантирует идеальный результат [1]. Эффективным будет считаться та кластеризация, которая сумеет помочь пользователю ориентироваться в новостных статьях.

Результатом кластеризации будут кластеры, в каждом из которых содержится подмножество общего множества документов, объединённые некоторой общей темой. Данные кластеры не будут привязаны к конкретным темам, как в случае статических тем. Наоборот, кластеры будут зависеть от коллекции документов. Так как в новостном агрегаторе эта коллекция зависит от времени, то и кластеры будут зависеть от времени, подстраиваясь под коллекцию. Следовательно, это устраняет недостаток статических тем, применяемый в существующих агрегаторах. В свою очередь, даже если забыть про погрешность при применении машинного обучения для кластеризации статей, то, в отличии от формальных статических тем, провести чёткую границу, что считать за тему, и как определить принадлежность новости к этой теме довольно сложно. Поэтому неточность разбиения новостей на темы при кластеризации повысится. Это можно считать недостатком подхода динамического разделения новостей по темам.

Для кластеризации новостей потребуется использовать какой-либо метод кластеризации. В данном случае требуется выполнить жёсткую и плоскую кластеризацию. Жёсткая кластеризация подразумевает собой строгое разделение элементов по кластерам (т.е. один элемент принадлежит ровно одному кластеру). Плоская кластеризация подразумевает отсутствие взаимосвязей между кластерами (в противовес ей ставится иерархическая кластеризация). Одним из самых популярных методов плоской и жёсткой кластеризации является метод k-средних. Задача кластеризации методом k-средних является одной из старейших и важнейших задач вычислительной геометрии. Данная задача является NP-трудной, но 25 лет назад Лойд представил метод поиска локального решения, и этот метод активно

применяется и по сей день. Обычно называемый просто «методом k-средних», алгоритм Ллойда начинается с произвольных k центров, которые случайным образом достаются из общего множества точек. Потом каждая точка присваивается ближайшему центру, и далее каждый центр пересчитывается как центр масс присвоенных ему точек. Эти последние два шага повторяются, пока процесс не стабилизируется. Можно проверить, что значение ошибка кластеризации при этом монотонно уменьшается, подтверждая, что в ходе алгоритма никакая конфигурация не повторяется. Так как есть только k^n возможных исходов кластеризации, процесс всегда имеет конец. Привлекательным для использования метод делают его скорость и простота, но не точность. Причём это зависит не от специально выбранных плохих центров, плохие результаты могут выдаваться с высокой вероятностью даже при случайно выбираемых центрах.

Существует улучшенная версия данного алгоритма, которая отличается способом выбора начальных центров для алгоритма k-средних. В ней вероятность взятия нового центра рассчитывается по формуле

$$p(x) = \frac{D\ x\ ^2}{\mathrm{x} \in \mathrm{X} D\ x\ ^2} \#\ 1$$

где x – точка, рассматриваемая как центр;

X – множество всех входных точек;

$D\ x\ $ – расстояние от x до ближайшего центроида.

Первый центр выбирается случайно. Остальные центры выбираются в соответствии с формулой 1. Эта модификация называется k-средних++ [5].

2.2 Векторные модели для новостных документов

Для применения метода k-средних++ на документы нужно некоторым образом определить вычисления центра для заданного набора документов и дистанции от центра до какого-либо документа. В информационном поиске данная проблема решается с помощью представления документа в виде вектора, где каждому термину (из всего набора терминов) соответствует некоторое действительное число. С векторным представлением работать

удобно, поэтому данный подобный подход популярен не только в кластеризации, а в любых других методах машинного обучения над документами.

Векторная модель является популярной, самой распространённой алгебраической моделью для представления текстовых документов в виде векторов идентификаторов. В ней каждый документ может быть представлен как многомерный вектор ключевых слов (извлечённых из документа) в эвклидовом пространстве. Вес каждого ключевого слова определяет релевантность этого слова в документе [6]. Следовательно, документ в виде вектора можно представить следующим образом

$$D_j = w_{1j}, w_{2j}, w_{3j}, w_{4j}, .., w_{nj} \quad \# \ 2$$

где w_{ij} – вес ключевого слова i в документе j;

n – общее количество ключевых слов.

Размерность всех векторов будет фиксированной. Стоит отметить, что рассчитываются веса именно для ключевых слов (далее, терминов), так как в большинстве случаев используются не слова документов в исходном виде, а их нормализованный вариант. Понятие термин означает класс всех лексем, состоящий из одной и той же последовательности символов, включенный в словарь системы информационного поиска [1]. В данном случае будет выполняться простейшая нормализация (сложная нормализация должна быть осторожна, так как в итоге может сильно ухудшить результат), в которой будут обрезаться последние буквы «й», «ь», «ъ» и гласные, если исходное слово имеет длину более 4 букв. Также все небуквенные символы будут заменены на пробелы с целью упрощения поиска лексем. Минимальная нормализация – важный процесс для получения хорошего результата, однако дальнейшее её улучшение является не настолько решающим фактором, как выбор подходящей векторной модели. Рассмотрим три популярных способа векторного представления документов, начиная от самого простого и заканчивая более сложным:

1. наличие/отсутствие термина в документе;

2. по частоте появления термина в документе (нормализованный вариант модели «мешок слов»);

3. по частоте появления термина в документе и обратной частоте документов с данным термином (модель TF-IDF).

Попробуем провести кластеризацию методом k-средних++ на этих моделях и сравним результаты. Для каждой модели кластеризация будет происходить 5 раз, чтобы компенсировать неудачный выбор начальных кластеров. Итоговый результат – кластеризация с самой минимальной квадратичной ошибкой. Количество кластеров – 15.

В качестве дистанции будем использовать угловое расстояние. Это расстояние хорошо подходит для векторных моделей документов. Данная дистанция учитывает только схожесть векторов, когда эвклидова дистанция сближает вектора в том числе и по отсутствию тех или иных терминов. Данная дистанция похожа на косинусную меру схожести. Так как во всех моделях вектора будут нормализированы, косинусную меру схожести можно вычислить с помощью скалярного произведения векторов. Тогда угловая дистанция между векторами определяются по следующей формуле:

$$d\ x, y\ = \frac{arccos\ x * y}{\pi}\ \#\ 3$$

где x и y – вектора, расстояние между которыми ищется.

Первая рассматриваемая модель проставляет значение компонента вектора по наличию термина в документе. Если он присутствует, то значение равно 1, если отсутствует – 0 [1]. Получается следующая формула весов для документа D_j:

$$w_{ij} = \begin{matrix} 1, \text{если } D_j \text{ содержит термин } i \\ 0, \text{иначе} \end{matrix}\ \#\ 4$$

Далее вектора нормализируется и на них выполняется кластеризация. Результат кластеризации: 7/15. Подобная модель может использоваться для классификации, так как новости можно различать по наличию тех или иных ключевых слов, и модель обучения с учителем способна стать чувствительной к ним. Но для кластеризации (обучение без учителя) данная модель подходит не

так хорошо, потому что тогда какое-либо местоимение имеет такое же влияние на результаты кластеризации, как и какое-либо действительно важное ключевое слово (например, чья-либо фамилия или название организации). В данной модели самым большим весом будут обладать компоненты для терминов, содержащихся в наибольшем количестве документов кластера (но совсем необязательно, что данные термины будут что-то говорить о кластере).

Вторая модель каждому термину присваивает частоту его появления в документе. Потом все векторы нормализируются (приводятся к единичному вектору), чтобы можно было оценить распределение терминов [1]. Если не нормализировать, а брать только по частоте терминов, то результат будет портить длина вектора (размер документа). На практике данная модель показала себя лучше предыдущей: 10/15. Данная модель лучше первой тем, что учитывает частоту появления термина в документе.

Третья модель называется «tf-idf». Эта модель является одной из самых распространённых для выполнения поиска документов по запросу [7]. Проверим её эффективность для новостных документов на практике. TF-IDF работает путём определения относительных частот слова в некотором документе в сравнении с обратной пропорцией этого числа во всём корпусе документов. Интуитивно понятно, что данное вычисление определяет насколько релевантно данное слово в данном документе. Слова, которые встречаются в небольшом количестве документов имеют более высокие значения TF-IDF, чем часто повторяющиеся слова, такие как артикли и предлоги [8]. Из-за своей монотонности, данная модель присваивает относительно высокие веса редким (нечастым) словам и относительно низкие веса очень частым словам. Согласно теории информации, такая схема способна определить вес информации каждого слова, содержащегося в словаре [9].

Формальная процедура реализации TF-IDF немного отличается в зависимости от её применения. Веса в векторе документа D_j перед нормализацией рассчитывается по формуле

$$w_{ij} = \log\frac{N}{df_i} * \begin{matrix} 1 + \log tf_{ij}, \text{если } tf_{ij} > 0 \\ 0, \text{иначе} \end{matrix} \quad \# \ 5$$

где tf_{ij} – частота термина i в документе D_j;

N – общее количество кластеризируемых документов;

df_i – количество документов, содержащих термин i.

На практике данная модель показывает себя лучше других двух и выдает 12/15. Концентрация новостей в кластер происходит по ключевым словам. Они перевешивают «мусорные» слова благодаря обратной документной частоте (чем больше документов, в которых есть данный термин, тем менее значим он), применяемой в данной модели, и поэтому набирают самый больший вес в центроиде. Исходя из результатов, в новостном агрегаторе имеет смысл применять модель TF-IDF.

2.3 Наименование кластеров

Для того, чтобы пользователь мог заранее видеть примерное содержание кластера, и решить, стоит ли ему туда навигироваться, нужно каким-либо образом этот кластер именовать. По сравнению с заранее заданными темами, у кластеризации есть следующие главные отличия:

- именование кластера не задаётся заранее, а исходит из его содержания;
- кластеру трудно дать название в виде слова или словосочетания из-за его динамической сути.

