Пермский филиал федерального государственного автономного
образовательного учреждения высшего образования
Национальный исследовательский университет
«Высшая школа экономики»

Факультет социально-экономических и компьютерных наук

Соломатин Роман Игоревич

РАЗРАБОТКА СИСТЕМЫ ДЛЯ АВТОМАТИЧЕСКОГО СБОРА, АНАЛИЗА И ВИЗУАЛИЗАЦИИ ИНФОРМАЦИИ ПО ЭТИЧНОСТИ КОМПАНИЙ

Выпускная квалификационная работа

студента образовательной программы «Программная инженерия»
по направлению подготовки 09.03.04 Программная инженерия

Руководитель
к.т.н., доцент кафедры
информационных технологий в
бизнесе НИУ ВШЭ-Пермь

А. В. Бузмаков

Пермь, 2023 год

Аннотация

Соломатин Р. И. «Разработка системы для автоматического сбора, анализа и визуализации информации по этичности компаний»: выпускная квалификационная работа бакалавра по направлению подготовки 09.03.04 «Программная инженерия». – Пермь, 2023.

Выпускная квалификационная работа посвящена разработке программного обеспечения для построения индекса этичности компаний. Цель работы заключается в создании системы для автоматической оценки этичности компаний. Для этого будут собраны отзывы о компаниях с разных сайтов, разработаны алгоритмы анализа естественного языка, проанализированы отзывы и составлен индекс.

Работа включает: 74 страниц (4 главы), 22 количество, 20 таблиц и 6 приложений. Библиографический список содержит 47 источников.

Оглавление

Введение

Эффективность работы компаний зависит от многих факторов, в том числе и от этики делового поведения и ответственности их сотрудников. Компании, которые придерживаются высоких стандартов этики и интегрируют их в свою культуру, обычно имеют более лояльных клиентов и успешнее конкурируют на рынке[1]. Кроме того, соблюдение этических норм и принципов способствует укреплению репутации компании, что может привести к привлечению талантливых сотрудников и установлению долгосрочных партнерских отношений с другими компаниями и организациями. В целом, этика делового поведения играет важную роль в формировании имиджа компании и ее успеха на рынке.

В данной работе под этикой будет пониматься нацеленность компаний на принятия каких-то действий, которые краткосрочно не обязательно выигрышных для бизнеса, но которые увеличивают лояльность клиентов. Например, у клиента банка задержали зарплату и он не делает платеж по кредиту. Формально банк может по кредитному договору назначить штраф за неисполнение клиентом обязательств, но войдя в положение клиента, банк может не назначить или отменить такой штраф.

В настоящее время существует несколько сервисов, которые призваны оценивать этику компании на основании финансовых показателей[1] и судебных дел[2]. Это привело к ситуации, когда отдельные лица должны проводить много времени для исследования разных компаний, чтобы определить насколько этична компания. Одним из доступных источников оценки этики – отзывы, которые оставляют пользователи и описывают положительный и отрицательный опыт взаимодействия с компанией. Это часто включает в себя просмотр отзывов с различных веб-сайтов, что может занять много времени и не всегда может дать исчерпывающую или точную картину, так как не включает в себя качество обслуживания.

Таким образом из публичной информации можно получить этичность компании, но это занимает много времени.

[1] `https://kontur.ru/expert`, `https://www.esphere.ru/products/spk/financial`

[2] `https://proverki.gov.ru/portal/public-search`

Для решения этой проблемы требуется реализовать систему, которая собирает и анализирует отзывы потребителей с различных веб-сайтов, чтобы дать более полную и точную оценку этической практики компании. Затем собранные данные анализируются с помощью методов обработки естественного языка и машинного обучения, для выявления закономерностей и тенденций, связанных с этической практикой компании. Полученный анализ может быть использован для разработки надежной и достоверной системы оценки этичности компаний.

Объект исследования – оценка этичности компаний.

Предмет исследования – автоматизация оценки этичности компаний на основании отзывов клиентов.

Цель работы – создание системы для автоматической оценки этичности компаний.

Исходя из поставленной цели, необходимо:

1. Провести анализ оценки этичности и требований.
2. Реализовать систему для оценки этичности.
3. Провести тестирование системы.

Этап анализа должен включать:

1. Анализ проблемы предметной области.
2. Анализ требований к системе.
3. Анализ существующих алгоритмов.

Этап проектирования должен включать:

1. Проектирование серверной части.
2. Проектирование модели для определения этичности.
3. Проектирование клиентской части приложения.

Этап реализации должен включать:

1. Сбор данных.
2. Обучение моделей.
3. Реализации серверной части.

Этап тестирования должен включать:

1. Тестирование модели.

2. Тестирование серверной части.

3. Тестирование клиентской части.

В ходе выполнения анализа, проектирования и реализации приложения используется объектно-ориентированный подход. Результаты анализа и решения задач проектирования формализуются с помощью диаграмм UML.

Глава 1 Анализ предметной области

В данной главе представлен аналитический обзор оценок этичности компаний и алгоритмов машинного обучения, а также обзор существующих программных решений для поставленной проблемы.

Анализ предметной области следует разделить на следующие пункты:

1. Анализ процесса определения этичности компаний сейчас позволяет понять, как этот процесс происходит и как его лучше всего автоматизировать.
2. Анализ оценок этичности компаний для того, чтобы в дальнейшем определить этичность компаний.
3. Анализ существующих решений выполняется с целью выделения их сильных и слабых сторон по отношению к решаемой проблеме.
4. Анализ алгоритмов обработки естественного языка позволяет понять с помощью каких алгоритмов можно найти полезную информацию в текстах.
5. Анализ требований к системе позволит выделить функциональные и нефункциональные требования.

1.1. Анализ определения этичности компании

Этичность компаний уже давно вызывает широкий общественный интерес, особенно их поведение в спорных ситуациях и предоставление услуг, ориентированных на клиента. В последние годы все большее внимание уделяется оценке этичности компаний [1, 2, 3], особенно в банковском секторе и через призму экологических, социальных и управленческих факторов (ESG). Необходимость в таких оценках становится все более острой по мере того, как общество продолжает бороться с последствиями неправомерных действий корпораций и более широким воздействием корпоративной деятельности на общество и окружающую среду.

Сейчас процесс поиска этичной компании выглядит следующим образом: сначала ищутся компании, которые предоставляют желаемые услуги. Далее они изучаются, чтобы определить их этичность. Этот процесс включает в себя:

1. Просмотр отчетности компании.

2. Анализ ее финансовой деятельности.

3. Изучение информации о социальной ответственности.

Для этого клиенты компаний обращаются к различным источникам информации, таким как веб-сайты компаний, рейтинговые агентства, исследовательские организации и другие источники. Потом, изучаются социальные сети компании или отзывы пользователей на разных сайтах, форумах и социальных сетях, чтобы получить дополнительную информацию и оценить общее мнение о компании. После изучения каждой компании люди выбирают ту, которую они считают наиболее этичной и социально ответственной. Блок-схема данного поиска рис. 1.1. Важным фактором для определения этичности компании может быть ее социальная ответственность, устойчивость бизнеса и соблюдение норм и стандартов в области финансовой деятельности.

В целом, процесс поиска компаний и определения их этичности может быть длительным и требует серьезного подхода. Люди могут использовать различные источники информации, чтобы сделать осознанный выбор и инвестировать свои деньги в компанию, которая соответствует их ожиданиям и требованиям.

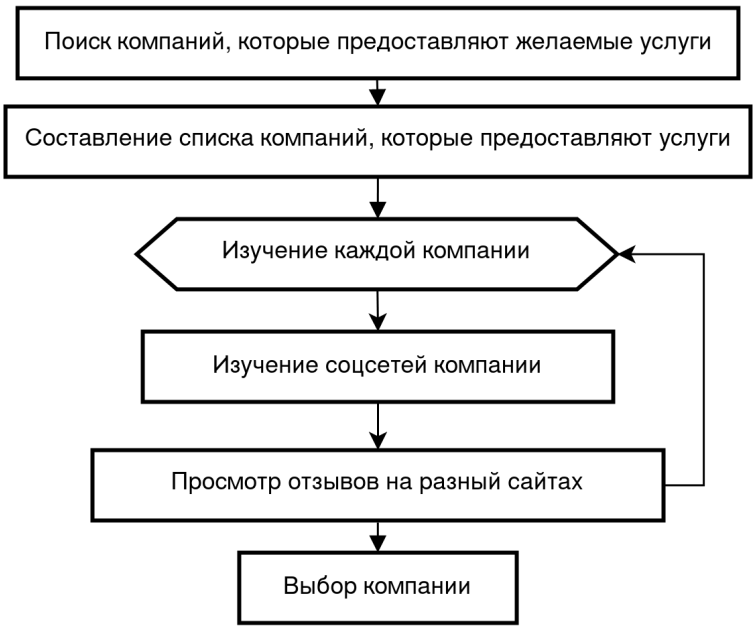


Рисунок 1.1 – Диаграмма того, как сейчас происходит поиск компании

1.2. Анализ оценок этичности компаний

Оценка этики компании – это не одноразовый процесс, а скорее длительный и непрерывный процесс, который позволяет понять и оценить действия, политику и практику компании с течением времени. Оценка включает в себя рассмотрение соблюдения компанией отраслевых этических стандартов и передовой практики, а также мониторинг любых изменений в этической позиции компании с течением времени. Кроме того, участие в диалоге с компанией и консультации с организациями, специализирующимися на оценке корпоративной ответственности могут дать ценную информацию об этических практиках компании.

Компаниям важно оставаться этичными, так как в долгосрочной перспективе это приносит большую прибыль и улучшает показатели бизнеса, чем неэтичный способ ведение бизнеса[4, 1]. Насколько этична компания можно рассматривать с двух сторон, самой компании и их клиентов. Со стороны компаний можно выделить факторы, которые можно получить из их отчетности:

- размер капитала, чтобы они не могли обанкротиться.
- влияние на окружающую среду.
- куда идут инвестиции [5].

Для пользователей одними из ключевых факторов можно выделить:

- качество пользовательского сервиса [6], как правило пользователи оставляют отзывы на сайтах по пятибалльной шкале.
- насколько навязчивы услуги компании [7], как правило пользователи оставляют отзывы на сайтах по пятибалльной шкале.

В данной работе этичность компаний будет определяться по отзывам клиентов, которые могут в которых содержатся проблемы качества услуг и качество сервиса.

1.3. Анализ существующих решений

Существует несколько индексов, предназначенных для измерения этичности – индекс Dow Jones Sustainability Indices (DJSI) [8] и FTSE4GOOD [9].

DJSI оценивает показатели устойчивости компаний различных секторов на основе экономических, экологических и социальных критериев. Компании отбираются

на основе их показателей по сравнению с аналогичными компаниями в том же секторе. Процесс оценки включает в себя тщательную оценку компаний по различным критериям, включая корпоративное управление, экологический менеджмент, трудовую практику, права человека и социальные вопросы.

Аналогичным образом, индекс FTSE4GOOD предназначен для оценки деятельности компаний, которые демонстрируют эффективную практику экологического, социального и управленческого менеджмента (ESG). Компании отбираются на основе их практики ESG и оцениваются по различным критериям, включая изменение климата, права человека и корпоративное управление.

Индексы DJSI и FTSE4GOOD разработаны для того, чтобы помочь инвесторам определить компании, которые привержены этической практике. Эти индексы предоставляют инвесторам стандартизированный способ сравнения компаний на основе их показателей. Это помогает инвесторам принимать более обоснованные инвестиционные решения и побуждает компании внедрять устойчивую практику для привлечения инвестиций.

Для российских компаний нет аналогичных индексов. Сейчас данные об этичности компаний можно получить из агрегаторов отзывов и отчётности. Агрегаторы позволяют собрать информацию о клиентском обслуживании, а отчетность компаний о положении дел в целом. Но сейчас не существует способов, как можно оценить все вместе.

1.4. Анализ требований к системе

Исходя из интервью с заказчиком система должна уметь:

1. Показывать историю изменений индекса с возможностью фильтровать по:
    1. Годам.
    2. Отраслям компаний, с возможностью множественного выбора.
    3. Компаниям, с возможностью множественного выбора.
    4. Моделям, с возможностью множественного выбора.
    5. Источникам, с возможностью множественного выбора.
2. Агрегировать значения индекса по годам и кварталам.

3. Анализировать текстовые отзывы для построения индекса этичности на основании позитивности или негативности отзывов.
4. Иметь возможность добавления анализа текста несколькими вариантами.
5. Сохранять тексты для последующего анализа другими методами.
6. Система должна собирать данные с сайтов banki.ru, sravni.ru и комментарии из групп «вконтаке».

На основе описания функциональных требований была создана диаграмма вариантов использования, которая представлена на рисунке 1.2.

