МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математического обеспечения вычислительных комплексов и
информационных систем

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ МОНИТОРИНГА ТРЕВОЖНОСТИ В НОВОСТЯХ

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студентки 2 курса 273 группы
направления 02.04.03 — Математическое обеспечение и администрирование
информационных систем
факультета КНиИТ
Кожиной Ольги Олеговны

Научный руководитель
д. ф.-м. н., профессор _____ Д. К. Андрейченко

Заведующий кафедрой
д. ф.-м. н., профессор _____ Д. К. Андрейченко

Саратов 2021

ВВЕДЕНИЕ

Актуальность темы. Пандемия изменила до неузнаваемости привычный образ жизни: всеобщая самоизоляция наложила право вето на непосредственное, живое общение, которое совершенно необходимо людям. Стремясь следить за последними событиями, пользователи читают новости, чье содержание создает некую психологическую нагрузку. Так, изо дня в день, люди подвергаются влиянию новостных каналов, которые в свою очередь истолковывают произошедшие события, применяя эмоциональную окраску, а сами пользователи становятся более или менее тревожными. Очевидно, что в начале самоизоляции люди были более встревожены, чем в конце первой волны, однако количественное измерение тревожности является сложной и новой задачей.

Таким образом, можно говорить о создании «термометра» новостей, с помощью которого можно отслеживать «градус тревожности» в интернете.

Цель магистерской работы — создание приложения для мониторинга тревожности в новостях.

Поставленная цель определила **следующие задачи**:

1. изучить теорию построения ETL-систем;
2. рассмотреть анализ тональности текста;
3. выбрать и описать подходящую математическую модель для анализа тональности;
4. изучить трехуровневую архитектуру;
5. спроектировать архитектуру приложения;
6. выбрать программные продукты для реализации приложения и описать их достоинства;
7. построить ETL-систему;
8. реализовать модель машинного обучения;
9. разработать уровни представления и бизнес-логики.

Методологические основы разработки приложения для мониторинга тревожности в новостях представлены в работах Е. Большаковой, К. Воронцова, Н. Ефремовой, а также в работах Р. Кимбалла, Х. Карау, П. Венделла, Э. Конвински и М. Захария.

Теоретическая и/или практическая значимость магистерской работы. Количественное измерение тревожности в новостях является сложной

и новой задачей, ранее никаких программных продуктов для отслеживания «градуса напряженности» в интернете создано не было. Также применение системы мониторинга новостей может помочь в создании комфортной операционной обстановки на рабочих местах.

Структура и объем работы. Магистерская работа состоит из определений, введения, двух разделов, заключения, списка использованных источников и десяти приложений. Общий объем работы — 84 страницы, из них 53 страницы — основное содержание, включая 28 рисунков, цифровой носитель в качестве приложения, список использованных источников — 38 наименований.

1 Краткое содержание работы

1.1 Теория построения системы анализа новостей

Первый раздел магистерской работы посвящен трем важным темам для построения приложения мониторинга тревожности в новостях:

1. построение ETL-систем;
2. анализ тональности документа;
3. трехуровневая архитектура.

ETL (Extract, Transform, Load) — это общий термин для всех процессов миграции данных из одного источника в другой. Всякая ETL-система вне зависимости от назначения должна обеспечивать выполнение трех основных функций:

- извлечение — это процесс извлечения данных из одного или нескольких источников, а также их загрузка в промежуточную область и подготовка к преобразованию (проверка данных на соответствие спецификациям и др.);
- трансформация данных — преобразование форматов и кодировки. Кроме того, на этом этапе может происходит очистка, а также анализ данных (например, агрегация);
- загрузка данных — запись преобразованных данных, включая метаданные, в целевую базу данных или хранилище данных.

В архитектурах прикладных систем ETL обычно стоит между OLTP- и OLAP-системами. OLTP-система (Online Transaction Processing) ориентирована на потоковую обработку транзакций небольшого размера в режиме реального времени, выполняет большое количество задач за короткое время, но медленно работает со сложными аналитическими запросами. На быстрой обработке сложных интерактивных аналитических запросов специализируются OLAP-системы (Online Analytical Processing). С помощью OLAP-куба можно ответить на потенциально любые количественные и пространственно-временные запросы [1].