Рассмотрим два способа наименования кластера. Первый способ заключается в том, что в качестве имени кластера берётся самый близкий к центроиду элемент, т.е. именем кластера будет заголовок самой близкой к центроиду новости. Плюсом такого подхода является понятность заголовка. Он ведь уже написан человеком, а не сгенерирован программой. Минусом подхода является невысокая возможность представить весь кластер одной новостью. Хотя она и близка к центроиду, и поэтому может дать общее представление о кластере, заголовок одной новости не всегда способен полностью раскрыть его содержание. В качестве устранения недостатка можно показать пользователю не одну, а некоторое число самых близких к центроиду новостей. Однако и в

таком случае, новостей может быть недостаточно, а пользовательский инетфейс будет захламлён. Пример работы этого способа представлен на рисунке 2.2.

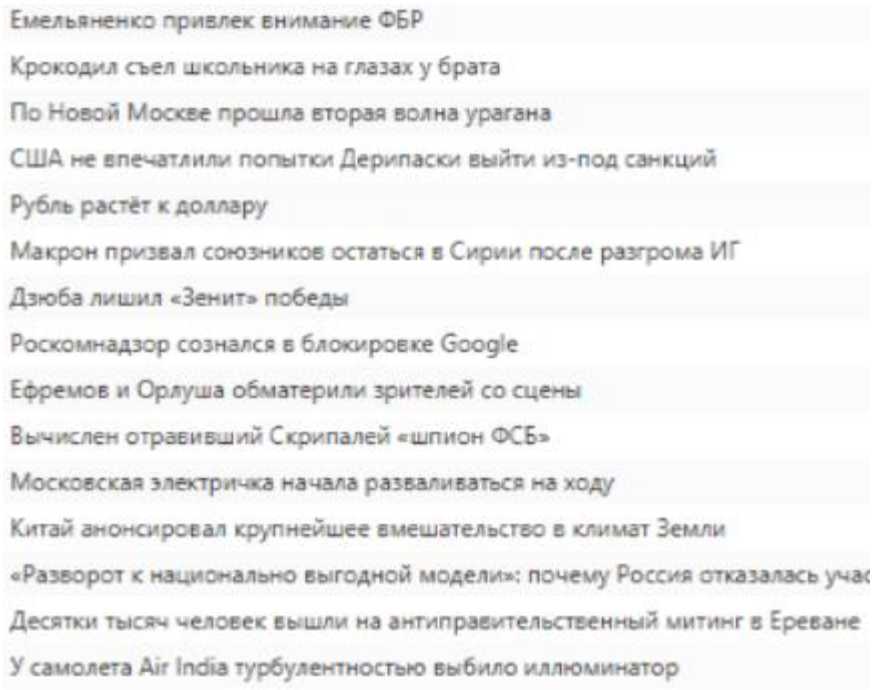

Рисунок 2.1 – Пример именования кластеров на основе заголовков самых близких к центроиду документов

Во втором способе в качестве имени кластера перечисляются через пробел термины центроида кластера. Перечисляются они в порядке уменьшения веса в векторе центроида. В данном случае возьмём первые 10 терминов. Исходя из логики высчитывания центроида в алгоритме k-средних++ (центр масс), самые большие значения получат те термины, которые имели больший вес в векторах кластера. Поэтому эти термины смогут дать пользователю представление о содержании кластера, что является плюсом подхода. Однако набор терминов не всегда удобно читать, что является несомненным минусом. Слово после нормализации может потерять окончания, и все термины переведены в нижний регистр. Пример работы этого способа представлен на рисунке 2.2. Можно исправить эту проблему, переводя термины обратно в слова (например, в самое первое слово, которое было преобразовано в этот термин). Однако читабельность наименования кластера всё равно будет хуже, чем в случае заголовков. Пример представлен на рисунке 2.3.

умер врач человек лет больниц актер сам дом смерт умерл
матч сборн лиг команд чемпионат клуб зенит арсенал счетом тренер
русал санкц сша минфин компан процентов финансов дерипаск алюмин трамп
принц уильям уэст сын кейт кань герцогин рэпер кембриджск певец
ураган москв мчс погод столиц ветр пожар ветер станц гроз
курс доллар рубл бирж торгов евр нефть вырос санкц свидетельствуют
тел великобритан кож британских вид рак гусениц daily учен университет
украин донбасс киев операц лиц задержан дел отношен конфликт задержал
telegram мессенджер роскомнадзор блокировк пользовател google суд доступ адресов фсб
самолет аэропорт авиакомпан штат рейс посадк инцидент аварийн разбилс пассажир
емельяненк ufc поединк боец россиянин стил bellator смешанног побед американц
полицейск полиц город улиц человек автомобил машин дорог толп пешеходов
армен лидер кор пашинян саргсян премьер трамп кндр ереван чен
земл законопроект тысяч быть петербург москв екатеринбург рубл вод люд
газ газпром групп украин сектор дидж смерт avicii транзит берглинг

Рисунок 2.2 – Пример именования кластеров на основе весов центроида

Украины украинский армии Донбассе военные Польши Порошенко памятник Киеве АТО
матчей лиги сборной Реала команду Зенит клуб нападающий минуте чемпионов
курс доллара рублей бирже евро Торговый московской вырос мск индекса
дождя улице погоды тысяч надувная грозы полосу ветра городе водитель
принц Уильям Артур сын Джигарханяна Кейт пользователи обложку снимок света
Кореи КНДР Ким Чен Мун полуострова МЧС корейского южную двигаться
НПФ регулятора лицензию страхования пенсионного системы ставки пенсионная нарушения сохра
Емельяненко россиянин Bellator Гран боец поединка UFC смешанного весе карьеры
суда отношении дела действия Москву детей мужчина возраста виновным летия
чемпионат мира сборной IBU спортсмен федерации соревнований союзе биатлонистов турнира
института полигон Facebook работе увольнении университета Telegram странице Максим имени
Telegram технологий основатель мессенджера компания Amazon Google Роскомнадзора пользовате
самолеты посадки военные аэропорту разбился судна Джорджа транспортные BBC 130
Пашинян Армении премьер партия парламента оппозиции Республиканская министр фракции акци
Трампа США президент Дональда Франции Макрон Кореи Вашингтон домой страны

Рисунок 2.3 – Пример именования кластеров на основе весов центроида с восстановлением исходных слов

Как видно по рисунку 2.1 и 2.2, второй способ показал себя лучше. Хотя первый способ большинству кластеров дал подходящие названия, содержание некоторых кластеров по их названию неочевидно. Третий же способ (рисунок 2.3) мы и будем использовать, потому что он устраняет недостаток второго способа.

3 РАЗРАБОТКА НОВОСТНОГО АГРЕГАТОРА С КЛАСТЕРИЗАЦИЕЙ НОВОСТЕЙ НА ОСНОВЕ ВЕКТОРНЫХ МОДЕЛЕЙ

3.1 Описание архитектуры программного решения

Новостной агрегатор будет иметь клиент-серверную архитектуру. Схема такой архитектуры представлена на рисунке 3.1. В ней все вычисления проводятся на стороне сервера, на который и будет основная нагрузка. Данная архитектура в разрабатываемом новостном агрегаторе обоснована следующими причинами:

- затратность процесса кластеризации;
- отсутствие причины повторять этот процесс на одних и тех же данных (нет смысла каждому клиенту заново кластеризировать новостные документы, гораздо выгоднее выполнять кластеризацию на стороне сервера);
- необходимостью хранить данные для проведения кластеризации (нет смысла хранить их на каждом клиенте отдельно).

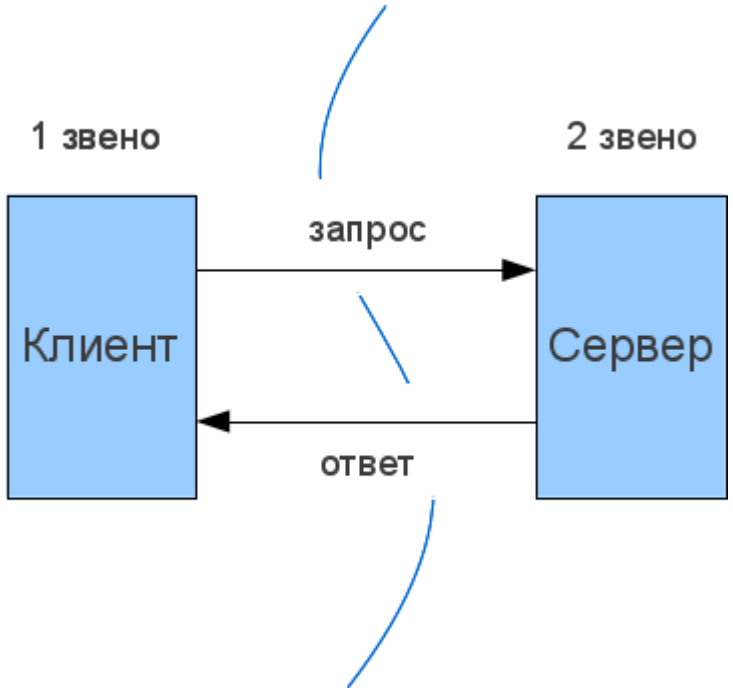

Рисунок 3.1 – Схема клиент-серверной архитектуры

Клиент – программа, выполняющаяся на стороне пользователя, в задачи которой входят запрос с сервера результаты кластеризации и выведения их на пользовательский интерфейс. Данный клиент можно охарактеризовать как тонкий. Клиент должен иметь интуитивно-понятный интерфейс,

удовлетворяющим описанным требованиям, и предоставлять пользователю основной функционал новостного агрегатора.