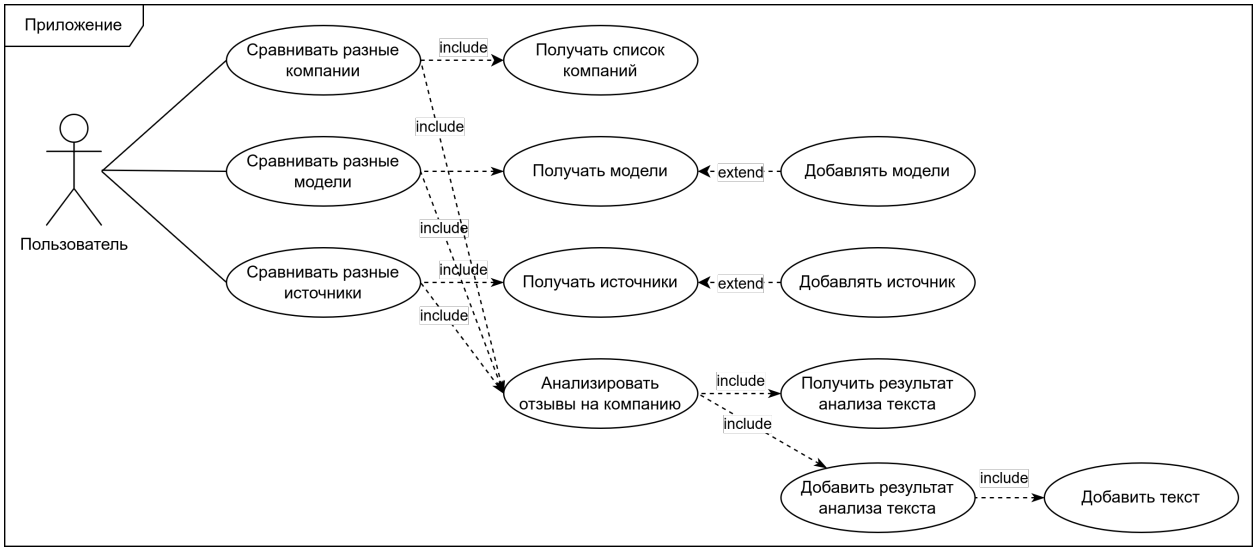


Рисунок 1.2 – Диаграмма вариантов использования

Также были получены нефункциональные требования:

1. Построение графика не должно занимать больше секунды.
2. Сбор данных должен происходить автоматически.
3. Данные должны обрабатываться автоматически.
4. Система должны способна работать с большим объемом информации. Несколько гигабайт текста.
5. Система должна быть стабильной и надежной, обеспечивая непрерывную работу без сбоев или перебоев.

1.5. Анализ метрик классификации

Исходя из собранных требований в данной работе будет решаться задача классификации. Для определения качества работы алгоритма будут рассмотрены несколько метрик:

1. Доля правильных ответов
2. Точность
3. Полнота
4. F-мера

Для лучшего понимания этих метрик рассмотрим матрицу ошибок. Матрица ошибок1.1 является таблицей, которая показывает количество верно и неверно классифицированных примеров для каждого класса. Она состоит из четырех значений:

- Истинно-положительные (True Positives, TP): количество примеров, которые были правильно классифицированы как положительные.
- Истинно-отрицательные (True Negatives, TN): количество примеров, которые были правильно классифицированы как отрицательные.
- Ложно-положительные (False Positives, FP): количество примеров, которые были неправильно классифицированы как положительные.
- Ложно-отрицательные (False Negatives, FN): количество примеров, которые были неправильно классифицированы как отрицательные.

Таблица 1.1 – Матрица ошибок

	y=1	y=0
f(x)=1	True Positives, TP	False Positives, FP
f(x)=0	False Negatives, FN	True Negatives, TN

Доля правильных 1.1 ответов (Accuracy) показывает, как часто модель правильно классифицирует примеры. Она вычисляется как отношение числа верно классифицированных примеров к общему числу примеров:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.1}$$

Точность 1.2 (Precision) показывает, какая доля примеров, классифицированных как положительные, действительно является положительными. Она вычисляется как отношение числа истинно-положительных примеров к сумме истинно-положительных и ложно-положительных примеров:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1.2}$$

Полнота 1.3 (Recall) показывает, какая доля положительных примеров была правильно классифицирована. Она вычисляется как отношение числа истинно-положительных примеров к сумме истинно-положительных и ложно-отрицательных примеров:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{1.3}$$

F-мера 1.4 (F1 Score) является гармоническим средним между точностью и полнотой. Она позволяет учесть оба показателя и оценить баланс между ними. Лучше всего подходит для несбалансированных выборок. F-мера вычисляется по формуле:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1.4}$$

Выбор метрики для определения качества работы алгоритма будет зависеть от данных для ее обучения.

1.6. Алгоритмы для обработки естественного языка

В требованиях было заявлено, что для оценки этичности компаний надо обрабатывать текстовые отзывы, поэтому в данной работе будут рассмотрены алгоритмы по обработке естественного языка.

Алгоритмы машинного обучения для анализа текста получили широкое распространение для извлечения информации из неструктурированных данных с помощью больших помеченных наборов данных. Среди различных используемых методов несколько алгоритмов оказались особенно эффективными в этой области. Каждый из этих алгоритмов обладает уникальными характеристиками, которые делают их хорошо подходящими для определенных задач. В этом разделе будут рассмотрены следующие алгоритмы:

1. Мешок слов
2. TF-IDF
3. Word2Vec
4. FastText
5. BERT

1.6.1. Мешок слов

Мешок слов [10] – метод анализа, который позволяет упрощенно представить текст, как таблицу, где для каждого документа показано количество вхождений слова. В данной модели все слова предстают как множество слов без учёта грамматики и порядка. В процессе обработки текст разбивается на отдельные слова или токены, игнорируя грамматические правила и порядок слов. Затем строится словарь, содержащий все уникальные слова из текстового корпуса. Каждому слову в словаре присваивается уникальный идентификатор. После построения словаря каждый документ представляется в виде вектора, где каждый элемент вектора соответствует слову из словаря, а значение элемента – количество вхождений этого слова в документе. Таким образом, каждый документ представляется в виде разреженного вектора, где большинство элементов равно нулю. Например, для предложений «Мама мыла раму»(1) и «Иван поломал раму»(2) результат работы алгоритма показан в таблице 1.2:

Таблица 1.2 – Пример мешка слов

	мама	мыла	раму	иван	поломал
текст1	1	1	1	0	0
текст2	0	0	1	1	1

1.6.2. TF-IDF

TF-IDF [11] (TF частотность слова (term frequency), IDF – обратная частота документов (inverse document frequency)) – статистический показатель, применяемый для оценки важности слова в контексте документа. Большой вес в TF-IDF получат слова с высокой частотой в пределах конкретного документа и с низкой частотой употреб-

лений в других документах. В процессе TF-IDF текст разбивается на отдельные слова или токены, игнорируя грамматические правила и порядок слов.

TF – частота слова в коллекции. Таким образом, оценивается важность слова t_i по формуле 1.5:

$$TF(t,d) = \frac{n_t}{\sum_k n_k}, \tag{1.5}$$

где n_t – число вхождений слова t в документ, а в знаменателе – общее число слов в данном документе.

IDF – инверсия частоты, с которой некоторое слово встречается в документах коллекции. Чем реже слово встречается в документах, тем выше значение IDF и тем больше важности придается слову в контексте корпуса. Оно рассчитывается по формуле 1.6.

$$IDF(t,D) = \log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|}, \tag{1.6}$$

где $|D|$ – число документов в коллекции, а $|d_i \in D|t \in d_i|$ – число документов из коллекции D, в которых встречается t (когда $n_t \neq 0$).

Таким образом, мера TF-IDF является произведением двух множителей 1.7:

$$\text{TF-IDF}(t,d,D) = TF(t,d) \times IDF(t,d) \tag{1.7}$$

1.6.3. Word2Vec

Word2Vec [12] – это алгоритм обработки естественного языка, который используется для получения векторных представлений (эмбеддингов) слов на основе их семантического контекста. Он работает на основе распределенного представления слов, идея которого состоит в том, что слова, встречающиеся в похожих контекстах, имеют схожие семантические значения. Алгоритм предлагает две основные архитектуры: CBOW (непрерывный мешок слов) и Skip-gram.

1. Непрерывный мешок слов: Архитектура CBOW состоит в обучении модели для предсказания целевого слова на основе контекстных слов. Например, для предложения «Мама мыла раму» модель CBOW пытается предсказать слово «мыла» на основе остальных слов.

2. Skip-gram: Архитектура Skip-gram работает в обратном направлении по сравнению с CBOW. Она предсказывает контекстные слова на основе целевого слова. То есть для слова «мыла» модель Skip-gram пытается предсказать остальные контекстные слова.

Результатом обучения Word2Vec являются векторные представления слов, где каждое слово представлено вектором фиксированной длины. Векторы слов сохраняют в себе семантическую информацию о значениях слов и их семантической близости.

1.6.4. FastText

FastText [13]- это библиотека и метод машинного обучения, разработанный командой Facebook AI Research, для обработки естественного языка. Он является эффективным инструментом для работы с текстовыми данными и создания векторных представлений слов.

FastText расширяет идею Word2Vec, добавляя поддержку для обработки подслов. В отличие от Word2Vec, который работает только на уровне слов, FastText представляет слова как комбинации символьных n-грамм. N-граммы - это последовательности символов фиксированной длины, которые могут быть префиксами, суффиксами или внутренними частями слова.

1.6.5. BERT

BERT [14] (Bidirectional Encoder Representations from Transformers) – это нейросетевая языковая модель, которая относится к классу трансформеров. Она состоит из 12 «базовых блоков» (слоев), а на каждом слое 768 параметров.

На вход модели подается предложение или пара предложений. Затем разделяется на отдельные слова (токены). Потом в начало последовательности токенов вставляется специальный токен `[CLS]`, обозначающий начало предложения или начало последовательности предложений. Пары предложений группируются в одну последовательность и разделяются с помощью специального токена `[SEP]`, затем к каждому токену добавляется эмбеддинг, показывающий к какому предложению относится токен. Потом все токены превращаются в эмбеддинги 1.3 по механизму описанному в работе [15].

При обучении модель выполняет на 2 задания:

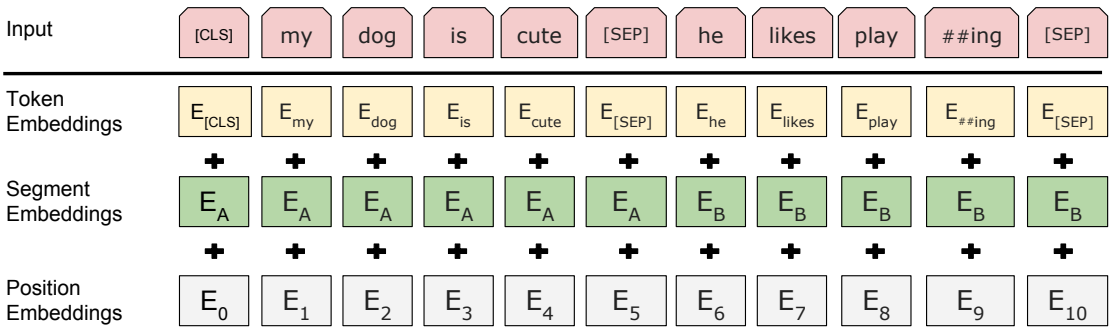


Рисунок 1.3 – Пример ввода текста в модель

1. Предсказание слова в предложении

   Это задание обучается следующим образом – 15% случайных слов заменяются в каждом предложении на специальный токен `[MASK]`, а затем предсказываются на основании контекста. Однако иногда слова заменяются не на специальны токена, в 10% заменяются на случайный токен и еще в 10% заменяются на случайное слово.

   Поскольку стандартные языковые модели 1.4, такие как ELMo [16] и GPT [17], либо смотрят текст слева направо, либо справа налево, они не подходят для некоторых типов заданий. Однако BERT является двунаправленной моделью, что означает, что для каждого слова он может рассмотреть его контекст и использовать эту информацию для предсказания замаскированного слова.

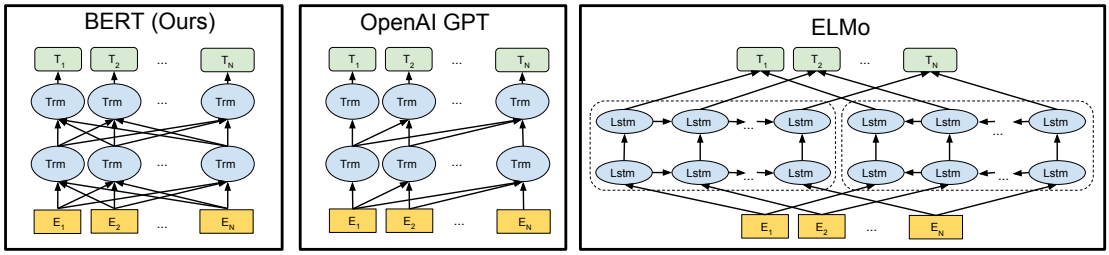


Рисунок 1.4 – Сравнение принципов работы BERT, ELMo, GPT

2. Предсказание следующего предложения

   Для того чтобы обучить модель, которая понимает отношения предложений, она предсказывает, идут ли предложения друг за другом. Для этого с 50% вероятностью выбирают предложения, которые находятся рядом и наоборот. Пример ввода пары предложений в модель 1.5.