ETL часто воспринимается как способ перенести данные из различных источников в единое КХД. Корпоративное хранилище данных (КХД, DWH — Data Warehouse) — это предметно-ориентированная база данных, предназначенная для подготовки отчётов, интегрированного бизнес-анализа и оптимального принятия управленческих решений на основе данных.

Следует иметь в виду, что некоторые проблемы нельзя решить автоматически: разработчики должны самостоятельно выбирать источники данных, контролировать разрозненность конечных данных, а также отслеживать появление новых форматов представления данных.

Анализ тональности документа. С развитием социальных сетей и, в частности, рекомендательных систем, интегрированных в различные сервисы, автоматический анализ тональности текстов становится одной из самых актуальных технологий обработки текстов.

Рассмотрим основные подходы к анализу тональности текстов:

- методы, основанные на словарях и правилах, используют специально созданные словари оценочных слов и выражений, а также учитывают контекст употребления слов с помощью лингвистических правил;
- методы на основе машинного обучения — используемые в данной работе;
- гибридные методы, в которых в машинном обучении используются оценочные словари [2].

В данной работе каждая новость будет разделена на классы «позитивная» и «негативная», соответственно, для анализа тональности текста будет решаться задача классификации с помощью логистической регрессии. В искусственном интеллекте и машинном обучении классификация — задача разделения множества наблюдений (объектов) на группы, называемые классами, на основе анализа их формального описания.

В математической статистике логистическая регрессия является широко используемой статистической моделью, которая использует логистическую функцию для моделирования зависимости выходной переменной от набора входных. Предположим, что есть только два класса, «positive» (с условным обозначением $+$) и «negative» ($-$). Тогда вероятность того, что некоторое значение принадлежит классу $+$ равна P_+, классу $-$ равна $P_- = 1 - P_+$. Поэтому результат логистической регрессии находится в интервале $[0, 1]$.

Для создания системы автоматического анализа тональности с помощью машинного обучения с учителем отбирается некоторое количество текстов для обучения классификатора, которые вручную размечаются по тональности, затем выбирается алгоритм классификации и признаки, в виде которых будет представлен текст для этого алгоритма. После этого производится обучение классификатора.

В качестве вычислительной системы выбран Apache Spark — всё-в-одном для работы с большими данными. С помощью движка Spark можно в один проход загрузить данные при помощи SQL-запроса, а затем оценить их при помощи машинного обучения на Spark ML. Дополняет Spark библиотека SparkMLlib, реализующая конвейеры машинного обучения, которые можно сохранить и переиспользовать. Основные абстракции SparkMLlib: Dataframe, Transformer, Estimator, Pipeline, PipelineModel [3].

Трехуровневая архитектура — это широко применяемая архитектура программного обеспечения, в которой приложения разделены на три логических и физических уровня:

- уровень представления: обеспечивается взаимодействие с пользователем приложения — это может происходить в веб-браузере или в графическом пользовательском интерфейсе приложений;
- уровень приложения: информация с уровня представления обрабатывается с помощью бизнес-логики. Также промежуточный уровень может работать с уровнем данных посредством вызовов API — добавлять, удалять и изменять данные;
- уровень данных: предназначен для хранения и управления информацией, обработанной приложением. Напрямую уровень данных и уровень представления не взаимодействуют, только с использованием посредника в виде уровня приложения.

Логическое и физическое разделение функциональных возможностей является главным преимуществом трехуровневой архитектуры: уровни можно создавать, настраивать и оптимизировать независимо друг от друга [4].

Общение уровня бизнес-логики и уровня представления в данной работе организовано с использованием REST (REpresentational State Transfer) — это архитектура, т. е. принципы построения распределенных гипермедиа систем, включая универсальные способы обработки и передачи состояний ресурсов по протоколу HTTP. Придумал идею и термин Рой Филдинг в 2000 году. REST на сегодняшний день практически вытеснил все остальные подходы, в том числе дизайн, основанный на протоколе SOAP.

Чтобы упростить ускорить разработку web-приложений, существует множество фреймворков с библиотеками шаблонов. Одним из таких фреймворков является Play Framework — платформа разработки с открытым кодом, написан-

ный на Scala и Java, использует паттерн проектирования Model-View-Controller (MVC).