В задачи сервера входят задачи по сборке, обработке и предоставлению результатов кластеризации клиентской стороне. Именно в серверной стороне реализован основная логика новостного агрегатора. Сервер должен периодически загружать новые новости и выполнять кластеризацию на новых данных.

Сервер запускается и подгружает новости (с жёсткого диска и с интернета по источникам). Клиент позволяет пользователю подгрузить с сервера последние результаты кластеризации. Благодаря такой архитектуре, множество клиентов может получить результаты кластеризации, выполнив её один раз на стороне сервера.

3.2 Описание используемого языка и библиотек

Для реализации клиентской и серверной частей новостного агрегатора будет применятся один из самых популярных языков программирования – Java 8. Под этот язык существует множество готовых библиотек, которые помогут облегчить процесс разработки [10]. Разработка будет происходить в среде IDEA. Чтобы было удобно импортировать существующие библиотеки, для сборки проекта будет использоваться Maven – фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM, являющемся подмножеством XML.

Клиентская сторона будет представлять собой программу с графическим интерфейсом. В Java 8 самым удобным инструментом для построения GUI является JavaFX, так как он поддерживает вёрстку интерфейса на языке FXML [11]. Также JavaFX позволяет без проблем перенести интерфейс на платформу Android [12]. Для упрощения написания кода будем использовать Lombok, который позволяет заменять часто повторяющийся код с помощью аннотаций (например, геттеры и сеттеры), и Apache Commons Lang, реализующий часто используемый, но отсутствующий в стандартной библиотеке функционал (например, проверка строки на значение null или пустоту).

Серверная часть гораздо сложнее клиентской. Рассмотрим серверную сторону в соответствии с её работой. Во-первых, как и в клиентской части, нам потребуется библиотека Lombok и Apache Commons Lang для таких же целей. Далее учтём, что серверной стороне при работе с источниками часто приходится иметь дело с потоками ввода и вывода. Для этих целей будем использовать библиотеку утилит Apache Commons IO. Для кластеризации методом k-средних++ будем использовать реализацию от Apache Math. Для подсчёта частоты слов нам потребуется коллекция MultiSet, которая присутствует в библиотеки Apache Commons Collections. Для парсинга HTML-страниц и извлечения из них текста, будем использовать библиотеку JSoup.

Для взаимодействия клиента и сервера будем использовать средства Java API для веб-сервисов на основе XML (Java API for XML Based Web Services, далее JAX-WS) – это прикладной программный интерфейс языка Java для создания веб-служб, являющийся частью платформы Java EE [13]. Клиент и сервер должны иметь необходимые классы для представления передаваемых данных. Так как коммуникация клиента и сервера довольно простая – клиент просто запрашивает результаты кластеризации – JAX-WS является подходящим решением, простом в реализации. Итого следующие зависимости используются в файле pom.xml сервера:

- commons-math3;
- commons-collections4;
- jsoup;
- commons-lang3;
- commons-io;
- lombok;
- jaxws-api.

3.3 Описание структуры данных

Для того, чтобы избежать последовательного просмотра текстов при обработке документов, заранее составляется индекс документов. Один раз

собрав информация о терминах в документе, можно будет её использовать для дальнейшей работы.

Рассмотрим матрицу «термин-документ». Такая матрица содержала бы большое количество нулей (документ, в котором нет термина). Хранить такую матрицу со всеми её нулями неэффективно. Поэтому для хранения частот терминов в документах вместо неё используется структура данных, называемая инвертированным индексом. Сначала мы записываем в память словарь терминов. Затем для каждого термина мы создаем список, в котором указаны документы, содержащие данный термин. Каждый элемент списка, содержащий информацию о том, что термин находится в документе, а также (довольно часто) о координате термина в этом документе, называемой словопозицией [1]. Схема инвертированного индекса представлена на рисунке 3.2.

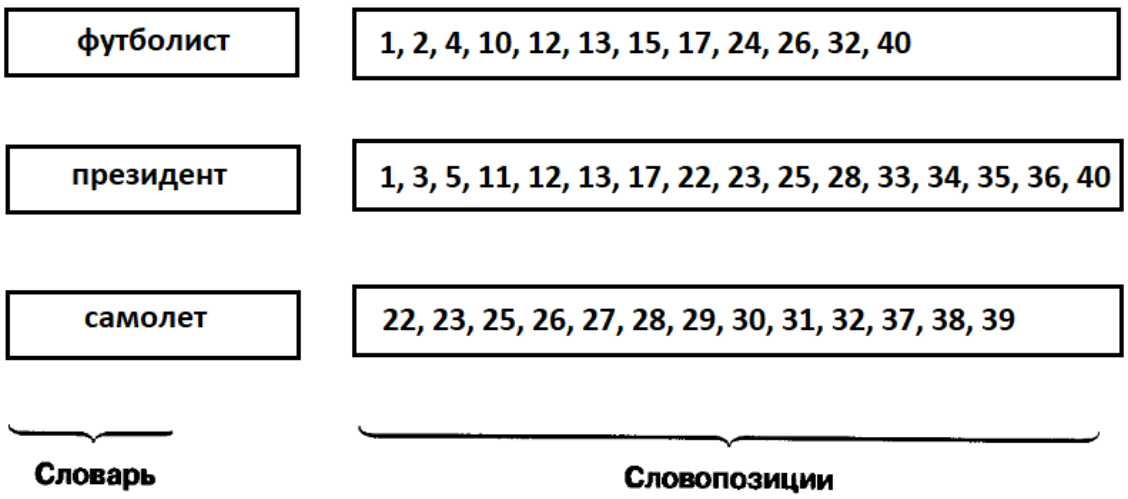

Рисунок 3.2 – Схема инвертированного индекса

Реализацией инвертированного индекса в новостном агрегаторе (а точнее, в его серверной части) являются классы InvertedIndex и InvertedList. Для хранения словаря используется класс HashMap [14], который связывает термин типа String с препозицией, реализуемой в классе InvertedList. Диаграмма классов инвертированного индекса представлена на рисунке 3.3. Диаграммы классов можно построить с помощью средств среды разработки IntelliJ IDEA [15-16].

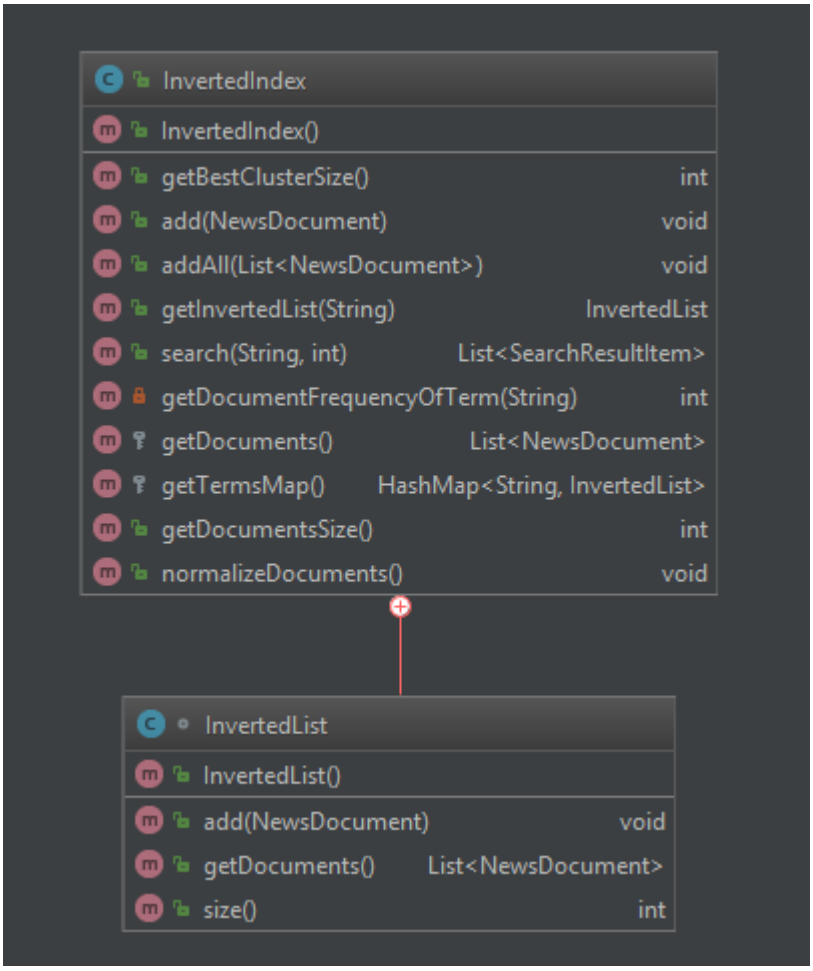

Рисунок 3.3 – Диаграмма классов инвертированного индекса

Как видно из рисунка, для хранения данных о новостной статье используется класс NewsDocument. Данный класс строится на основе JSoup-документа. О новости хранится следующая информация:

- дата публикации;
- заголовок;
- источник;
- ссылка (для открывания в браузере);
- содержание новости.

В данном классе происходит подсчёт частот терминов в документах. После окончательного формирования инвертированного индекса, частоты терминов нормализируются. Диаграмма класса NewsDocument представлена на рисунке 3.4.