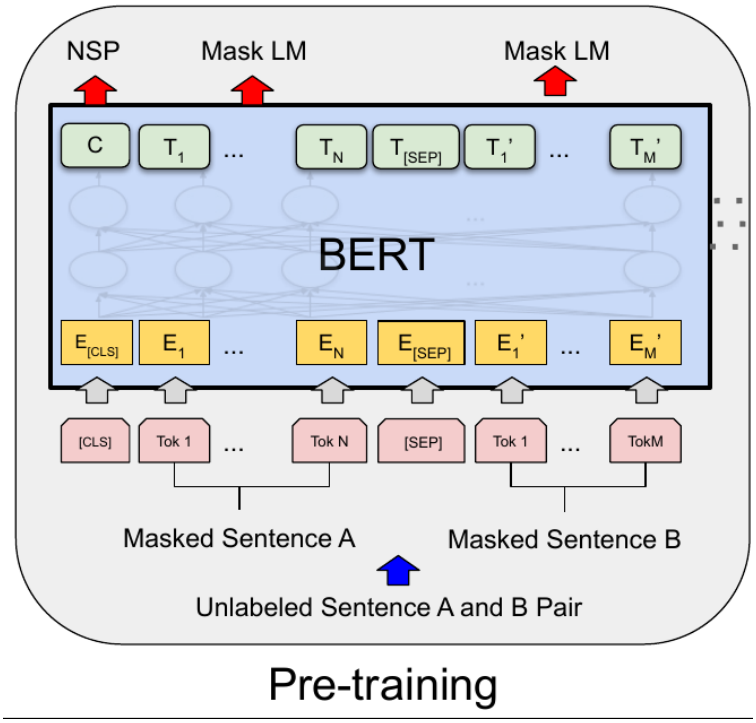


Рисунок 1.5 – Схемам работы BERT

1.6.6. Выводы

В данном разделе были рассмотрены различные алгоритмы для обработки текста. Каждый алгоритм может подойти к той или иной задаче, поэтому в данной работе они все будут рассмотрены.

1.7. Алгоритмы для классификации

Для определения на сколько этична компания, каждый отзыв будет классифицироваться. Для этого будут рассмотрены следующие алгоритмы:

1. Логическая регрессия
2. Метод опорных векторов
3. Случайный лес
4. Градиентный бустинг

1.7.1. Логистическая регрессия

Логистическая регрессия[18] является одним из наиболее распространенных алгоритмов машинного обучения, который применяется для задач классификации. Она

основана на логистической функции 1.8, которая преобразует входные данные в вероятности принадлежности к определенным классам.

$$f(z) = \frac{1}{1 + \exp(-z)}, \tag{1.8}$$

где z – скалярное произведение весов модели, на признаки ответа.

1.7.2. Метод опорных векторов

Метод опорных векторов[19] (Support Vector Machine, SVM) — это алгоритм машинного обучения, который используется для задач классификации и регрессии. Он основан на принципе максимизации зазора (margin) между классами объектов.

Идея SVM заключается в том, чтобы найти оптимальную гиперплоскость, которая разделяет два класса объектов в максимально возможном зазоре. Гиперплоскость представляет собой (n-1)-мерную гиперплоскость в n-мерном пространстве, где n - количество признаков.

Данная модель часто применяется для текстовой классификации[20].

1.7.3. Случайный лес

Случайный лес[21](Random Forest) является ансамблевым методом машинного обучения, который объединяет несколько решающих деревьев для решения задач классификации и регрессии. Каждое дерево строится независимо от других на основе различных подвыборок данных (bootstrap samples) и случайного подмножества признаков. Это позволяет деревьям быть разнообразными и уменьшает корреляцию между ними. На каждом узле дерева выбирается лучший признак и значение порога, на которое данные будут разделены на две ветви. Этот выбор делается на основе критерия информативности, такого как индекс Джини (для классификации) или среднеквадратичная ошибка (для регрессии).

1.7.4. Boosting

Градиентный бустинг[22] (Gradient Boosting) - это алгоритм машинного обучения, который также является ансамблевым методом, но в отличие от случайного леса, градиентный бустинг строит ансамбль слабых моделей (обычно решающих деревьев) последовательно, обучая каждую модель на ошибках предыдущих моделей.

Сначала инициализируется начальная модель, которая может быть простой, например, константой или средним значением целевой переменной. Это начальное предсказание будет постепенно улучшаться в процессе построения ансамбля моделей. Затем строится первая модель. Ошибка между предсказанными значениями первой модели и фактическими значениями вычисляется с помощью функции потерь, такой как среднеквадратичная ошибка (для задач регрессии) или логистическая функция потерь (для задач классификации). После обучения первой модели вычисляется градиент функции потерь по отношению к предсказаниям первой модели. Градиент показывает, в каком направлении и насколько сильно нужно скорректировать предсказания первой модели, чтобы уменьшить ошибку. Далее строится следующая модель, которая обучается на остатках первой модели. Остатки представляют собой разницу между фактическими значениями и предсказаниями первой модели. Вторая модель приближает остатки, чтобы улучшить предсказания

Одни из самых популярных реализаций градиентного бустинга:

1. Scikit-learn – эта реализация предлагает простую реализации градиентного бустинга с удобным интерфейсом.
2. XGBoost[23] – предоставляет много дополнительных возможностей и оптимизаций, которые делают ее одной из наиболее мощных библиотек для градиентного бустинга. Обладает высокой производительностью и позволяет обучаться на GPU[24], что ускоряет процесс.
3. CatBoost[25] – относительно новой библиотекой градиентного бустинга, разработанной компанией Yandex. Он имеет сходные функции и возможности с XGBoost, но также включает в себя некоторые уникальные возможности, которые лучше позволяют обрабатывать параметры, которые отвечают за категориальные признаки. Также имеет поддержку обучения на GPU[26].

1.7.5. Выводы

Все рассмотренные алгоритмы часто используется для анализа текста и не ясно, какой из них справится лучше в посфтавленной задаче. Поэтому в данной работе будут рассмотрены все эти алгоритмы.

1.8. Выбор технологий для разработки

Для реализации этой системы будет использоваться язык Python[27]. Для этого языка разработано много библиотек, которые позволят быстро реализовать алгоритмы обработки естественного языка, в частности в этом проекте будет использоваться Pytorch [28] и HuggingFace [29]. Для реализации API будет использоваться FastAPI, что позволит разработать API для системы с автоматической генерацией документации.

Для хранения данных будет использоваться объектно-реляционная система управления базами данных PostgreSQL, что позволит обрабатывать большие объемы данных. Для работы с ней будет использоваться Code first подход, с помощью Python библиотек Sqlalchemy и Alembic для изменения схемы данных (миграций).

1.9. Выводы по главе

По итогам анализа предметной области, можно сделать вывод о том, что определение этичности компаний является важной задачей, так как с ней сталкиваются многие люди ежедневно и тратят много времени, которую можно автоматизировать с помощью алгоритмов машинного обучения. Обзор существующих решений показал, что сейчас нет индекса, который бы учитывал мнение клиентов для анализа этичности, и может потребоваться разработка нового средства, учитывающего особенности задачи. Наконец, анализ требований к системе позволяет определить необходимые функциональные и нефункциональные требования, которые будут учитываться при разработке решения.

Глава 2 Проектирование системы

В данной главе представлена общая архитектура системы, базы данных и каждого модуля отдельно.

Этап проектирования следует разделить на следующие пункты:

1. Определение основных компонентов приложения и проектирование архитектуры системы.
2. Проектирование базы данных и модулей для работы, обработки, сбора и агрегации данных.
3. Проектирование модели для обработки естественного текста.

Данная глава предоставляет описание системы, каждого компонента и их взаимосвязь в достижении желаемого результата.

2.1. Создание метода для оценки этичности

Для оценки этичности компаний будет рассматриваться на сколько позитивны или негативны отзывы, которые оставили на них. Изначально использовалась разность позитивных и негативных долей среди всех отзывов (базовый индекс, Base index). Данный способ хорошо работает для большого количества отзывов, но для компаний с небольшим количеством отзывов он не будет показывать правдивую картину.

Для этого будет рассмотрено на сколько компания лучше, чем компании в среднем. Для этого будет рассчитано среднее значение – разность позитивных и негативных долей среди всех отзывов за год для каждого источника компании (средний индекс, Mean Index). И рассчитана дисперсия.

Поскольку позитивность и негативность отзыва это дискретная величина, то она будет иметь распределение Бернулли[30]. Дисперсия этого распределения рассчитывается по формуле pq для одной точки, а для последовательность $\frac{pq}{n}$, где p вероятность 1, и q вероятность 0. В данной задаче p будет рассчитываться, как доля класса среди всех отзывов, а q как доля остальных предложений.

В итоге для получения индекса средней компании будет браться абсолютная разность базового индекса и среднего индекса и из этого значения будет вычитаться

дисперсия, что сделает значения индекса более нейтральным и менее подверженным к выбросам. Если значение индекса будет меньше нуля, то нет уверенности в том, что оно будет отлично от 0 и поэтому будет браться максимум из этого значения и 0.

$$\text{Base index} = \frac{\text{positive} - \text{negative}}{\text{total}}$$

$$\text{Std index} = \sqrt{\frac{\text{positive} \cdot \text{(total-positive)}}{total^3} + \frac{\text{negative} \cdot \text{(total - neutral)}}{total^3}}$$

$$\text{Index} = \begin{cases} \max(\text{Base Index} - \text{Mean Index} - \text{Std Index}, 0), & \text{Base index} > \text{Mean index} \\ \min(\text{Base Index} - \text{Mean Index} + \text{Std Index}, 0), & \text{Base index} < \text{Mean index} \end{cases} \tag{2.1}$$

positive – количество позитивных предложений,

negative – количество негативных предложений,

total – количество предложений,

Mean index – среднее значения для пар источник сбора данных и модели, которая обрабатывала предложения.

2.2. Проектирование определения наиболее подходящей модели

Для определения наиболее подходящей модели будет использоваться набор данных, состоящий из 6,000 предложений, размеченных тремя экспертами с учетом этических практик. Однако классы предложений в этом наборе данных оказались несбалансированными, как показано на диаграмме 2.1 (с отрицательными предложениями, обозначенными «-», в большем количестве, чем положительными предложениями, обозначенными «+»). Поэтому в качестве основной метрики будет использоваться F1-мера, так как она наиболее подходит для работы с несбалансированными наборами данных. Для улучшения работы алгоритмов для определения класса каждого предложения будет использоваться наиболее часто встречающийся класс, назначенный экспертами.

Для обработки текста будут рассмотрены алгоритмы, результат работы которых будет подаваться на вход алгоритму кластеризации:

1. Мешок слов.
2. TF-IDF.
3. Word2Vec обученный на русском языке.

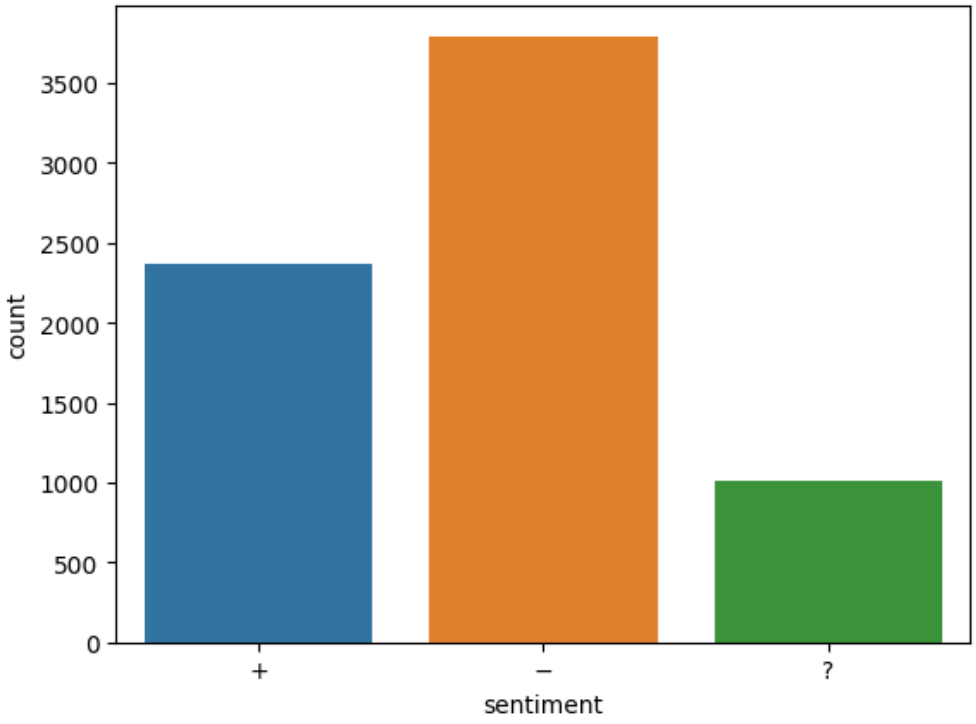


Рисунок 2.1 – Распределение классов

4. fastText обученный на русском языке [31].
5. Модификация BERT для русского языка RuBERT [32].
6. RuBERT дообученый на классификацию эмоций.
7. Дообученый RuBERT на собранных данных.