MVC впервые был реализован в языке Smalltalk. Перед разработчиками стояла задача разделения бизнес-логики, данных и графического интерфейса, поэтому изначально MVC состоял из трех частей, давших ему название:

— Модель — это бизнес-логика приложения; она обладает знаниями о себе самой и не знает о контроллерах и представлениях.

— Представление отвечает за отображение данных от модели и иногда может иметь код с бизнес-логикой внутри.

— Контроллер перехватывает событие извне и, в соответствии с заложенной в него логикой, реагирует на это событие, изменяя модель посредством вызова соответствующего метода. Модель отсылает сообщение о том, что в ней произошли изменения, и все представления, получившие сообщение, обращаются к модели за обновленными данными, а затем отображают их.

1.2 Разработка приложения для мониторинга тревожности в новостях

Второй раздел магистерской работы посвящен реализации самого приложения. Общая архитектура приложения представлена на схеме 1.

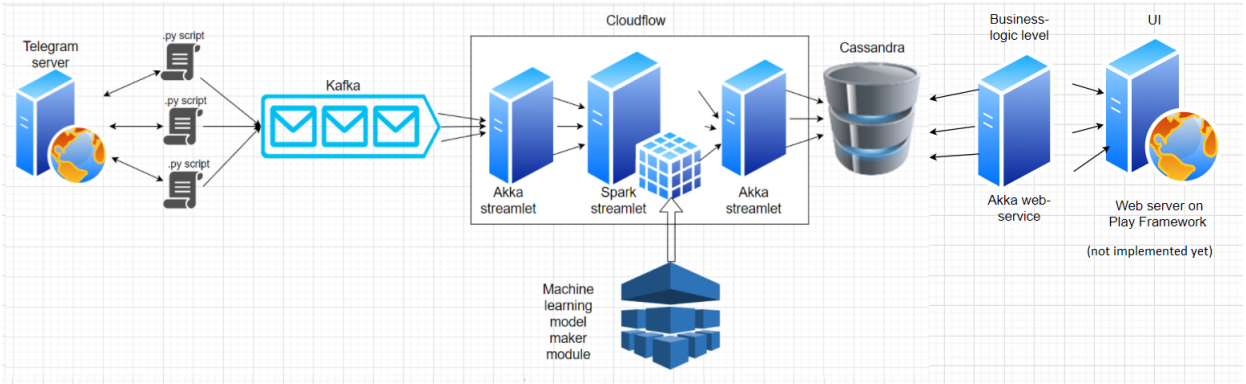

Рисунок 1 – Архитектура приложения

Таким образом, на рисунке изображена архитектура приложения, в котором:

1. новостным ресурсом является Telegram, а именно новостные каналы в Telegram;
2. новости запрашиваются с помощью Telegram-бота, написанного на языке python, который можно запустить в нескольких экземплярах;

3. полученные сообщения отправляются в брокер сообщений Kafka;
4. из Kafka новости вычитываются Akka-стримлетом на платформе Cloudflow с использованием коннектора Alpakka. Все три продукта написаны компанией Lightbend на языке Scala, соответственно, код стримлета тоже реализован на Scala;
5. полученные новости анализируются Spark-стримлетом с использованием уже обученной модели;
6. другой Akka-стримлет записывает новости в базу данных Cassandra;
7. модель на Spark обучается на размеченных данных в отдельном микросервисе;
8. появился модуль построения модели машинного обучения;
9. добавился стримлет на Spark, который принимает на вход новость и отправляет на выход новость с оценкой тональности, с помощью модуля, описанного выше;
10. чтение из Kafka и запись в Cassandra разбиты по отдельным стримлетам;
11. добавился уровень бизнес-логики, соединяющий UI и базу данных Cassandra;
12. появился уровень представления, отвечающий за визуализацию данных.

Telegram является крупной медиаплощадкой, позволяющей легко разрабатывать собственные приложения на её основе. Один бот заменяет множество нетривиальных парсеров в случае использования сайтов новостных агенств, дает больше метаинформации и экономит время разработчика.