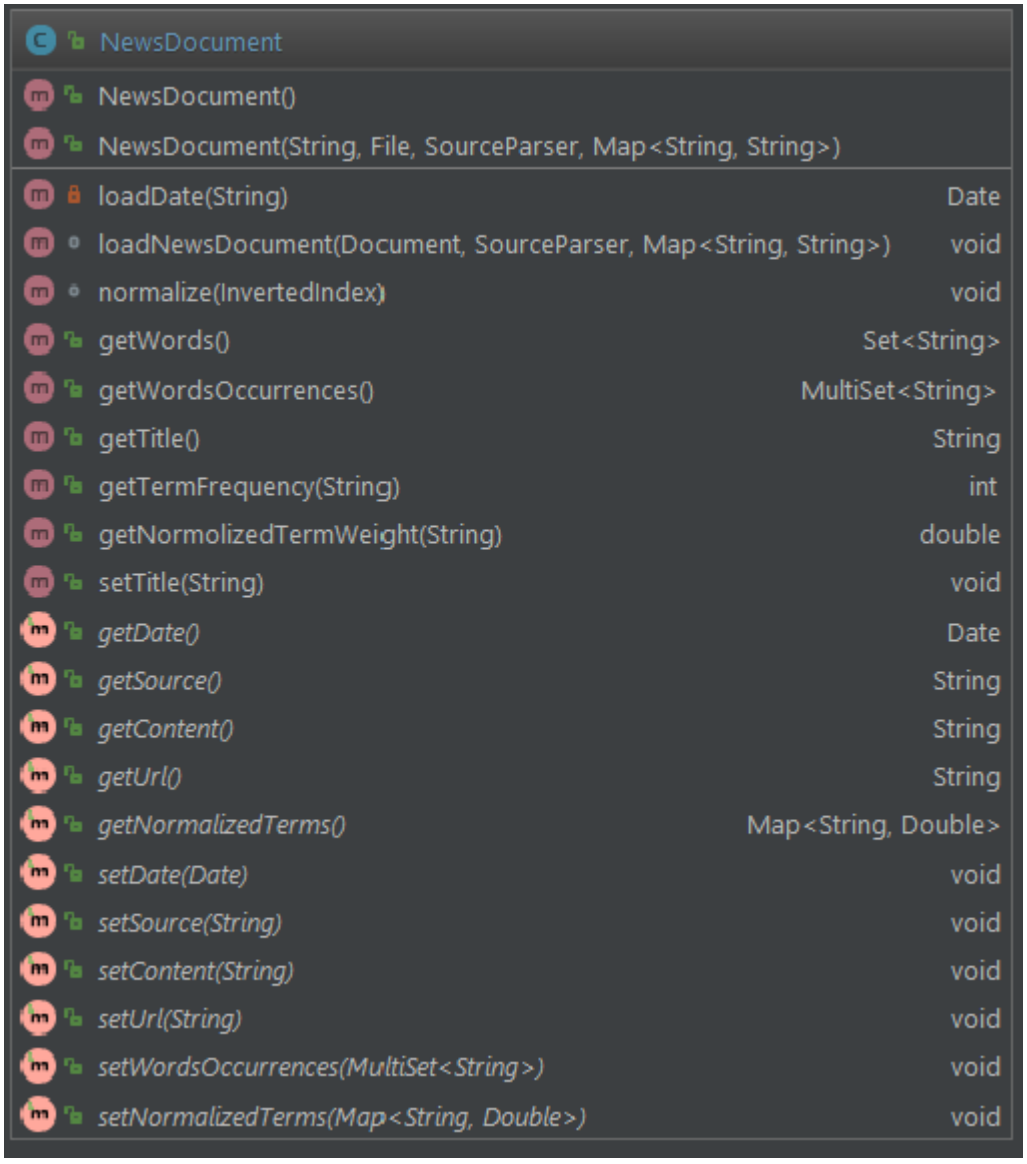

Рисунок 3.3 – Диаграмма класса NewsDocument

Для выполнения кластеризации на документах с помощью KMeansPlusPlusClusterer – класса от Apache Math, реализующего кластеризацию методом k-средних++ – требуется класс, реализующий интерфейс Clusterable. Этот контракт требуется для выполнения метода cluster, фактически аргумент этого метода – коллекция объектов, реализующих класс Clusterable. Исходный код кластеризации показан в приложении А.

Для этих целей реализуется класс DocumentVector. Класс довольно простой. Представляет собой обычный вектор в виде массива действительных чисел, и объекта типа NewsDocument, чтобы была возможность восстановить новость после проведения кластеризации. Диаграмма класса DocumentVector представлена на рисунке 3.4.

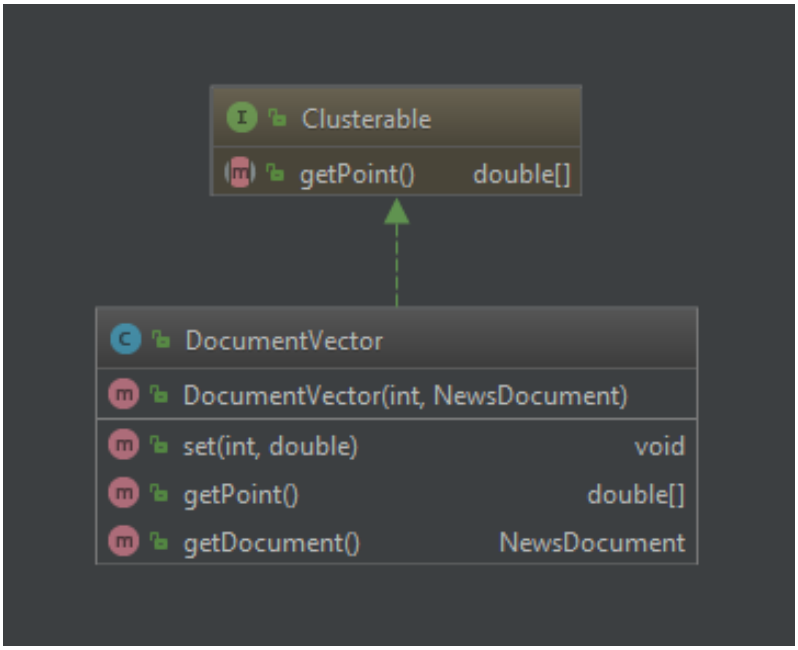

Рисунок 3.4 – Диаграмма класса DocumentVector

Для хранения соответствия термина тому или иному компоненту вектора используется класс VectorizedDocuments. Это довольно простой класс. Он хранит в себе список векторов и HashMap, где в качестве ключа используется термин, а в качестве значения – целое число (положительное). HashMap нужна для быстрого получения компонента (индекса массива вектора) по термину (строке). Диаграмма этого класса представлена на рисунке 3.5.

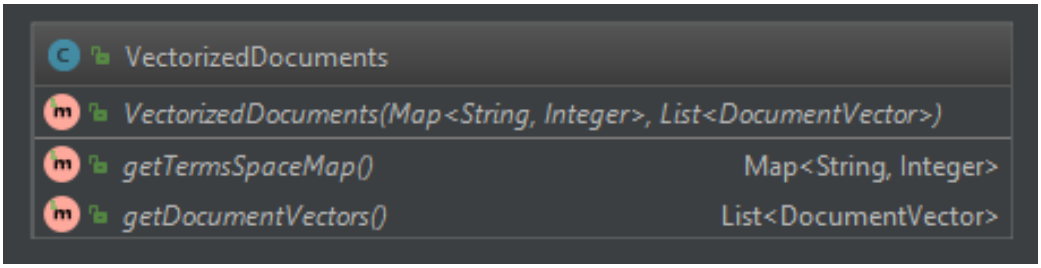

Рисунок 3.5 – Диаграмма класса VectorizedDocuments

Рассмотрим теперь класс, отвечающие за загрузку и хранение новостных статей. Класс RawNewsDocument создаётся на основе данных, полученных из RSS (заголовок, дата публикации, ссылка). При сохранении новости, экземпляр данного класса выкачивает новость по ссылке и сохраняет на жёсткий диск (с указанием директории сохранения и имени файла). Диаграмма класса представлена на рисунке 3.6.

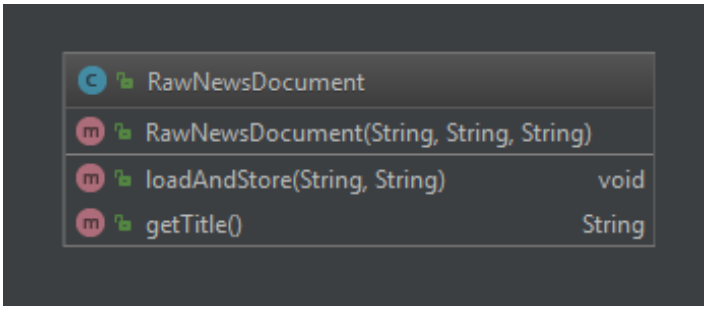

Рисунок 3.6 – Диаграмма класса RawsNewsDocument

После проведения кластеризации, мы должны получить результат. В дальнейшем этот результат будет передан клиенту. Для хранения результатов кластеризации используются следующие классы:

- ClusteringResult – класс для результата, один экземпляр отражает один результат кластеризации;
- ClusterModel – класс для хранения информации о кластере в результатах кластеризации;
- ItemModel – класс, представляющий собой новостную статью в результатах кластеризации.

Класс ClusteringResult хранит в себе ошибку кластеризации и список кластеров (ClusterModel). Класс ClusterModel хранит в себе информацию о имени кластера и списке его элементов (новостных статей, класса ItemModel). Класс ItemModel хранит в себе заголовок новости, источник, ссылку, дату публикации и дистанцию от центра кластера.

Как видно, данные классы хранят тот самый минимальный набор информации, требуемый клиенту для отображения результатов кластеризации. Какую-либо лишнюю информацию данные классы содержать не должны, так как будут пересылаться к клиенту по сети с помощью JAX-WS. В отличии от других описанных классов, данные структуры используются и на клиенте тоже.