Перед подачей в модели данные отзывы будут предварительно очищены от цифр и ссылок, а также произведено их приведение к начальной форме (лемматизация). Для моделей, основанных на BERT, отзывы будут обрабатываться в двух вариантах: исходный текст и текст, подвергнутый обработкам.

Для определения наиболее подходящего алгоритма определения этичности будут перебираться все пары метода обработки текста и моделей для классификации. Для каждого алгоритма классификации будут подобраны оптимальные параметры, которые позволят сделать наилучшее предсказание на данных.

Для логистической регрессии будут подобраны следующие параметры:

1. C – параметр отвечающий за силу регуляризации алгоритма. Будет подбираться от 0.00001 до 100.
2. Penalty – параметр отвечающий за тип регуляризации l1 или l2.

3. Solver – тип алгоритма оптимизации libliniar[18] или saga[33].
4. Max_iter– количество итераций для обучения. Будет подбираться от 100 до 1000.

Для метода опорных векторов будут подобраны следующие параметры:

1. C – параметр отвечающий за силу регуляризации алгоритма. Будет подбираться от 0.00001 до 100.
2. Kernel – тип ядра для SVM (linear, poly, rbf или sigmoid).
3. Gamma – коэффициент ядра (scale или auto).
4. Degree – степень полиномиального ядра. Будет подбираться от 1 до 5

Для случайного леса будут подобраны следующие параметры:

1. N_estimators – количество итераций для обучения. Будет подбираться от 100 до 1000.
2. Max_depth – глубина дерева. Будет подбираться от 1 до 10.
3. Max_features – количество признаков для разбиения на каждом узле (корень из количества параметров или логарифм по основанию 2).
4. Criterion – критерий для разбиения узлов дерева (Джини или кросс-энтропия).

Для градиентного бустинга (scikit-learn) будут подобраны следующие параметры:

1. Learning_rate – сколько вносит каждое дерево в алгоритм. Будет подбираться от 0.00001 до 1.
2. N_estimators – количество итераций для обучения. Будет подбираться от 100 до 1000.
3. Max_depth – глубина дерева. Будет подбираться от 1 до 10.
4. Max_features – количество признаков для разбиения на каждом узле (корень из количества параметров или логарифм по основанию 2).

Для xgboost будут подобраны следующие параметры:

1. N_estimators – количество итераций для обучения. Будет подбираться от 100 до 1000.
2. Max_depth – глубина дерева. Будет подбираться от 1 до 10.

3. Learning_rate – сколько вносит каждое дерево в алгоритм. Будет подбираться от 0.00001 до 1.
4. Gamma – минимальное уменьшение функции потерь, необходимое для создания нового разбиения на узле. Будет подбираться от 0 до 20.
5. Subsample – доля обучающих примеров, используемых для обучения каждого дерева. Будет подбираться от 0 до 1.

Для catboost будут подобраны следующие параметры:

1. N_estimators – количество итераций для обучения. Будет подбираться от 100 до 1000.
2. Max_depth – глубина дерева. Будет подбираться от 1 до 10.
3. Learning_rate – сколько вносит каждое дерево в алгоритм. Будет подбираться от 0.00001 до 1.

Потом результаты работы каждого алгоритма с лучшими гиперпараметрами будут сравниваться между друг другом по F1 и так определиться лучшая модель.

2.3. Проектирование архитектуры системы

Система будет разделена на отдельные независимые компоненты (микросервисы), что позволит ей быть надежной (если в какой-то части системы будут сбои, то остальная часть системы продолжит работать) и масштабируемой (легко добавлять новые компоненты). Каждый микросервис системы будет представлять собой docker container[34], который будет управляться с помощью docker compose. Каждый сервис будет реализовывать отдельный компонент бизнес-логики и взаимодействовать с другими компонентами через REST API.

Было выделено 4 главных компонента бизнес логики:

1. Работа с базой данных – это HTTP API, который обеспечивает возможность сохранения и получения данных из базы данных. Данный компонент принимает запросы на сохранение данных, получение информации из базы данных и возвращает результаты обработки этих запросов.
2. Сбор данных – компонент, который отвечает за сбор информации с нескольких источников. Для этого используется несколько независимых сборщиков данных, которые работают с различными сайтами и другими источниками.

3. Обработка данных – данный компонент содержит несколько моделей, которые используются для анализа данных. Эти модели производят различные виды анализа, от простой фильтрации и сортировки до более сложных операций анализа и прогнозирования.
4. Агрегирование данных – этот компонент отвечает за агрегацию обработанных данных в единый индекс. Данный индекс может быть использован для удобного представления полученных результатов в виде отчетов и графиков. Данный модуль нужен для того чтобы быстро получать новые графики, так как агрегирование всех отзывов для компаний может занимать много времени.

Результат архитектуры системы на рис. 2.2.

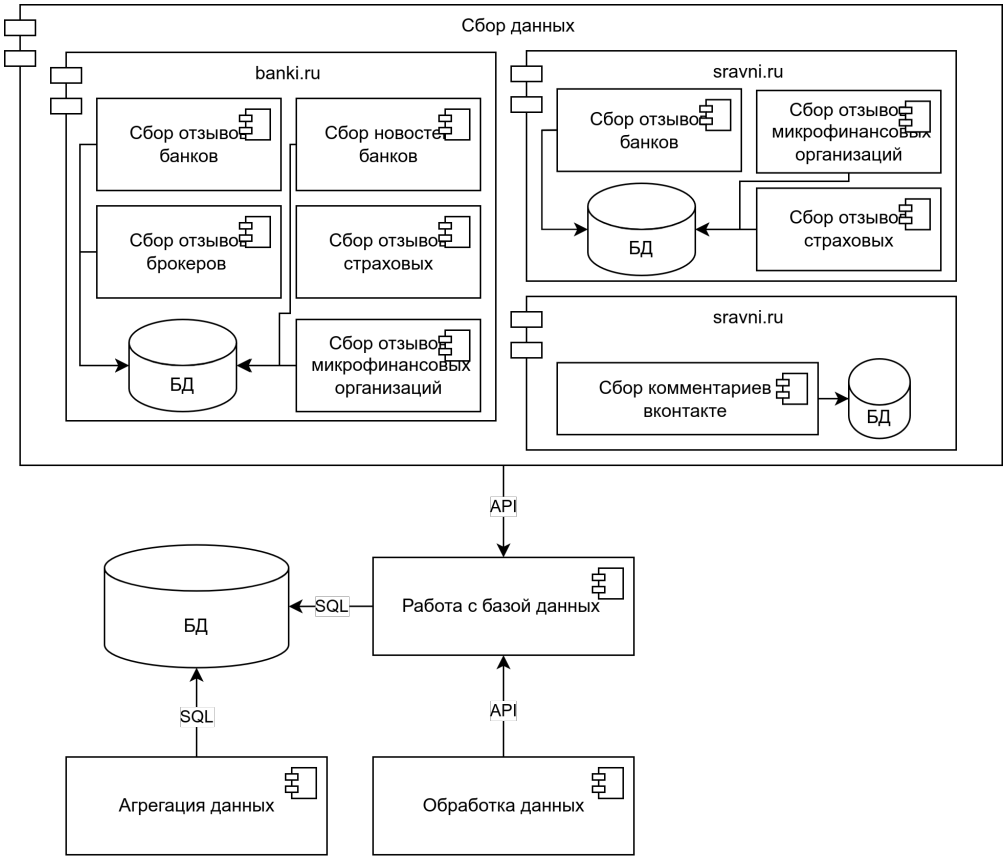


Рисунок 2.2 – Диаграмма архитектуры системы

Сервис для работы с базой данных, который будет обеспечивать сохранение и получение информации из различных сервисов сбора и обработки данных. Для этого будет предоставлен API, который будет использоваться для отправки и получения данных.

Сервисы сбора данных будут отправлять собранные тексты в формате JSON на сервис работы с базой данных с помощью HTTP запросов. Кроме того, информация, необходимая для сбора данных, будет храниться в базах данных соответствующих сервисов. У каждого сборщика данных будет своя база данных, что соответствует принципам микросервисной архитектуры.

Сервис агрегации данных будет периодически обновлять базу данных один раз в день для обеспечения актуальности данных.

Сервис сбора данных будет включать несколько моделей машинного обучения, которые будут использоваться для анализа данных, полученных из сервиса сбора данных. После обработки данных, результаты будут отправляться обратно в сервис сбора данных.

2.4. Проектирование базы данных

Исходя из поставленных требований было решено разделить базу данных на 2 подчасти:

1. Основная база данных будет хранить данные.
2. База данных для агрегации будет позволять быстро получать агрегированные данные.

2.4.1. Проектирование основной базы данных

На основании требований была разработана следующая схема базы данных:

Таблица сфер компаний позволяет в дальнейшей удобно фильтровать данные в зависимости от типа компании.

Таблица 2.1 – Таблица сфера компании

Название	**Тип**	**Описание**
Идентификатор	Целое	Уникальный идентификатор
Сфера компании	Строка	

Таблица со списком компании будет хранить основную информации о компаниях.

Таблица 2.2 – Таблица компаний

Название	**Тип**	**Описание**
Идентификатор	Целое	Уникальный идентификатор
Название компании	Строка	
Описание компании	Строка	Дополнительное поле для сохранения вспомогательной информации о компании
Лицензия компании	Строка	По лицензии компаний может будет сопоставлять компании на разных сайтах
Код сферы компании	Целое	Внешний ключ из таблицы Сфера компании

Аналогично для сфер компаний таблица для типов источников позволяет удобно работать с данными в дальнейшем.

Таблица 2.3 – Таблица тип источников

Название	**Тип**	**Описание**
Идентификатор	Целое	Уникальный идентификатор
Название типа источника	Строка	

Таблица источников будет хранить информацию об источниках и когда было последнее обновление данных для них (в полях «состояние сборщика данных» и «дата последнего сбора данных»). Поле «состояние сборщика данных» будет иметь формат json, так как для разных источников информации потребуется сохранять информацию в различном виде и сложно определить наиболее подходящий формат заранее.

Таблица 2.4 – Таблица источники

Название	**Тип**	**Описание**
Идентификатор	Целое	Уникальный идентификатор
Сайт	Строка	Сайт источника
Код типа источника	Целое	Внешний ключ из таблицы тип источника
Состояние сборщика данных	JSON	Данные о текущем состояние сборщика данных, если возникнет сбой
Дата последнего сбора	DateTime	Точка когда сбор данных закончился, для дальнейшего сбора данных

Аналогично для сфер компаний таблица для типов модели позволяет удобно работать с данными в дальнейшем.

Таблица 2.5 – Таблица тип модели

Название	**Тип**	**Описание**
Идентификатор	Целое	Уникальный идентификатор
Название модели	Строка	

Таблица модели позволяет сохранять информацию о различных моделях в дальнейшем.

Таблица 2.6 – Таблица модели

Название	**Тип**	**Описание**
Идентификатор	Целое	Уникальный идентификатор
Название модели	Строка	
Код типа модели	Целое	Внешний ключ на таблицу тип модели

Таблица текст сохраняет мета информацию о тексте отзыва.

Таблица 2.7 – Таблицы текст

Название	**Тип**	**Описание**
Идентификатор	Целое	Уникальный идентификатор
Ссылка	Строка	Ссылка на текст
Код источника	Целое	Внешний ключ из таблицы источники
Дата текста	DateTime	Время публикации текста
Заголовок	Строка	Заголовок текста
Код компании	Целое	Внешний ключ на компанию
Количество комментариев	Целое	

Так как Bert на вход принимает отдельные предложения, было решено сделать для них отдельную таблицу.

Таблица 2.8 – Таблица предложений

Название	**Тип**	**Описание**
Идентификатор	Целое	Уникальный идентификатор
Код текста	Целое	Внешний ключ из таблицы тексты
Предложение	Строка	
Номер предложения	Целое	Порядковый номер предложения в тексте

Так как результат работы модели может отличать в зависимости от ее типа, то поле «результат» будет массивом.

Таблица 2.9 – Таблица результатов анализа текстов

Название	**Тип**	**Назначение**
Идентификатор	Целое	Уникальный идентификатор
Код предложения	Целое	Внешний ключ из таблицы предложения
Код модели	Целое	Внешний ключ из таблицы модели
Результат	Вещественный массив	Результат работы модели
Обработано	Логическое	Показатель, обработано ли предложение или нет

Диаграмма полученной схемы базы данных рис. Б.1.

2.4.2. Проектирование базы данных для агрегации

При сборе функциональных требований было выявлено, что надо быстро показывать количество собранных отзывов и индекс компаний.

Обработанные данные из таблицы 2.9 агрегируются для каждого квартала и рассчитываются по формуле 2.1.