В качестве новостных ресурсов было выбрано 23 канала ($rian\_ru$, $oldlentach$, $eurosportru$, $bbcrussian$, $tass\_agency$, $radiosvoboda$, $thebell\_io$, $aavst55$, $varlamov\_news$, $brodetsky$, $nplus1$, $russica2$, $maester$, $olegderipaska$, $margaritasimonyan$, $eschulmann$, $tinkoffjournal$, $radiogovoritmsk$, $dohod$, $dr\_alex\_sosnowski$, $kashinguru$, $zhirinovskylive$, $robabayan$), из них 6 относится к официальным новостным агенствам, 2 канала политиков, 10 каналов журналистов, 5 каналов информационных аполитических агенств.

В подразделе «Написание Telegram-бота» описывается разработка скрипта, позволяющего считывать сообщения из публичных каналов с помощью библиотеки Telethon [5] на языке Python. Сообщения выгружаются в Kafka — это распределенный горизонтально масштабируемый отказоустойчивый журнал коммитов [6]. Написанный скрипт, запущенный в 3 экземплярах, позволяет

выгрузить в Kafka около 40 тысяч сообщений в минуту. В приложении Kafka играет роль посредника между Telegram-ботом и стримлетами на Cloudflow.

В подразделе «Akka-стримлет на Cloudflow с использованием Alpakka» описываются выбранные технологии, приводятся примеры кода с использованием фреймворков, описывается структура разработанного проекта. Выбранные для написания основной части приложения технологии (Scala + Cloudflow + Akka + Alpakka) позволяют построить высоконагруженное приложение для потоковой обработки данных.

Платформа Cloudflow позволяет быстро разрабатывать и управлять распределенными потоковыми приложениями в Kubernetes. Cloudflow позволяет легко разбить потоковое приложение на более мелкие составляющие и связать их вместе с помощью контрактов на основе схемы. Cloudflow интегрируется с популярными потоковыми движками, такими как Akka, Spark и Flink [7].

Akka — это фреймворк для двух языков, работающих под JVM, Java и Scala, предоставляющий готовую среду для построения высоконагруженных систем, которые эффективно используют ресурсы машины, позволяет строить отказоустойчивых реактивных систем «из коробки» и описывать логику для потоковой обработки данных [8].

Alpakka — это open-source фреймворк на основе Akka Streams, который является набором коннекторов — множеством библиотек, позволяющих читать из и писать в различные базы данных, облачные хранилища, брокеры сообщений и т.д. Alpakka является коннектором к более чем 50 ресурсам [9].

В итоге в ETL-системе было создано 3 стримлета: первый Akka Streamlet, вычитывающий сообщения из Kafka с помощью коннектора Alpakka, второй Spark Streamlet, использующий предобученную и сохраненную модель машинного обучения, третий Akka Streamlet, сохраняющий оцененные новости в базу данных Cassandra с помощью коннектора Alpakka. Главным критерием выбора Cassandra как базы данных была именно высокая скорость записи, позволившая за 5 минут загрузить больше 120 тысяч строк.

В подразделе «Шаги построения модели» описываются необходимые шаги для построения модели машинного обучения вместе с участками написанного кода:

1. разметка данных — для классификации использовались размеченные данные, взятые с соревнования «Sentiment Analysis in Russian» с платформы

kaggle [10];

2. загрузка данных — на этом шаге происходит загрузка данных в DataFrame, с фильтрацией, разбиением на классы и отделением тестовых данных;
3. подготовка данных для извлечения признаков — на этом шаге происходит токенизация, нормализация, стеммизация и лемматизация данных при помощи библиотеки mystem от Яндекса;
4. извлечение признаков — применение TF-IDF к данным;
5. применение логистической регрессии и составление пайплайна — на этом шаге новость классифицируется как позитивная или негативная;
6. обучение модели;
7. тестирование модели — точность модели составила 86%;
8. сохранение модели — модель сохраняется на диск для дальнейшего использования.

Следующий подраздел описывает код, позволяющий использовать сохраненную модель в пайплайне Cloudflow.

Подраздел «Разработка с использованием Akka HTTP» описывает создание web-сервиса, обрабатывающего GET-запрос с параметром для создания необходимой агрегации данных из Cassandra и отвечающего по REST UI-сервису. Akka HTTP — это модуль Akka, разработанный на основе библиотек akka-actor и akka-stream, который представляет собой набор инструментов для разработки или использования сервисов на основе HTTP. Akka HTTP предоставляет DSL (англ. domain specific language, язык, специфический для предметной области) для описания HTTP-маршрутов и способов использования этих маршрутов. Каждый маршрут (route) разделен на более мелкие части, называемые директивой (directive) для обработки определенного запроса.