Рассматривая эти классы в клиентской программе, стоит отметить, что есть небольшие отличия. JAX-WS для пересылке экземпляра, у которого есть поле-список, требует создание этого списка в конструкторе. В случае сервера, используется ArrayList, чтобы использовать известную информацию о количестве элементов в списке. На стороне

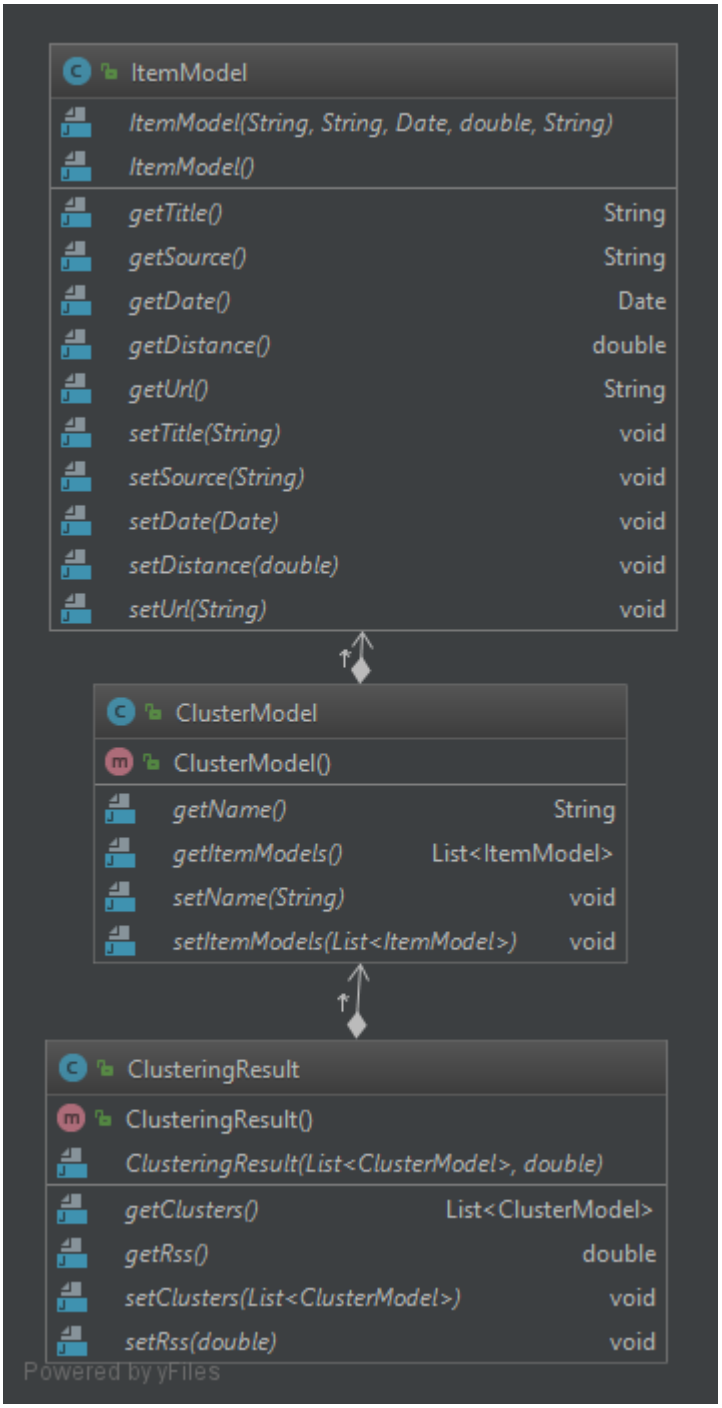

Рисунок 3.7 – Диаграмма классов для представления результата кластеризации

3.4 Реализация серверной части

Точка входа в программу – класс NewsAggregatorServer, который занимается подгрузкой новостных статей, их кластеризацией и запуском веб-сервиса, для предоставления результатов клиенту. Загрузка новостей в список объектов класса NewsDocument происходит с помощью статического метода класса DocumentLoader. Далее, подготовив объект класса DocumentProcessor, с помощью него производится кластеризация. Результат (ClusteringResult)

устанавливается в объект веб-сервиса для передачи данных клиенту. Веб-сервис реализуется интерфейсом ClusteringWebService и классом ClusteringWebServiceImpl. Для организации регулярной подгрузки, в геттере и сеттере результатов кластеризации в классе ClusteringWebServiceImpl используется Lock. Он обеспечивает безопасную работу нескольких потоков. В итоге, когда сервер закончил кластеризацию, он ждёт освобождение объекта lock, занимает его сам, подменяет результаты и освобождает для дальнейшего использования результатов.

Рассмотрим подробнее реализацию кластеризации и наименования кластеров. Для построения векторных моделей и их дальнейшей кластеризацией с генерированием имён для кластера используется класс DocumentProcessor. Диаграмма этого класса представлена на рисунке 3.8. Работа с классом происходит следующим образом:

1. создаётся экземпляр класса, который в дальнейшем будет использоваться для кластеризации;
2. добавляются новостные документы для формирования инвертированного индекса;
3. вызывается метод финализации, который запустит построение векторных моделей;
4. вызывается метод кластеризации для проведения кластеризации новостных документов.

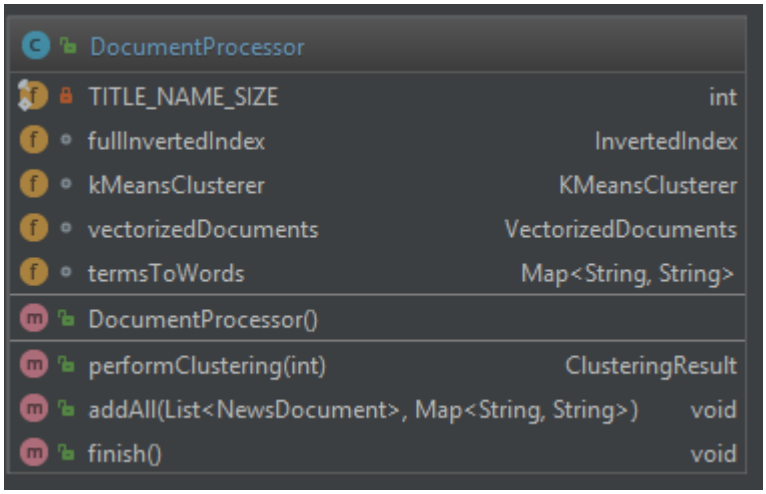

Рисунок 3.8 – Диаграмма класса DocumentProcessor

Метод добавления новостей просто собирает новости в инвертированный индекс. Фактически в данном методе всего лишь вызывается метод добавления нескольких новостей у объекта инвертированного индекса (метод addAll у fullInvertedIndex). Который уже вызывает метод add для каждого элемента из списка. Внутри метода add достаются термины новостного документа и добавляются в инвертированный список. Алгоритм добавления представлен на рисунке 3.9.