Таблица 2.10 – Таблица для расчета и показа индекса

Название	**Тип**	**Описание**
Идентификатор	Целое	Уникальный идентификатор
Год	Целое	Год за который был агрегирован индекс
Квартал	Целое	Квартал за который был агрегирован индекс
Название модели	Строка	
Сайт источника	Строка	
Тип источника	Строка	
Название компании	Строка	
Код компании	Целое	Для запросов через API
Нейтральный	Целое	Количество нейтральных предложений за период
Позитивный	Целое	Количество позитивных предложений за период
Негативный	Целое	Количество негативных предложений за период
Базовый индекс	Вещественное	Индекс для расчета итогового индекса
Средний индекс	Вещественное	Индекс для расчета итогового индекса
Std индекс	Вещественное	Индекс для расчета итогового индекса
Индекс	Вещественное	Рассчитанный индекс

Собранные отзывы из таблицы 2.7 агрегируются для каждого месяца и рассчитывается количество собранных отзывов за месяц.

Таблица 2.11 – Таблица для расчета и показа индекса

Название	**Тип**	**Описание**
Идентификатор	Целое	Уникальный идентификатор
Дата	DateTime	
Квартал	Целое	Квартал за который был агрегирован индекс
Тип источника	Строка	
Сайт	Строка	
Количество отзывов	Целое	

Диаграмма полученной схемы базы данных рис. Б.2.

2.5. Проектирование модуля работы с данными

Модуль будет представлять собой HTTP API для работой с базой данных.

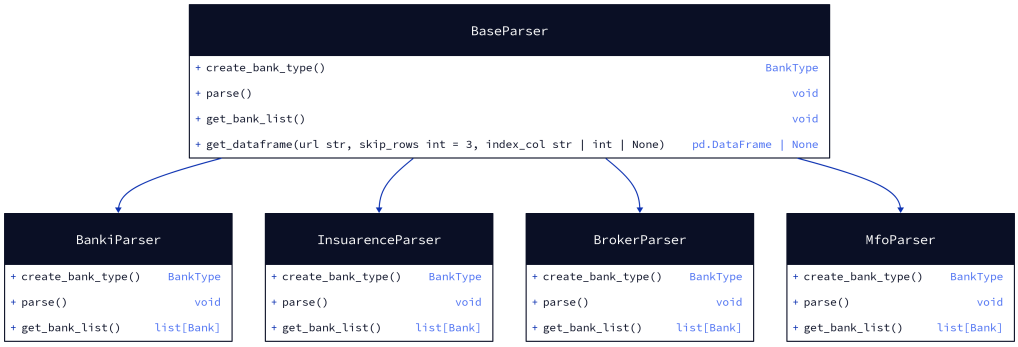
Для работы с базой данных будут созданы классы, которые представляют ORM-модель для работы с базой данных.

При первом старте приложение будет получаться список компаний (банки, брокеры, микрокредитные организации и страховые) с сайта «Центрального банка России» и помещаться в базу данных. Из этих данных будет собираться лицензия компании и название компании, для микрокредитных организаций дополнительно будет собираться основной государственный регистрационный номер (ОГРН), так как под одной лицензией может работать несколько компаний. При последующих стартах приложение будет проверяться, что в каждом списке есть компании и новые компании не будут выгружаться.

Далее создаются объекты класса Bank с использованием полученных данных и добавляются в список cbr_banks, затем он возвращается как результат работы функции.

Таким образом, принцип работы данного алгоритма заключается в извлечении необходимых данных из HTML-кода веб-страницы и преобразовании их в объекты клас-

са Bank, что позволяет автоматизировать процесс получения и анализа информации о компаниях. Диаграмма классов рис. **??**.



Для работы с источниками текстов необходимо сделать запросы для типов источников и самих источников. Также для обновления состояния сборщика данных надо сделать отдельный метод `PATCH`, который позволит обновлять время и состояние источника данных по идентификатору. Также при создании источника будет проверяться существует ли такой тип источника или нет. Если его не существует, то такой тип будет создаваться.

Сохранение текстов будет доступно по методу `POST` с передачей данных о тексте и состоянии сборщика данных. При выполнении запроса должно обновляться состояние сборщика данных, а каждый текст должен сохраняться, как набор предложений. При получении предложений должны выбираться такие предложения, которые еще не обработаны моделью.

Работа с моделями будет происходить аналогично источникам. При сохранении модели будет проверяться есть ли такой тип модели или нет. Если его нет, то он будет создан.

Также необходима возможность получения списка компаний с помощью API по различным сферам работы.

В результате проектирования должно получиться API, которое реализует запросы представленные в таблице B.

2.6. Проектирование модуля агрегации данных

Для построения индекса этичности компаний будет ежедневно агрегироваться база данных и перестраиваться индексы.

2.7. Проектирование модуля сбора данных

У всех сборщиков данных одинаковый принцип работы (рис. **??**):

1. Сборщик данных запрашивает у модуля работы с базой данных список сохраненных компаний. Модуль отвечает на запрос, отправляя список сохраненных компаний обратно.
2. Сборщик данных запрашивает у сайта для сбора данных список компаний на сайте. Сайт отправляет список компаний обратно в сборщик данных.
3. После получения списка компаний, сборщик данных сохраняет только те компании, которые уже есть в основной базе данных. Это делается для того, чтобы связать компании которые представлены на сайте и в базе данных.
4. Затем, сборщик данных начинает собирать данные для каждой компании из списка. Это может быть сделано путем отправки запросов к API сайта или сканирования страниц сайта для поиска нужных данных. Собранные данные затем сохраняются в основной базе данных. Сбор данных будет происходить до тех пор пока не соберутся все отзывы для компании, или дата отзыва дойдет до даты предыдущего сбора данных.

Для реализации сборщиков данных было решено сделать базовый класс, который представляет собой интерфейс с функцией `parse`. Из него наследуются интерфейсы для сбора данных для каждого сайта (banki.ru, sravni.ru, vk.com). Диаграмма классов рис. Г.1. От этих базовых классов для каждого сайта будут наследоваться классы, которые собирают отзывы компаний из различных сфер. Было выбрано такое решение, так как представление информации в рамках одного сайта в различных разделах может сильно различаться. Также у каждого сборщика данных будет своя база данных для сохранения информации о компаниях.

2.7.1. Проектирование сбора данных с banki.ru

Для получения данных с сайта banki.ru будут отправляться запросы на их внутренний API. Для запросов надо иметь идентификатор компании с сайта, также надо иметь идентификатор компании из модуля работы с базой данных. Исходя из требований получилась база данных 2.12. Диаграмма полученной схемы базы данных рис. Б.3.

Таблица 2.12 – Таблица для сайта banki.ru

Название	**Тип**	**Описание**
Идентификатор	Целое	Уникальный идентификатор
Идентификатор компании	Целое	Идентификатор банка в основной базе данных
Имя компании	Строка	
Код компании	Строка	Код компании для запросов по API

С этого сайта будут собираться данные о компаниях из пяти сфер:

1. **Отзывы на банки.** Список банков будет получаться из `https://www.banki.ru/widget/ajax/bank_list.json`. Затем они будут сравниваться по номеру лицензии с банками, которые есть в базе данных. Для получения отзывов о банках будут отправляться запросы на `https://www.banki.ru/services/responses/list/ajax/` и в параметры ссылки будет передаваться код банка и номер страницы с отзывами и из полученного json будут собираться данные об отзывах.
2. **Новости о банках.** В качестве списка компаний будет использоваться такой же список, как и для банков. Для получения текста новостей сначала будет собираться список новостей для компании. Для этого будут отправляться запросы на `https://www.banki.ru/banks/bank/\{bank.bank_code\}/news/` в зависимости от банка. Затем по каждой ссылке будет обрабатываться html код страницы и собираться текст новости.
3. **Отзывы на страховые компании.** Список компаний будет получаться из `https://www.banki.ru/insurance/companies/`. Затем они будут сравни-

ваться по номеру лицензии со страховыми, которые есть в базе данных. После этого будут собираться отзывы по `https://www.banki.ru/insurance/companies/`. Затем из каждой страницы компании для будет обрабатываться html код страницы и браться данные отзывов.

4. **Отзывы на брокеров.** Для получения списка компаний данные будут браться из `https://www.banki.ru/investment/brokers/list/`. Затем они будут сравниваться по номеру лицензии с брокерами, которые есть в базе данных. После этого будут собираться отзывы по `https://www.banki.ru/investment/responses/company/broker/`. Затем из каждой страницы компании для будет обрабатываться html код страницы и браться данные отзывов.
5. **Отзывы на микрокредитные организации.** Для получения списка компаний данные будут браться из `https://www.banki.ru/microloans/ajax/search`. Затем они будут сравниваться по номеру лицензии и ОГРН с компания, которые есть в базе данных. После этого будут собираться отзывы по `https://www.banki.ru/microloans/responses/ajax/responses/`. Затем из полученного json собираются отзывы о компании.

В конце сбора данных для каждого типа компаний собранные отзывы будут отправляться в модуль работы с базой данных.

2.7.2. Проектирование сбора данных с sravni.ru

Для получения данных с сайта sravni.ru будут отправляться запросы на их внутренний API. Для запросов надо иметь идентификатор компании с сайта, также надо иметь идентификатор компании из модуля работы с базой данных, также для некоторых запросов надо иметь псевдоним компании (alias). Исходя из требований получилась база данных 2.7.2. Диаграмма полученной схемы базы данных рис. Б.4.

Таблица 2.13 – Таблица для сайта sravni.ru

Название	**Тип**	**Описание**
Идентификатор	Целое	Уникальный идентификатор

(Продолжение на следующей странице)

Продолжение таблицы 2.13 – Таблица для сайта sravni.ru

Название	**Тип**	**Описание**
Идентификатор компании	Целое	Идентификатор компании в основной базе данных
Код банка в sravni.ru	Целое	
Старый код компании в sravni.ru	Целое	
Псевдоним компании	Строка	
Название компании	Строка	

Диаграмма полученной схемы базы данных рис. Б.4

С этого сайта будут собираться данные о компаниях из трех сфер:

1. **Отзывы на банки.** Список банков будет получаться из `https://www.sravni.ru/proxy-organizations/organizations` с параметром `organizationType` равным `bank`. Затем они будут сравниваться по номеру лицензии с банками, которые есть в базе данных. Для получения отзывов о банках будут отправляться запросы на `https://www.sravni.ru/bank/\{bank_info.alias\}/otzyvy/` и в параметры ссылки будет передаваться псевдоним банка и номер страницы с отзывами. И из полученного json будут собираться данные об отзывах.
2. **Отзывы на страховые компании.** Список банков будет получаться из `https://www.sravni.ru/proxy-organizations/organizations` с параметром `organizationType` равным `insuranceCompany`. Затем они будут сравниваться по номеру лицензии со страховыми, которые есть в базе данных. Для получения отзывов о банках будут отправляться запросы на `https://www.sravni.ru/strakhovaja-kompanija/\{bank_info.alias\}/otzyvy/` и в параметры ссылки будет передаваться псевдоним страховой и номер страницы с отзывами. И из полученного json будут собираться данные об отзывах.
3. **Отзывы на микрокредитные организации.** Список банков будет получаться из `https://www.sravni.ru/proxy-organizations/organizations`

с параметром `organizationType` равным `mfo`. Затем они будут сравниваться по номеру лицензии и ОГРН с компаниями, которые есть в базе данных. Для получения отзывов о банках будут отправляться запросы на `https://www.sravni.ru/zaimy/\{bank_info.alias\}/otzyvy/` и в параметры ссылки будет передаваться псевдоним банка и номер страницы с отзывами. И из полученного json будут собираться данные об отзывах.

В конце сбора данных для каждого типа компаний собранные отзывы будут отправляться в модуль работы с базой данных.

2.7.3. Проектирование сбора данных с vk.com

Для получения на сайт vk.com будут отправляться запросы на их API. Для этого предварительно будут собраны данные о всех организациях, которые у них представлены на сайте и перемещены в базу данных 2.14. Диаграмма полученной схемы базы данных рис. Б.5.

Таблица 2.14 – Таблица для сайта vk.com

Название	**Тип**	**Описание**
Идентификатор	Целое	Уникальный идентификатор
Идентификатор на vk.com	Строка	
Имя компании	Строка	
Домен компании на vk.com	Строка	

Для доступа к API будет зарегистрировано приложение для получения ключа к нему. Для каждой компании будут выгружаться посты пока дата последней выгрузки не более чем дата последнего поста для этого будет отправляться запрос на `https://api.vk.com/method/wall.get`, куда будет подставляться токен приложения и идентификатор группы. Затем для каждого поста будут выгружаться комментарии по методу `https://api.vk.com/method/wall.getComments`, а затем отправляться в модуль работы с базой данных.

2.8. Проектирование модуля обработки данных

Данный модуль будет обрабатывать полученные отзывы с помощью полученной модели. Для этого каждый день он будет запускаться, получать тексты из модуля работы с базой данных и отправлять из обратно.

2.9. Выводы по главе

В данной главе были представлены результаты проектирования системы и ее отдельных компонентов и их взаимодействие, включая базы данных и API микросервисов, согласно выявленным требованиям из первой главы. Каждый микросервис был разработан с учетом принципов микросервисной архитектуры и обеспечивает определенную функциональность, необходимую для реализации системы в целом.

Была спроектирована база данных для хранения информации об отзывах, источниках, моделях и компаниях. Базы данных были спроектированы с учетом требований к масштабируемости и производительности системы.