Подраздел «Разработка на Play Framework» посвящен созданию UI-сервиса, взаимодействующего с уровнем бизнес-логики и отвечающего за визуализацию данных. В подразделе подробна описана структура проекта и назначение основных модулей вместе с примерами кода.

Подраздел «Использование Google Charts» описывает инструмент отрисовки графиков Google Charts и его интеграцию в приложение для визуализации данных.

ЗАКЛЮЧЕНИЕ

В настоящей работе были даны теоретические сведения о построении ETL-систем, а также было показано, как построить Extract-Transform-Load систему на основе новостных каналов. Работа содержит теоретические сведения об анализе тональностей текстов, были рассмотрены классификация, логистическая регрессия, а также программный продукт Apache Spark и библиотека SparkMLlib. В третьем разделе теории было уделено вниманию трехуровневой архитектуре, взаимодействию по REST и подробно разобран паттерн MVC.

Другие поставленные задачи: реализовать уровни бизнес-логики и представления, спроектировать и разработать ETL-систему, обучить и встроить модель машинного обучения в приложения — были достигнуты.

Таким образом, были разработаны следующие программы:

- написан Telegram-бот, вычитывающий новости из различных Telegram-каналов и отправляющий их в брокер сообщений Kafka;
- разработана ETL-система на платформе Cloudflow, а именно 3 стримлета: Akka Streamlet, вычитывающий сообщения из Kafka с помощью коннектора Alpakka, Spark Streamlet, использующий предобученную и сохраненную модель машинного обучения, второй Akka Streamlet, сохраняющий оцененные новости в базу данных Cassandra с помощью коннектора Alpakka;
- реализована модель машинного обучения на Spark с использованием библиотеки SparkMLlib, анализирующая тональность новости с точностью в 86%;
- построен web-сервис на Akka, вычитывающий новости из Cassandra для получения агрегированной информации и отвечающий на запросы UI-сервиса;
- написан UI-сервис на Play Framework, способный строить графики по данным, запрошенным в web-сервисе.

В результате работы было построено приложение, полностью отвечающее целям работы.

Отдельные части магистерской работы были представлены на научной конференции студентов факультета КНиИТ 30.04.2021 года, а именно был представлен доклад по теме «Построение ETL-системы на основе новостных каналов».

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Kimball, R.* The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data / R. Kimball, J. Caserta. — John Wiley and Sons, 2004. — P. 528.

2 *Большакова, Е.* Автоматическая обработка текстов на естественном языке и анализ данных / Е. Большакова, К. Воронцов, Н. Ефремова, Э. Клышинский, Н. Лукашевич, А. Сапин. — НИУ ВШЭ, 2017. — С. 269.

3 *Карау, Х.* Изучаем Spark. Молниеносный анализ данных / Х. Карау, П. Венделл, Э. Конвински, М. Захария. — ДМК-Пресс, 2015. — С. 304.

4 *Фаулер, М.* Архитектура корпоративных программных приложений / М. Фаулер. — Вильямс, 2007.

5 Telethon's Documentation [Электронный ресурс]. — URL: https://docs.telethon.dev/en/latest/ (Дата обращения 31.05.2021). Загл. с экр. Яз. англ.

6 Kafka Documentation [Электронный ресурс]. — URL: https://kafka.apache.org/documentation/ (Дата обращения 31.05.2021). Загл. с экр. Яз. англ.

7 Cloudflow Docs - version 2.0.13. Introducing Cloudflow [Электронный ресурс]. — URL: https://cloudflow.io/docs/current/index.html (Дата обращения 31.05.2021). Загл. с экр. Яз. англ.

8 Akka Documentation [Электронный ресурс]. — URL: https://akka.io/docs (Дата обращения 31.05.2021). Загл. с экр. Яз. англ.

9 Alpakka Documentation [Электронный ресурс]. — URL: https://doc.akka.io/docs/alpakka/current/ (Дата обращения 31.05.2021). Загл. с экр. Яз. англ.

10 Sentiment Analysis in Russian [Электронный ресурс]. — URL: https://www.kaggle.com/c/sentiment-analysis-in-russian/data (Дата обращения 31.05.2021). Загл. с экр. Яз. англ.