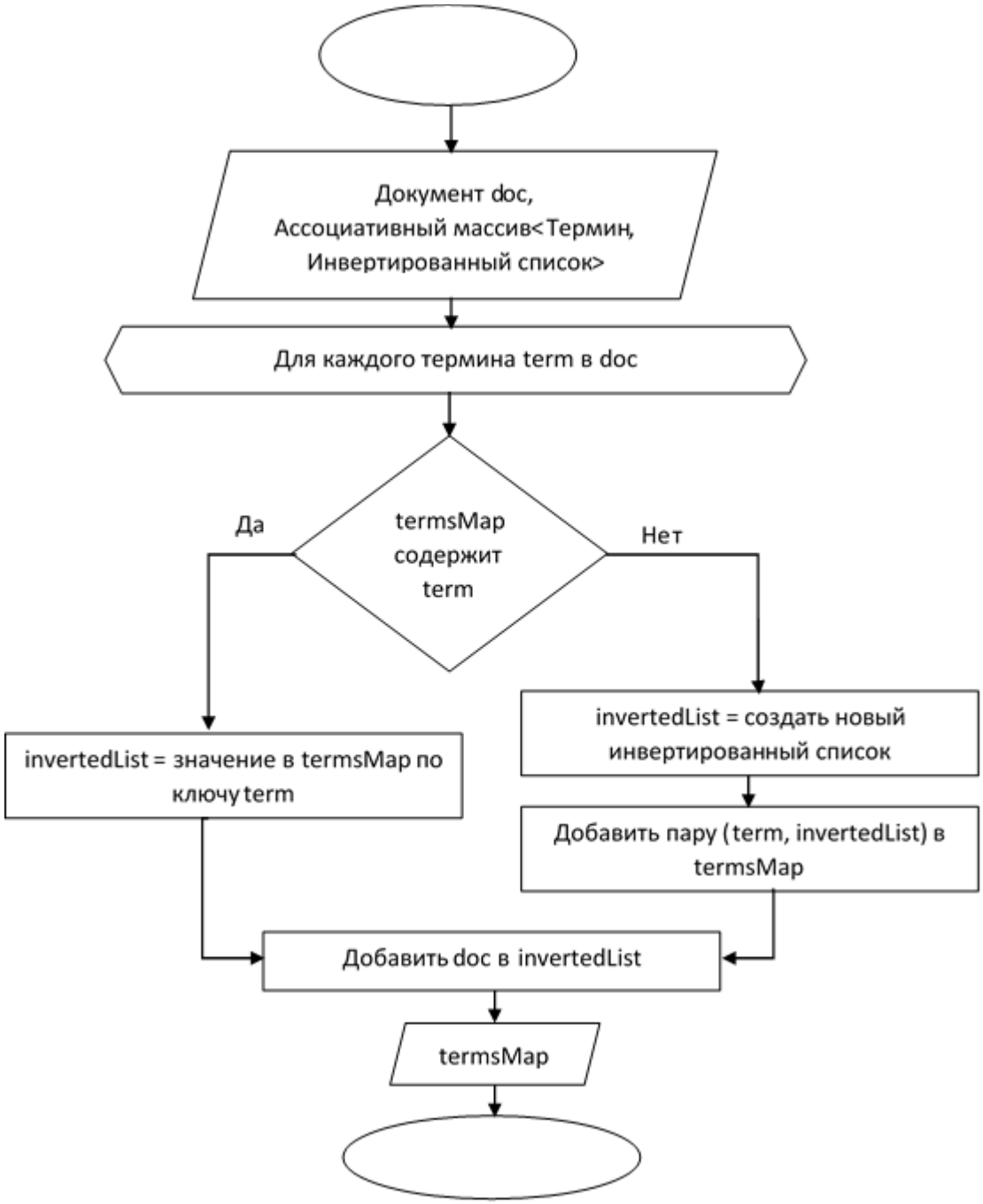

Рисунок 3.9 – Блок-схема добавления новости в инвертированный список

Кластеризация в классе DocumentProcessor выполнялась с помощью экземпляра класса KMeansClusterer (рисунок 3.10). Как видно по рисунку 3.10, данный класс создавался с указанием метрики. Интерфейс метрики берётся из библиотеки Apache Math. Есть список поддерживаемых метрик (например, эвклидовая метрика). Свою метрику можно сделать через реализацию этого интерфейса. В задачи класса KMeansClsuter входит полная кластеризация с генерацией имен и возвращением результата в виде ClusteringResult. DocumentProcessor в свою очередь просто оборачивает эту логику, вызывая метод с нужными параметрами (векторами документов, метрикой, размером имени кластера, количеством попыток).

KMeansClusterer тоже является обёрткой, над уже классом MultiKMeansPlusPlusClusterer (кластеризация методом k-средних++ заданное число раз подряд с выбором оптимального результата). После указания всех параметров, этот класс указывает список кластеров (класс CentroidCluster). Из него можно достать список элементов кластера и вектор центроида. Проведя кластеризацию, и достав результаты, KMeansClusterer формирует ClusteringResult.

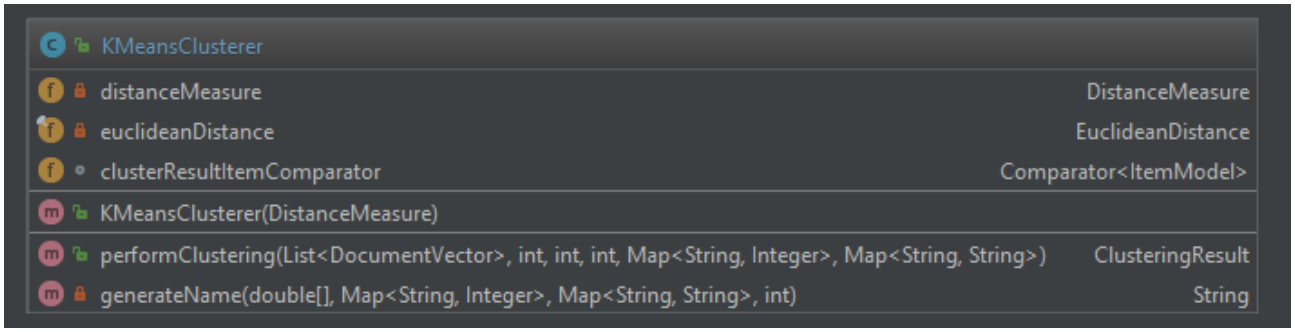

Рисунок 3.10 – Диаграмма класса KMeansClusterer

3.5 Реализация клиентской части

В клиентской части присутствует GUI, сделанный с помощью JavaFX. Точкой входа в программу является класс NewsAggregatorApp, расширяющий абстрактный класс Application с перезагрузкой метода start – именно таким образом запускаются JavaFX-приложения [17]. Внутри метода start создаётся PageManager. Потом из экземпляра PageManager (рисунок 3.11) достаётся сцена, которая устанавливается на главное окно.

PageManager отвечает за управления сценами. Как видно из логики работы метода start, загрузка сцен происходит в конструкторе данного класса. Помимо графики, данный класс также отвечает за установку соединения с сервером. Для переключения сцен используются соответствующие методы (рисунок 3.11). К каждой сцене привязан контроллер, отвечающей за обработку событий элементов управления. Все сцены описаны с помощью fxml, это является одной из главных достоинства JavaFX перед Swing [18]. Всего на пользовательском интерфейсе их две:

- сцена просмотра всех кластеров;
- сцена просмотра содержания кластера.

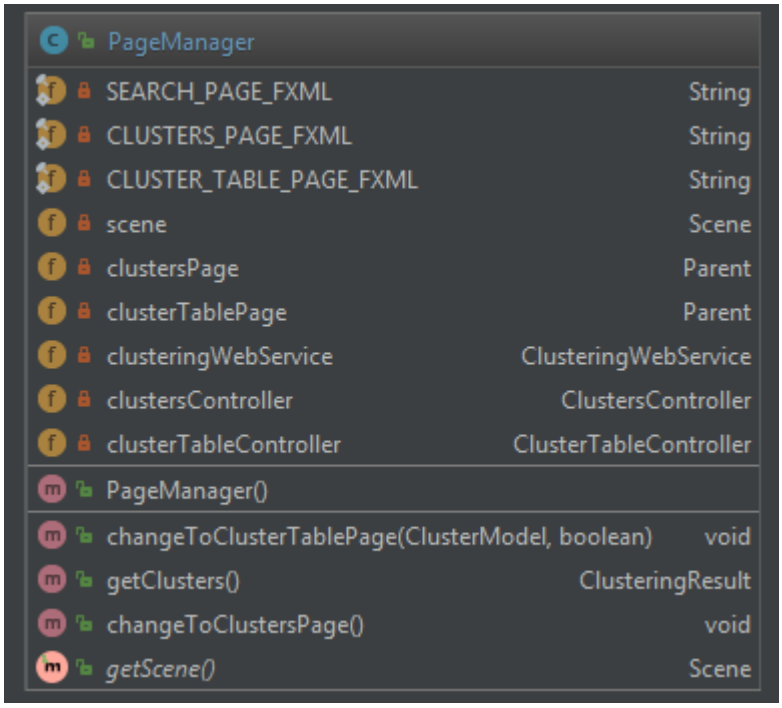

Рисунок 3.11 – Диаграмма класса PageManager

Первая сцена реализована в файле clusters_page.fxml. Она отвечает за отображения результатов кластеризации (ClusteringResult). Для этого имена всех кластеров показываются в виде элемента ListView. В качестве первого элемента устанавливается элемент «Все новости», отображающий все новости сразу. Для обновления результатов кластеризации используется кнопка «Обновить». События клика на кластер и на кнопку обрабатываются в

контроллере ClustersController. FXML сцены показан на рисунке 3.12. Пример работы показан на рисунке 3.13.

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<AnchorPane prefHeight="641.0" prefWidth="624.0"
            xmlns="http://javafx.com/javafx/8.0.121"
            xmlns:fx="http://javafx.com/fxml/1"
            fx:controller="application.ClustersController">
   <children>

      <Button fx:id="refreshButton"
              onAction="#handleRefreshButtonAction"
              AnchorPane.leftAnchor="20"
              AnchorPane.topAnchor="10" />