Также были спроектированы сервисы для работы с базой данных, ее агрегацией, сбором данных и обработки данных.

Эти результаты будут использоваться при разработке и реализации системы в следующих этапах проекта.

Глава 3 Реализация системы

В данной главе описывается реализация системы и каждого микросервиса, обучение модели.

Этап реализации можно разделить на пункты:

1. Реализация базы данных.
2. Реализации модулей для собора, работы и агрегации данных.
3. Обучение модели и реализация модуля обработки данных.
4. Развертывание системы.

3.1. Реализация определение наиболее модели

3.1.1. Обучение алгоритмов векторизации текста

Перед обучением алгоритмов надо обработать текст, чтобы повысить их эффективность. В первую очередь была осуществлена очистка от распространенных слов, также известных как стоп-слова, и были удалены знаки пунктуации.

Для лемматизации слов была применена библиотека spaCy[35], которая предоставила доступ к модели для русского языка «ru_core_news_md». Лемматизация является важным этапом, поскольку она помогает уменьшить размерность векторного пространства, используемого в методах анализа текстов, таких как мешок слов и TF-IDF.

Затем был произведен подсчет количество вхождений каждого слова 3.1. Дополнительно, из текстов были удалены слова, которые встретились менее 10 раз, так как они вносят излишнюю сложность в векторное пространство. В общей сложности такие слова составляют 85% от общего числа слов 3.2.

В результате работы алгоритма из текста «Тем самым оставив меня без средств к существованию, тем более я многодетный отец, единственный кормилец семьи!!!» получится «самым оставить средство единственный семья». Такие тексты подавались на вход мешка слов, TF-IDF и FastText.

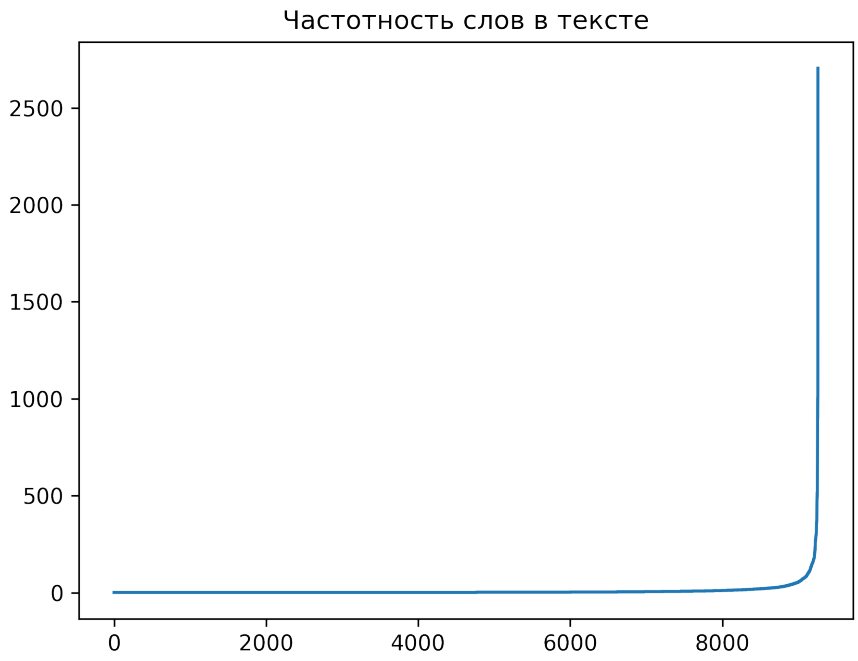


Рисунок 3.1 – График распространенности слов

Для обработки текста с использованием модели Word2Vec была выбрана модель «word2vec-ruscorpora-300» [36] из библиотеки gensim[37]. Однако, для корректной работы этой модели, необходимо добавить часть речи к каждому слову. Например, для слова «человек» необходимо добавить метку «NOUN», так как оно является существительным. Таким образом, входными данными для модели должно быть «человек$_{\text{NOUN}}$». Однако, такое преобразование текста усложняет использование данной модели, так как требуется дополнительная модель для определения частей речи слова. В связи с этим, было принято решение дополнительно воспользоватсья реализацией Word2Vec из библиотеки Navec.

Для обработки текста с помощью BERT будет использоваться 2 модели:

1. Модель от `DeepPavlov` `rubert-base-cased` [32], она обучена на русской Википедии и данных из Вконтаке
2. Модель от `blanchefort` `rubert-base-cased-sentiment-rurewiews`, за основу этой модели взят `rubert-base-cased` и был дообучен на данных отзывов, твиттера и данных из медицинский учреждений.

Рисунок 3.2 – График распространенности слов, которые встретились меньше 100 раз

В обе модели будет подаваться как исходный, необработанный текст, так и обработанный текст. Исходный текст используется, поскольку данные модели обучены на большом объеме текста и способны учитывать контекст каждого слова внутри предложения, что делает их более устойчивыми и позволяет достичь более точных результатов.

Кроме того, для оценки качества работы модели будет проведено дообучение модели RuBERT на специально собранных данных. Дообучение будет выполняться путем решения задачи классификации предложений. Для этого будет добавлен слой нейронов к предобученной модели, принимающий векторные представления исходного текста и предсказывающий вероятности принадлежности к различным классам. Этот слой называется «головой классификации» (classification head).

Модель была дообучена в течение 7 эпох на собранных данных, при этом последовательно размораживались слои модели. Для лучшего качества обучения модели использовался оптимизатор `Adam` [38]. В результате дообучения были достигнуты следующие значения метрики F1 0.75 при дообучении последнего слоя и 0.77 при дообучении

последних двух слоев. Эти результаты свидетельствуют о достигнутом качестве модели в решении задачи классификации текстов.

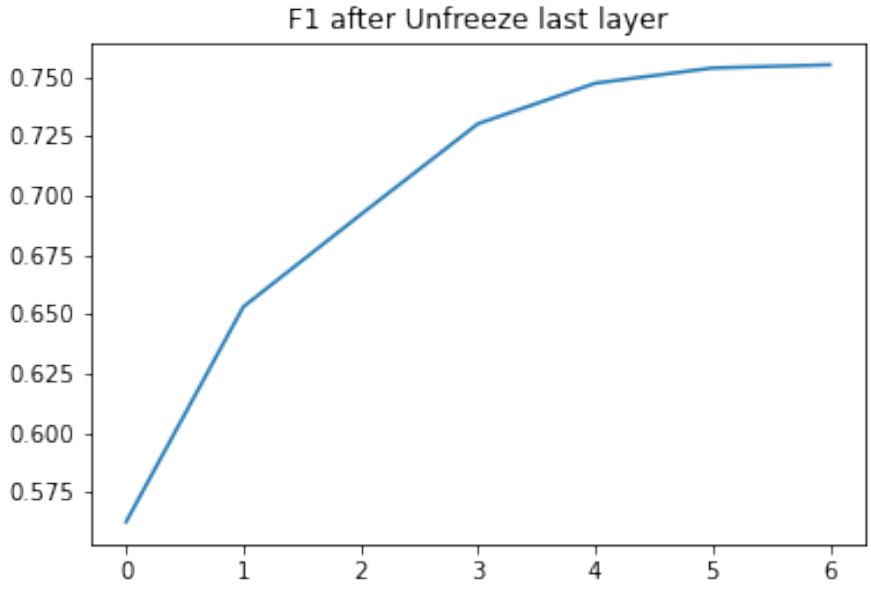


Рисунок 3.3 – График изменения метрики F1 при обучении последнего слоя

Для получения эмбеддингов из моделей BERT в цикле в модель подавались предложения и брались данные из последнего слоя и сохранялись.

3.1.2. Обучение алгоритмов классификации

Для лучшего сохранения результатов использовалась библиотека MLFlow[39]. Она поможет лучше сохранять и следить за результатами обработки. Пример интерфейса **??**. Для подбора гиперпараметров будет использоваться библиотека Optuna[40].

Для удобной интеграции с MLFlow цикл обучения каждой модели помещался в отдельную папку. Тип данных, который принимает на вход модель получается из аргументов с которым она запущена. Затем для каждой модели запускается 30 итераций для подбора гиперпараметров с помощью optuna. Для каждой итерации в MLFlow сохраняется параметры с которыми обучалась модель и результат метрики F1. После обучения всех моделей для подбора гиперпараметров выбираются наиболее оптимальные параметры и обучается итоговая модель и сохраняется в MLFlow. В итоге каждая модель запускалась с каждым типом обработки текста.

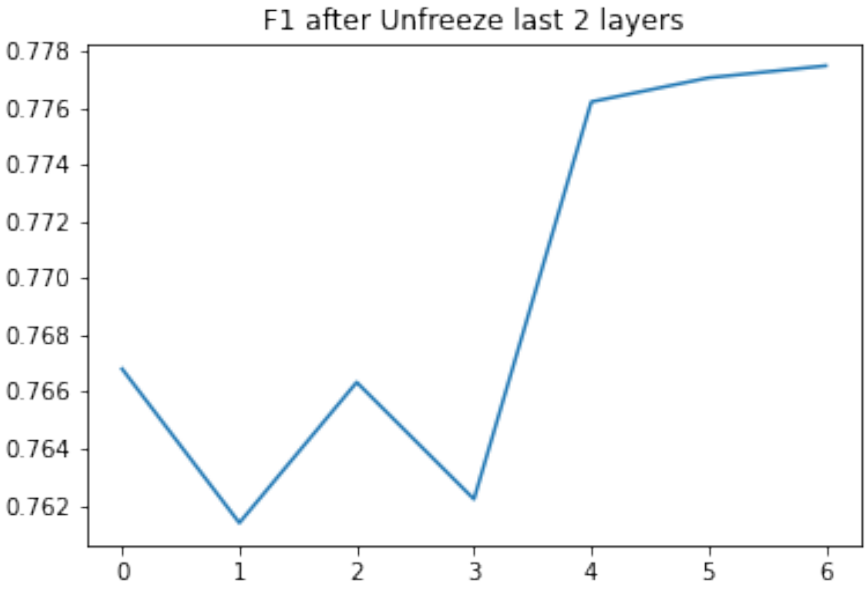


Рисунок 3.4 – График изменения метрики F1 при обучении последних 2 слоев

3.1.3. Результаты

В результате получилось, что лучший результат дало дообучение RuBERT и логистическая регрессия 3.5.

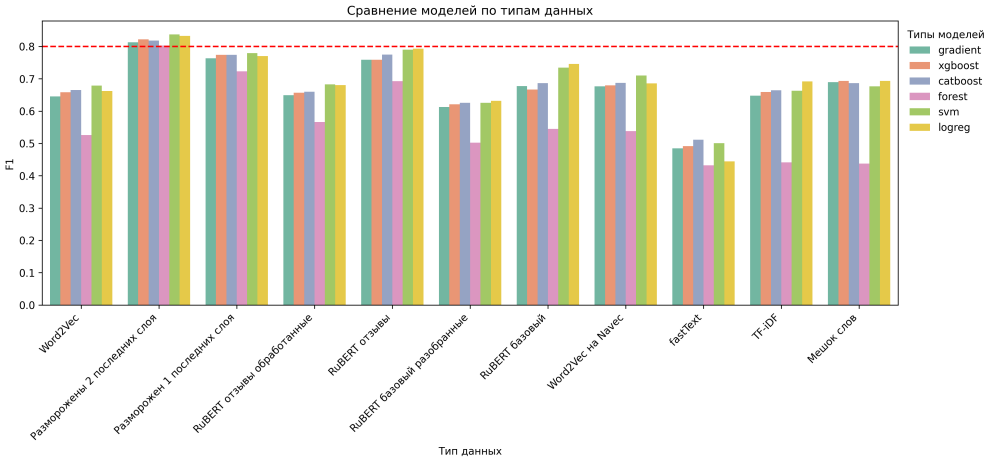


Рисунок 3.5 – Результат обучения моделей

Таблица 3.1 – Таблица результатов обучения моделей

Тип входных данных	**logreg**	**svm**	**forest**	**gradient**	**xgboost**	**catboost**
Мешок слов	0.69	0.68	0.44	0.69	0.69	0.69
TF-IDF	0.69	0.66	0.44	0.65	0.66	0.66
Word2Vec	0.66	0.68	0.53	0.65	0.66	0.67
Word2Vec Navec	0.69	0.71	0.54	0.68	0.68	0.69
fastText	0.44	0.5	0.43	0.48	0.49	0.51
RuBERT базовый	0.75	0.73	0.54	0.68	0.67	0.69
RuBERT базовый обработанные тексты	0.63	0.63	0.5	0.61	0.62	0.63
RuBERT эмоции	0.79	0.79	0.69	0.76	0.76	0.78
RuBERT эмоции обработанные тексты	0.68	0.68	0.57	0.65	0.66	0.66
RuBERT разморожен 1 последних слоя	0.77	0.78	0.72	0.76	0.77	0.77
RuBERT разморожены 2 последних слоя	0.83	0.83	0.8	0.81	0.82	0.82

3.2. Реализация базы данных

Для хранения информации в системе была выбрана СУБД PostgreSQL. Для создания базы данных был выбран подход «code first», который позволяет определить структуру базы данных в виде классов на языке Python. Для этого использовалась библиотека Sqlalchemy [41], которая обеспечивает ORM-модель для работы с базами

данных. При запуске приложения база данных будет создаваться автоматически на основе определенных классов.