      <ListView fx:id="clusterListView"
                onMouseClicked="#onClusterClick"
                editable="false"
                onKeyPressed="#onKeyPressed"
                AnchorPane.bottomAnchor="20"
                AnchorPane.leftAnchor="20"
                AnchorPane.rightAnchor="20"
                AnchorPane.topAnchor="50" />
   </children>
</AnchorPane>
```

Рисунок 3.12 – Сцена кластеров в виде FXML

Второй сценой является сцена отображения содержания кластера. Показ всех элементов кластера выполнен с помощью TableView. Данный элемент из JavaFX удачно подходит, так как изначально поддерживает сортировку по элементам [19].

Для фильтрации новостей по заголовку, используется элемент TextField. Туда вводится подстрока заголовка. Для возвращения назад к списку всех кластеров используется кнопка «Назад». Нажатие на элемент кластера вызывает открытие страницы с новостью в веб-браузере. Событие изменения

фильтра (на котором происходит фильтрация), событие кнопки и событие выбора элемента кластера обрабатываются контроллером ClusterTableController. FXML показан на рисунке 3.14. Пример работы – на рисунке 3.15.

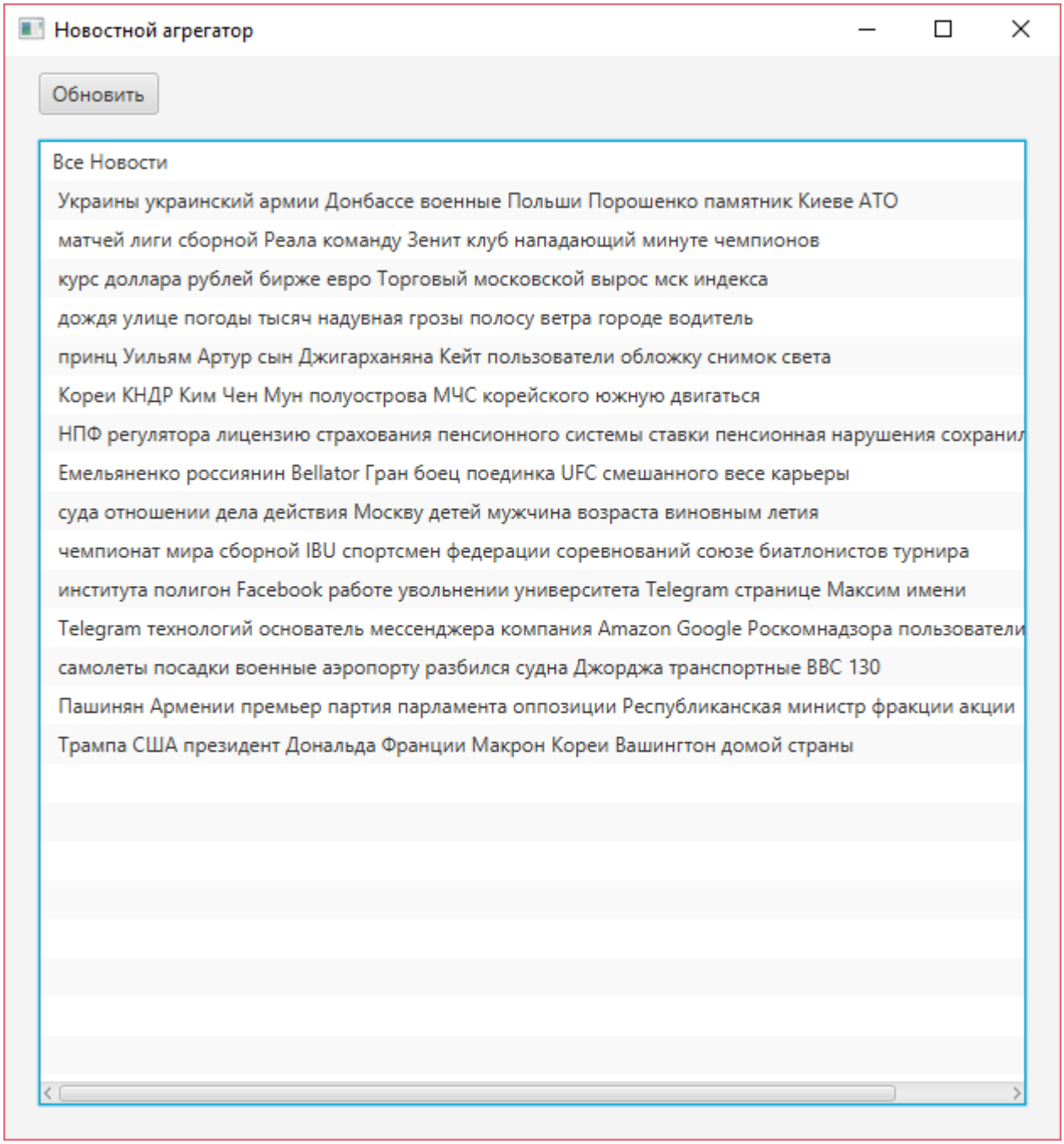

Рисунок 3.13 – Пример работы сцены всех кластеров

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<AnchorPane prefHeight="641.0" prefWidth="624.0"
            xmlns="http://javafx.com/javafx/8.0.121"
            xmlns:fx="http://javafx.com/fxml/1"
            fx:controller="application.ClusterTableController">
   <children>

      <Button onAction="#backButtonPressed"
              text="Back" AnchorPane.leftAnchor="20"
              AnchorPane.topAnchor="10" />

      <TextField fx:id="titleFilterTextField"
                 promptText="Фильтрация заголовка"
                 AnchorPane.leftAnchor="20"
                 AnchorPane.rightAnchor="20"
                 AnchorPane.topAnchor="55" />

      <TableView fx:id="clusterTableView"
                 onKeyPressed="#onKeyPressed"
                 AnchorPane.bottomAnchor="20"
                 AnchorPane.leftAnchor="20"
                 AnchorPane.rightAnchor="20"
                 AnchorPane.topAnchor="85">
          <columnResizePolicy>
           <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
        </columnResizePolicy>
      </TableView>

      </children>
</AnchorPane>
```

Рисунок 3.14 – Сцена содержания кластера в виде FXML

Одновременно эта же страница отвечает за показ всех новостей сразу. По логике они отличаются лишь показом одного столбца, а именно – столбца расстояния от центроида. Пример показа всех новостей представлен на рисунке 3.16.

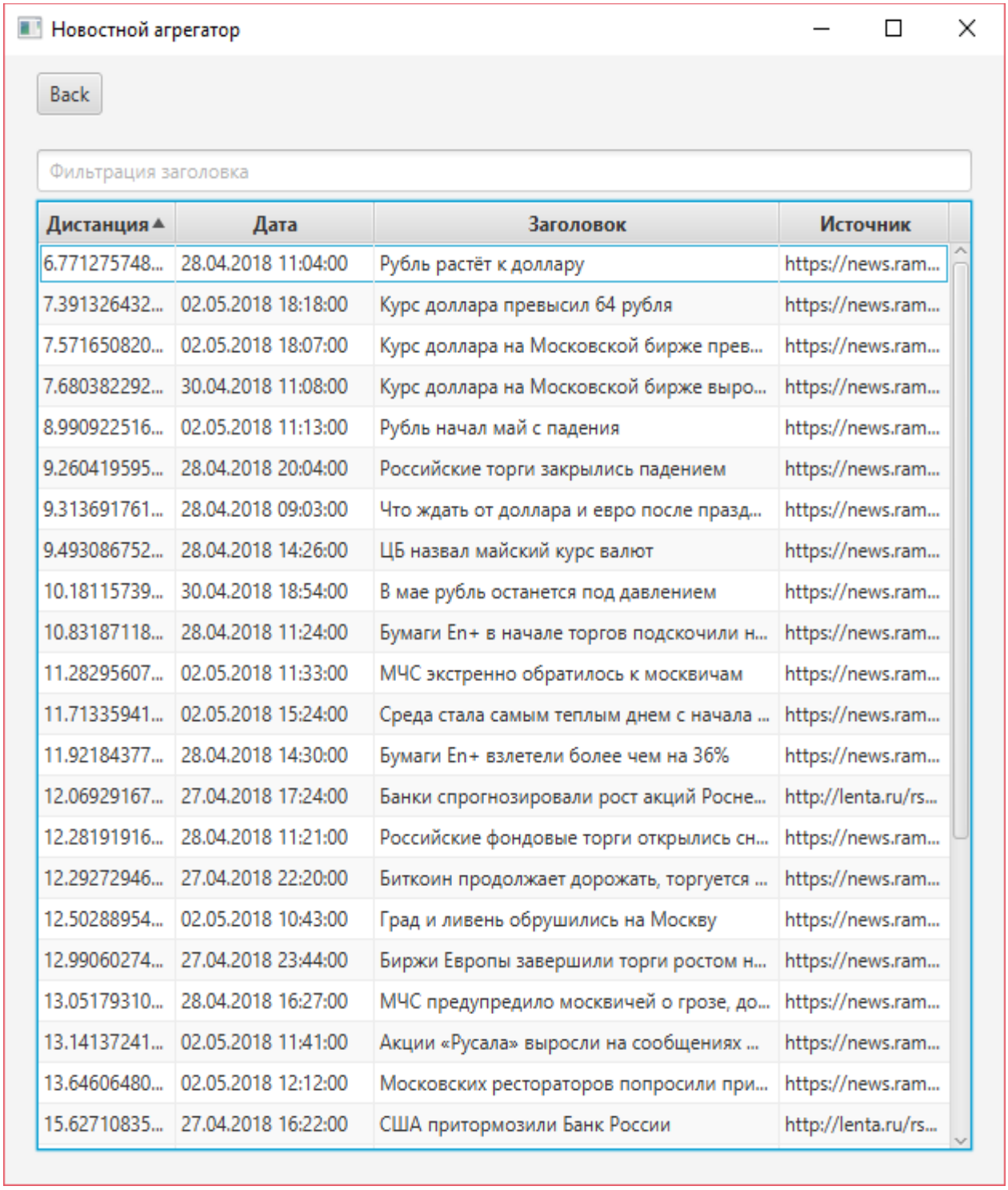

Рисунок 3.15 – Пример работы сцены содержания кластера

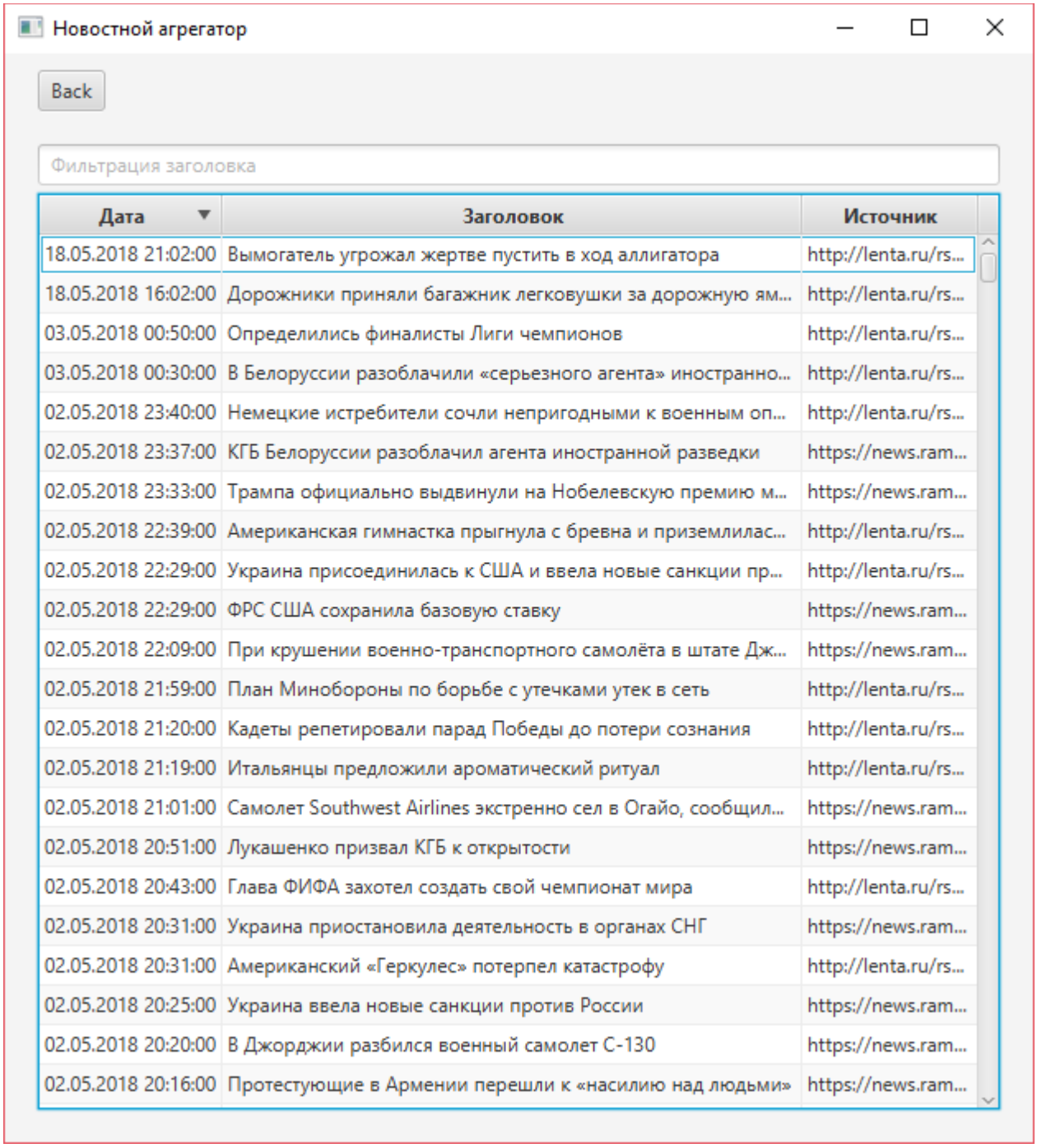

Рисунок 3.16 – Пример работы сцены содержания кластера для всех новостей

Обе сцены в качестве панели (менеджера компоновки) используют AnchorPane. Он позволяет задать расположение дочерних элементов относительно родительских [20]. Помимо мыши, управлять и выполнять события нажатия можно клавиатурой с помощью стрелочек и клавиши «Enter».

ЗАКЛЮЧЕНИЕ

В данной работе были изучено применение кластеризации на основе векторных моделях в новостных агрегаторах для разделения новостей на темы. Были изучены существующие новостные агрегаторы, были сформулированы их достоинства и недостатки. Для решения некоторых недостатков было предложено использовать кластеризацию новостей за счёт представления их в виде векторов с помощью векторных моделей и наименование полученных кластеров.

Поставленные задачи были выполнены и цель достигнута. Формирование тем было выполнено с помощью алгоритма кластеризации k-средних++. Были сравнены три векторных моделей для выполнения кластеризации, и выбрана самая эффективная. Для наименования векторов была выбрана модель на основе весов в векторе центроида и восстановлении слов по терминам. Был разработан новостной агрегатор с применением данного подхода.

Реализованный новостной агрегатор подтвердил проведённый анализ на практике. Он оказался способным выделять узкие для текущей коллекции новостей темы. Имена кластеров хорошо отражают их содержание. В качестве недостатков полученного агрегатора можно выделить ошибки кластеризации: не всегда новость принадлежит кластеру, и не всегда кластер хорошим образом отражает ту или иную тему.

В качестве дальнейшего развития данного подхода можно улучшать работу кластеризации либо дополнять полученное программное обеспечение другими функциями. Основные направления в первом случае: нахождение оптимального количества кластеров для алгоритма k-средних, разработка более эффективной метрики, применение других методов представления документов в виде векторов.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Г Кристофер Д. Маннинг, Прабхакар Рагхаван, Хайнрих Шютце. Введение в информационный поиск. : Пер. с англ. – М. : ООО “И.Д. Вильямс”, 2014 – 528 с
2. Климов В.С. Применение нейросетевых технологий распознавания образов для диагностики контактной сварки в производственной среде / В.С. Климов, А.В. Комиренко // Сварка и диагностика. 2013. № 2. С. 40-44.
3. Климов В.С. Диагностика контактной точечной сварки с использованием нейронной сети Хемминга. Часть. 1. Измерение сварочного сопротивления / В.С. Климов, А.С. Климов, А.К. Кудинов // Вестник машиностроения. 2016. № 10. С. 42-46.
4. Климов В.С. Контактной точечной сварки с использованием нейронной сети Хемминга. Часть. 2. Моделирование нейронной сети / В.С.208 Климов, А.С. Климов, А.К. Кудинов // Вестник машиностроения. 2016. № 11. С. 32-35.
5. David Arthur, Sergei Vassilvitskii, k-means++: the advantages of careful seeding, Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, January 07-09, 2007, New Orleans, Louisiana
6. Roul, R.K., Devanand O.R., Sahay S.K.: Web document clustering and ranking using Tf-Idf based Apriori Approach. In: IJCA Proceedings on International Conference on Advances in Computer Engineering and Applications, 2014
7. J. Ramos. Using tf-idf to determine word relevance in document queries. In First International Conference on Machine Learning, New Brunswick:NJ, USA, 2003. Rutgers University.
8. I. Arroyo-Fernández. Unsupervised Sentence Representations as Word Information Series: Revisiting TF-IDF, Mexico, UNAM, 2017.
9. C. De Boom, S. Van Canneyt, S. Bohez, T. Demeester, B. Dhoedt, "Learning semantic similarity for very short texts", Data Mining Workshop (ICDMW) 2015 IEEE International Conference on IEEE, 2015.

10. Шилдт, Герберт Java 8. Руководство для начинающих / Герберт Шилдт. - М.: Вильямс, 2015. - 720 с.

11. Эккель, Брюс Философия Java / Брюс Эккель. - М.: Питер, 2016. - 809 с.

12. Арун Гупта. Java EE 7. Основы = Java EE 7 Essentials. – М.: «Вильямс», 2014. – 336 с.

13. Vohra D. Java 7 JAX-WS Web Services // D. Vohra, Birmingham: Packt Publishing, 2012. – 64 p.

14. Кнут Дональд Э. Искусство программирования. Т. 3. Сортировка и поиск / Кнут Дональд Э. – 2-е изд.: Пер. с англ. – М.: Вильямс, 2005. – 824 с.

15. Krochmalski J. IntelliJ IDEA Essentials // J. Krochmalski. – Birmingham: Packt Publishing, 2014.-263 p.

16. Hudson O. Getting started with IntelliJ IDEA // O. Hudson, Birmingham: Packt Publishing, 2013. – 114 p.

17. Савитч, Уолтер Язык Java. Курс программирования / Уолтер Савитч. - М.: Вильямс, 2015. - 928 с.

18. Vos J. Pro JavaFX 8: A Definitive Guide to Building Desktop, Mobile, and Embedded Java Clients // J. Vos, W. Gao, J. Weaver, S. Chin, D. Iverson, New York: Apress, 2014. – 616 p.

19. Dea C. JavaFX 8: Introduction by Example // C. Dea, G. Grunwald, J. Pereda, M. Heckler, S. Phillips, New York: Apress, 2014. – 420 p.

20. Pevac I. Practicing Recursion with JavaFX // I. Pevac, Seattle: CreateSpace, 2018 – 168 p.

ПРИЛОЖЕНИЕ А

Исходный код кластеризации методом k-means++

```
// метод для кластеризации
public ClusteringResult performClustering(List<DocumentVector> points,
                                          int clustersSize,
                                          int numTrials,
                                          int titleNameSize, Map<String,
Integer> termsSpaceMap,
                                          Map<String, String> termsToWords)  {
    List<ClusterModel> clusters = new ArrayList<>(clustersSize);

    // кластеризация
    KMeansPlusPlusClusterer<DocumentVector> kMeansPPSingle =
            new KMeansPlusPlusClusterer<DocumentVector>(clustersSize, -1,
distanceMeasure);
    MultiKMeansPlusPlusClusterer<DocumentVector> kMeansMulti =
            new MultiKMeansPlusPlusClusterer<DocumentVector>(kMeansPPSingle,
numTrials);

    List<CentroidCluster<DocumentVector>> centroids =
kMeansMulti.cluster(points);

    double rss = 0;


    // обработка результатов
    for(CentroidCluster<DocumentVector> centroidCluster : centroids) {
        List<DocumentVector> centroidPoints = centroidCluster.getPoints();
        double[] centroid = centroidCluster.getCenter().getPoint();

        ArrayList<ItemModel> docs = new ArrayList<>(centroidPoints.size());

        for(DocumentVector docVec : centroidPoints) {
            double[] point = docVec.getPoint();
            double dist = distanceMeasure.compute(centroid, point);
            NewsDocument doc = docVec.getDocument();
            ItemModel clusterItem = new ItemModel(doc.getTitle(),
                    doc.getSource(),
                    doc.getDate(),
                    dist,
                    doc.getUrl());

            docs.add(clusterItem);

            double euclidDist = distanceMeasure.compute(centroid, point);
            rss += euclidDist;
        }

        // сортировка по расстоянию от центроида
        Collections.sort(docs, clusterResultItemComparator);

        ClusterModel model = new ClusterModel();
        model.setItemModels(docs);
        model.setName(generateName(centroid, termsSpaceMap,
                termsToWords, titleNameSize));
        clusters.add(model);
    }

    return new ClusteringResult(clusters, rss);
}
```