Для определения структуры базы данных был создан базовый класс `DeclarativeBase`, который является родительским для всех классов, определяющих таблицы базы данных. Каждая таблица базы данных определяется в виде отдельного класса, который наследует базовый класс и содержит определения столбцов и связей между таблицами.

Для ускорения работы запросов все поля, которые являются внешними ключами были проиндексированы. Также в таблице с информацией о текстах были добавлены индексы, которые извлекают из даты год и квартал.

Для обеспечения возможности модернизации базы данных в дальнейшем была использована библиотека alembic, которая обеспечивает миграции базы данных и позволяет вносить изменения в структуру базы данных без потери данных.

3.3. Реализация модуля работы с базой данных

Для реализации API используется асинхронный фреймворк FastAPI и для взаимодействии с базой данных асинхронная библиотека asyncpg. Для валидации приходящих данных и ответов для каждого запроса была создана своя модель с помощью библиотеки Pydantic. Также с помощью Pydantic был сделан класс для получения строки подключения к базе данных из переменных окружения.

При старте приложения сначала проверятся подключение с базой данных и проверяется ее версия, если она не актуальна, то выполняются миграции для ее актуализации. Затем проверятся список компаний, если список компаний пустой, то собирается данные о банках, брокера, страховых и микрофинансовых организациях.

Информация о банках будет собираться по ссылке `https://www.cbr.ru/banking_sector/credit/FullCoList/`. Алгоритм начинается с получения объекта BeautifulSoup [42], который содержит HTML-код веб-страницы. Затем происходит итерация по всем элементам таблицы, начиная со второй строки, так как в первой находится заголовки для каждой колонки. Для каждой строки таблицы находятся все ячейки, извлекаются регистрационный номер (номер лицензии) и название банка. В списке также есть платежные небанковские кредитные организации, которые имеют

буквы на конце лицензии, например `3511-K` у «Деньги.Мэйл.Ру». Для этого такие номера будут разделяться по «-» и браться номер и преобразовываться в число. Затем собранные данные помещаются в базу данных.

Для сбора данных о брокерах будет обрабатываться excel файл, который доступен по ссылке `https://www.cbr.ru/vfs/finmarkets/files/supervision/list_brokers.xlsx`, с помощью библиотеки pandas [43]. При запуске происходит загрузка таблицы с данными о брокерах в формате Excel, после чего данные из таблицы считываются. Затем происходит итерация по строкам таблицы и для каждой строки создается экземпляр класса Bank, который содержит информацию о банке-брокере, такую как номер лицензии, наименование организации и тип банка. Для удобства хранения номера лицензии, из них удалялись все знаки «-».

Для сбора данных о страховых будет обрабатываться excel файл, который доступен по ссылке `https://www.cbr.ru/vfs/finmarkets/files/supervision/list_ssd.xlsx`. Так как в файле много строк, которые не содержат номеров или наименований банков, то они удаляются из него. Номера лицензий хранятся в формате `СИ № 3847` или `ОС № 1083 - 05` и для получения номера берется первое число которое встретилось в строке с помощью регулярного выражения. Затем полученная информация помещается в базе данных.

Для сбора данных о микрофинансовых организациях будет обрабатываться excel файл, который доступен по ссылке `https://www.cbr.ru/vfs/finmarkets/files/supervision/list_ssd.xlsx`. В этом файле номер лицензии разбит по 5 ячейкам и в части из отсутствуют числа. Поэтому отсутствующие ячейки заполняются нулями и содержание ячеек объединяется для получения результата. Потом также берется название компании и эта информация помещается в базу данных.

API было реализовано согласно требованиям описанными во второй главе.

Алгоритм получения предложений для обработки проверяет, какие из них уже были обработаны моделью, а какие - нет. Если для каждого запроса искать пересечение множества предложений, которые еще не обработаны моделью и уже обработаны, это может занять много времени. Поэтому сначала выполняется запрос (3.1), который ищет предложения, еще не обработанные моделью. Если таких нет, то в таблицу с ре-

зультатами добавляются 100 000 предложений с пустыми результатами, чтобы было проще искать предложения при дальнейших запросах. Затем с помощью запроса (3.2) из таблицы с результатами выбираются предложения, еще не обработанные моделью. Ниже приведены SQL запросы, которые генерирует ORM.

```
INSERT INTO text_result (text_sentence_id, model_id, is_processed)
SELECT text_sentence.id, :model_id, false
FROM text_sentence
JOIN text ON text_result.text_id = text.id
JOIN source ON text.source_id = source.id
LEFT JOIN (
  SELECT text_result.text_sentence_id
  FROM text_result
  WHERE text_result.model_id = :model_id
) AS subq ON text_sentence.id = subq.text_sentence_id
WHERE source.site IN (:sources) AND subq.text_sentence_id IS NULL
LIMIT 100000;
```

Листинг 3.1: SQL запрос на вставку не обработанных предложений

```
SELECT text_sentence.id, text_sentence.sentence
FROM text_sentence
JOIN (
  SELECT text_result.text_sentence_id, text_result.id
  FROM text_result
  WHERE text_result.model_id = :model_id AND text_result.is_processed
    = false
  LIMIT :limit
) AS sub
ON text_sentence.id = sub.text_sentence_id;
```

Листинг 3.2: SQL запрос на получение еще не обработанных предложений

Для разделение текста на предложения при получении текста используется библиотека `nltk` [44].

Для валидации параметров отвечающих за тип индекса этичности, список источников и период агрегации для получения агрегированных данных были сделаны `Enum`-классы. Если в запрос для получения статистики был передан параметр показы-

вающий, что надо агрегировать только по годам, то в запрос подставлялась дополнительная часть с `group by`.

Для получения данных об обработанных предложения в зависимости от типа запрашиваемого индекса в запрос подставлялся нужный тип индекса и проводилась агрегация данных аналогично запросу на получение статистики.

3.4. Реализация модуля агрегации данных

Для реализации этого модуля для взаимодействии с базой данных используется синхронная библиотека psycopg2, а в качестве ORM Sqlalchemy, для регулярного обновления данных используется библиотека schedule, которая позволяет делать регулярные операции.

При запуске модуля начинается подсчет количества собранных отзывов и расчет индекса этичности в разных потоках.

Так как в базе данных находится очень много элементов, то было решено обновлять данные напрямую из SQL. Код запроса на расчет статистки 3.3.

```
INSERT INTO text_sentence_count (count_reviews, date, quarter,
   source_type, source_site)
SELECT COUNT(text.id) AS reviews_count,
       DATE_TRUNC('month', text.date) AS month,
       EXTRACT('quarter' FROM text.date) AS quarter,
       source_type.name AS source_type,
       source.site AS source_site
FROM text
JOIN source ON text.source_id = source.id
JOIN source_type ON source.source_type_id = source_type.id
GROUP BY month, quarter, source.site, source_type.name;
```

Листинг 3.3: SQL запрос на подсчет количества предложений

Запрос для создания запроса 3.4 на расчет данных было решено использовать несколько подзапросов:

1. Сначала рассчитывается логарифм результата обработки предложений для каждой колонки. Для избежания проблем с логарифмами к каждому значению добавляется маленькое число, так как у некоторые значения могут быть

нулевыми. Этот подзапрос создан для того, чтобы ускорить выполнение, так как этот расчет можно было объединить со следующим подзапросом, но из-за этого пришлось бы пересчитывать одинаковые значения несколько раз.

2. Затем для подсчета предложений разных типов определяется их категория. Для этого используется конструкция `case when`, где значение обработанных категорий сравнивается попарно.
3. Потом к полученным данным присоединяются данные из других таблиц. Извлекается информация о квартале и дате, значения с предыдущего шага суммируются. Сам запрос объединяется для каждого квартала компаний, для каждого источника отдельно.
4. И в конце полученные данные вставляются в таблицу.
5. Затем уже на агрегированных данных рассчитываются значение индекса согласно формуле 2.1.

```
INSERT INTO aggregate_table_model_result (bank_id, bank_name, quater,
    year, model_name, source_site, source_type, positive, neutral,
   negative, total)
SELECT
    extract(year from text.date) as year,
    extract(QUARTER from text.date) as quarter,
    bank.id as "bank_id",
    model.name as "model_name",
    source.site as "source_site",
    source_type.name as "source_type_name",
    sum(positive) as "positive",
    sum(neutral) as "neutral",
    sum(negative) as "negative",
    sum(positive+neutral+negative) as total
FROM
    (SELECT
        text_sentence_id,
        model_id,
        case when (log_positive > log_neutral) and (log_positive >
   log_negative) then 1 else 0 end as "positive",
```

```
        case when (log_neutral  > log_positive) and (log_neutral >
   log_negative) then 1 else 0 end as "neutral",
        case when (log_negative > log_neutral) and (log_negative >
   log_positive) then 1 else 0 end as "negative"
    FROM (
        SELECT
            text_sentence_id,
            model_id,
            (LOG(result[1]+0.0000001)) as "log_neutral",
            (LOG(result[2]+0.0000001)) as "log_positive",
            (LOG(result[3]+0.0000001)) as "log_negative"
        FROM text_result
        WHERE model_id = 1) t) pos_neut_neg
JOIN
    text_sentence ON pos_neut_neg.text_sentence_id = text_sentence.id
JOIN
    text ON text_sentence.text_id = text.id
JOIN
    bank ON text.bank_id = bank.id
JOIN
    source ON source.id = text.source_id
JOIN
    source_type ON source.source_type_id = source_type.id
JOIN
    model ON model.id = pos_neut_neg.model_ida
GROUP BY quarter, year, source.site, source_type.name, bank.id, model
   .name
```

Листинг 3.4: SQL запрос на агрегацию обработанных предложений

3.5. Реализация модуля сбора данных

Для каждого сайта будет создана отдельная папка (модуль) со схожей структурой:

Для реализации этого модуля для взаимодействии с базой данных используется синхронная библиотека psycopg2, а в качестве ORM Sqlalchemy, для регулярного обнов-

ления данных используется библиотека schedule, которая позволяет делать регулярные операции, для обработки html страниц используется библиотека BeautifulSoup, также для обработки данных используется библиотека Pydantic.

1. В файле `database` будет лежать схема модели базы данных.
2. `schemes` pydantic модели для обработки текста.
3. `queries` запросы в базу данных.

Также для всех сборщиков данных была выделена общая часть, включающая модуль запросов, модулей объектов и настроек, а также модуль для запросов к базе данных. Модуль запросов является модификацией библиотеки requests [45] и предоставляет возможность повторного выполнения запросов в случае неудачи и обработки формата json. Модуль моделей содержит pydantic классы объектов для работы с запросами к базе данных и обработки данных. Модуль настроек представляет pydantic класс, который получает данные о подключении к базе данных, ссылке на API и токен для работы с API ВКонтаке из окружения приложения. Модуль для запросов к API предоставляет набор функций для выполнения запросов.

Для удобства развертывания было решено запускать сборщик данных в зависимости от аргумента с которым запущен код. Потом при запуске в зависимости от переданных аргументов создается база данных и запускается сборщик. Процесс сбора данных запускается ежедневно с помощью библиотеки schedule.

3.5.1. Разработка модуля сбора данных с banki.ru

Сбор данных с bani.ru осложнен тем, что компании из разных сфер имеют разное представление на сайте, поэтому для каждой сферы нужен свой подход. Также стоит отметить, что для успешной отправки запросов на сайт, требуется в заголовках запроса добавлять параметр «X-Requested-With» со значением «XMLHttpRequest».

Для сбора данных был создан базовый класс, который реализует главный цикл сбора дынных. При запуске сборщика данных проверяется загружен ли список компаний в базу данных или нет, если нет то в базу данных загружается список компаний с сайта и проверяется какие компании уже есть в основной базе данных. Затем полученные компании сохраняются в базе данных сборщика. Для этого каждый класс должен будет реализовать функцию для получения списка компаний `load_bank_list`. Затем

запускается сбор данных. Сначала получается на каком момента остановился сборщик данных в прошлый раз из модуля по работе с базой данных. Далее берется количество страниц отзывов у компании. Потом для каждой компании берутся тексты с помощью функции `get_page_bank_reviews` и сохраняются тексты, которые не были еще собраны. Затем полученные тексты в модуль работы с базой данных.

Как реализована функция `load_bank_list` для различных сфер:

1. **Банки**. Для получения этого списка компаний будет отправляться запрос по адресу `https://www.banki.ru/widget/ajax/bank_list.json` и из полученного json собираться список компаний.
2. **Страховые**. Для получения списка компаний сначала загружается html страница со списком по адресу `https://www.banki.ru/insurance/companies/`. Затем в ищется элемент div с атрибутом `data-module` равным `ui.pagination`. Из этого элемента из атрибута `data-options` получается количество компаний и страниц с ними. Потом для каждой страницы с компаниями ищутся все элементы `tr` с атрибутом `data-test` равным `list-row`. Из этого элемента получается вся информация о компании. Потом полученные компании сравниваются с теми, что сохранены в основной базе данных и сохраняются в базу сборщика данных.
3. **Брокеры**. Для получения списка компаний отправляется запрос по адресу `https://www.banki.ru/investment/brokers/list/`, но без дополнительного заголовка, так как только без него появляются лицензии компаний. Потом из этого списка собирается информация о компаниях и сохраняется в базе данных.
4. **Микрофинансовые организации**. Для получения списка компаний отправляется запрос по адресу `https://www.banki.ru/microloans/ajax/search`. Сначала из полученного json получается количество страниц с компаниями. Затем для каждой страницы отправляются новые запросы и обрабатывается информация о компаниях. Потом собранные компания сравниваются с компаниями из основной базы данных по номеру лицензии и ОГРН

с компаниями из основной базы данных и сохраняются в базе данных сборщика данных.

Реализация функции `get_page_bank_reviews` для различных сфер:

1. **Банки**. Для получения отзывов будет делаться запрос по адресу `https://www.banki.ru/services/responses/list/ajax/` с параметрами для определения компании и номера страницы. Из полученного json соберутся отзывы и отправятся в основную базу данных.
2. **Новости**. Для получения текстов новостей сначала будут собираться адреса новостей, а затем уже сами тексты новостей. Для сбора адресов будет отправляться запрос на `https://www.banki.ru/banks/bank/\{bank_code\}/news/`, где «`bank_code`» код банка, также в качестве параметра запроса будет отправляться номер страницы. Для получения адресов будут браться элементы «a» с классом «text-list-link», также для отбора новых новостей будут обрабатываться даты. Для этого будут браться элементы «span» с классом «text-list-date». Потом по полученным ссылкам будет браться html код страниц и браться текст новости из элементов «p».
3. **Страховые и Брокеры**. Для этих сфер тексты отзывов получаются путем обработки html страниц. Они получаются из запросов на адреса `https://www.banki.ru/investment/responses/company/broker/` для брокеров и `https://www.banki.ru/insurance/responses/company/` для страховых, к этим ссылкам добавляется код компании и номер страницы для получения отзывов. Потом для получения текста отзывов ищутся элементы «div» с классом «`responses__item__message`» и из него берется текст. Затем собранные отзывы отправляются в модуль работой с базой данных.
4. **Микрофинансовые организации**. Для получения отзывов отправляются запросы на `https://www.banki.ru/microloans/responses/ajax/responses`, где в параметры передаются код компании и номер страницы. Затем из полученного json собираются отзывы и отправляются в основную базу данных.

3.5.2. Разработка модуля сбора данных с sravni.ru

При сборе данных со sravni.ru будут отправляться запросы на их внутреннее API, которое имеет схожую структуру для всех сфер компаний. При запуске сборщика данных проверяется загружен ли список компаний в базу данных или нет, если нет то в базу данных загружается список компаний. Он будет получать путем отправки запроса на `https://www.sravni.ru/proxy-organizations/organizations` с различным значением параметра `organizationType` («bank» для банков, «insuranceCompany» для страховых компаний и «microcredits» для микрофинансовых организаций). Потом полученный список компаний проверяется со списком, который сохранен в основной базе данных. Затем полученные компании сохраняются в базе данных сборщика.

Затем запускается процесс сбора данных. Сначала получается на каком момента остановился сборщик данных в прошлый раз из модуля по работе с базой данных. Потом для каждой компании получается список отзывов. Он получается путем отправки запроса по адресу `https://www.sravni.ru/proxy-reviews/reviews` с параметром «reviewObjectType» с такими же значениями, как для получения списка компаний, и идентификатором компании на сайте sravni.ru. В результате запроса получается json, в котором находится 1000 отзывов на компанию. Из этих отзывов выбираются новые отзывы с момента предыдущего сбора данных. Потом собранные данные отправляются в основную базу данных.

3.5.3. Разработка модуля сбора данных с vk.com

Для взаимодействия с API ВКонтакте был реализован класс, который делает запросы к API и подставляет обязательные параметры, такие как токен и версия API, так и параметры которые нужны для различных методов. Также этот класс регулирует количество запросов к API, так как разрешено делать не более трех запросов в секунду.

При запуске сборщика данных проверяется загружен ли список компаний в базу данных или нет, если нет то в базу данных загружается список отобранных заранее компаний. Затем запускается процесс сбора данных. Сначала получается на каком момента остановился сборщик данных в прошлый раз из модуля по работе с базой данных. Затем для каждой компании берет публикации в группе. Для публикаций у которых разница во времени с момента предыдущего сбора данных не более недели собираются новые

комментарии. Из собранных комментариев удаляются эмоджи и идентификаторы пользователей из ссылок на профиля ВКонтакте, которые имеют вид `(ID пользователя|Имя пользователя)`. Потом собранные комментарии отправляются в модуль работы с базой данных.

3.6. Реализация модуля обработки текста

При запуске данного модуля происходит загрузка модели, которая будет использоваться для обработки текстов. После загрузки модель готова принимать новые тексты из модуля работы с базой данных. Эти тексты могут быть получены, например, из внешних источников или из предыдущих этапов обработки данных.

После получения текстов модель выполняет их обработку, применяя соответствующие алгоритмы и методы анализа. Обработанные данные отправляются обратно в модуль работы с базой данных. Таким образом, информация, полученная в результате обработки текстов моделью, становится доступной для дальнейшего использования или анализа в рамках системы.

Этот процесс запускается ежедневно, что позволяет обрабатывать новые тексты, поступающие в систему с течением времени. Регулярное выполнение данного процесса обеспечивает актуальность и своевременность обработки текстов, а также поддерживает работоспособность системы в целом.

3.7. Развертывание системы

Для развертывания системы каждый компонент был выделен в отдельный контейнер docker [34], а для оркестрации приложений существует инструмент Docker Compose, который позволяет запускать многоконтейнерные приложения с помощью YAML-файлов конфигурации. Для установки переменных окружения в контейнеры используется файл «.env», в котором содержались переменные окружения для всех приложений и путь к этому файлу прописывался в параметрах `env_file`. Для установки в названия базы данных дополнительно в конфигурации контейнеров указывалась переменная окружения `POSTGRES_DB`.

Для реализации модуля взаимодействия с базой данной в качестве базового образа использовался `python:3.10`. Потом устанавливалась библиотека nltk и данные для

работы этой библиотеки с русским языком. Затем устанавливались зависимости приложения и оно запускалось с помощью библиотеки uvicorn. Для модулей агрегации базы и сбора данных в качестве базового образа использовался `python:3.10-slim`, так как он использует меньше памяти, чем обычный.

В данной работе определен сервис «database», который запускает контейнер с PostgreSQL версии 14.4. Контейнеру также присваиваются тома для хранения данных, которые будут использоваться внутри контейнера. Для обеспечения доступности сервиса в контейнере определены порты, через которые можно подключаться к базе данных. Кроме того, в конфигурации определен healthcheck, который проверяет работоспособность сервиса, выполняя команду `pg_isready` с заданными параметрами. Этот healthcheck запускается каждые 10 секунд и проходит 5 попыток, если проверка не прошла в заданный таймаут в 5 секунд. Такой подход обеспечивает более стабильную работу контейнера и позволяет оперативно реагировать на возможные проблемы.

Для модуля работы с базой данных определен сервис «api», который зависит от сервиса базы данных и начнет работу только после того, как у базы данных пройдет healthcheck.

Для модуля сбора для каждого сайта создавался новый контейнер и они запускались с помощью команды `python main.py --site name`, где вместо name было название сайта и из какой сферы собирались тексты. Все сборщики данных были выделены в отдельный профиль для удобства запуска.

3.8. Выводы по главе

В данной главе представлена реализация системы и ее отдельных компонентов, включая базы данных и API микросервисов, согласно выявленным требованиям из первой главы. Каждый микросервис был разработан с учетом принципов микросервисной архитектуры и обеспечивает определенную функциональность, необходимую для реализации системы в целом, согласно проектированию описанном в предыдущей главе.

В результате было собрано (рис. 3.6) 10 миллионов предложений для разных компаний и они были обработаны с помощью разработанной модели (рис. 3.7).

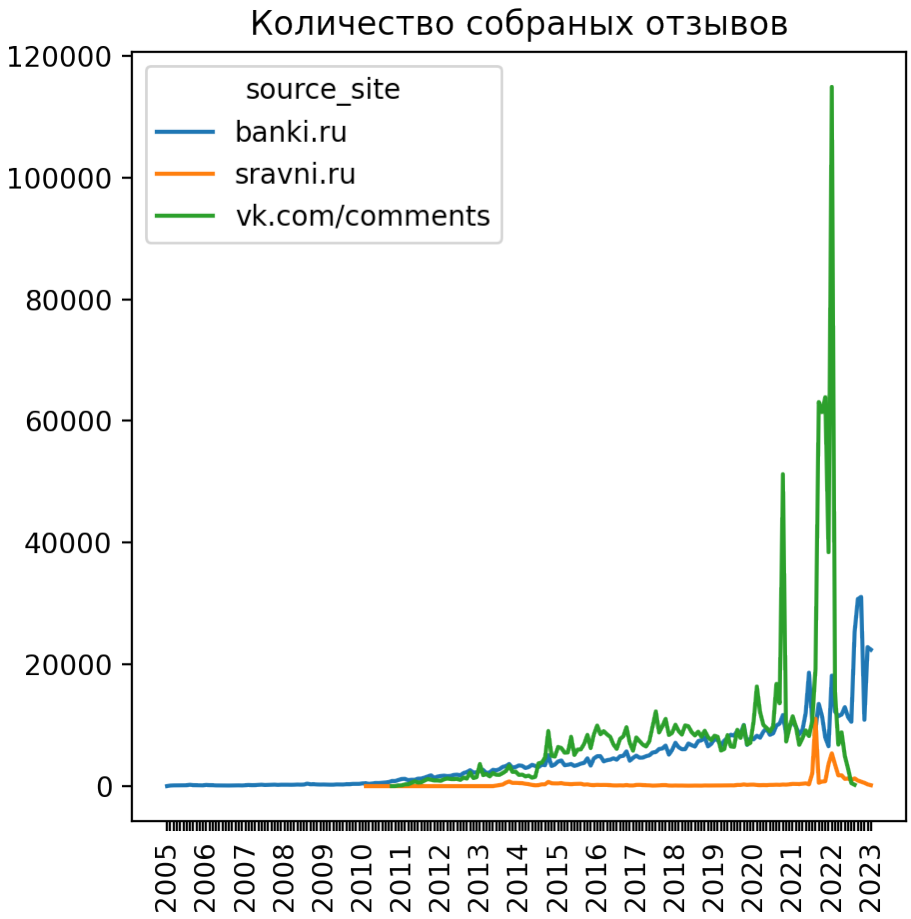


Рисунок 3.6 – График собранных предложений

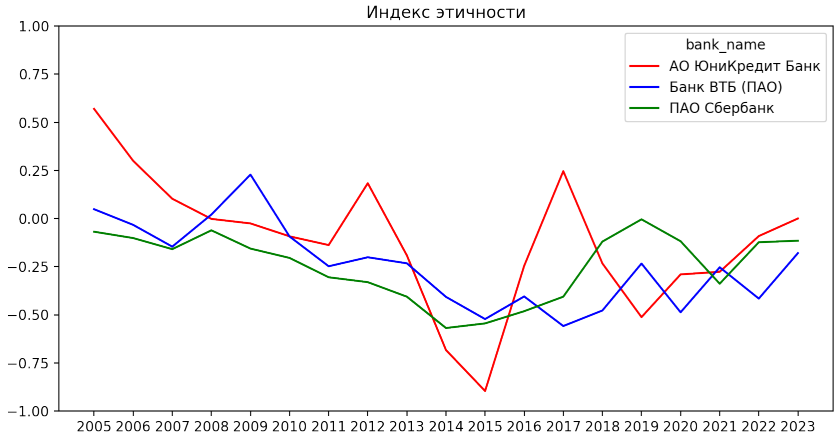


Рисунок 3.7 – График оценки этичности компаний

Глава 4 Тестирование системы

В данной главе описывается разные методы тестирования, которые были использованы при реализации системы и форматтеры кода и инструменты статического анализа также является важным, поскольку это позволяет снизить количество ошибок и повысить качество кода.

4.1. Форматирование кода

Во время реализации системы использовались различные инструменты для форматирования кода и обеспечения его качества, такие как black, isort, pyupgrade, flake8 и mypy.

Black – это инструмент для автоматического форматирования кода на Python. Он помогает унифицировать стиль кодирования и повысить читаемость кода. Он применяет определенный набор правил форматирования и применяет их к коду.

Isort – это инструмент для автоматической сортировки импортов в Python-коде. Он обеспечивает единообразие сортировки импортов, что может повысить читаемость кода и уменьшить количество ошибок.

Pyupgrade – это инструмент, который автоматически обновляет код Python до более новых версий языка, что позволяет использовать новые возможности языка и уменьшить количество устаревших функций и библиотек.

Flake8 – это инструмент для обнаружения синтаксических и стилистических ошибок в коде Python. Он проверяет код на соответствие стандартам кодирования и выдает предупреждения, если обнаруживает ошибки.

Mypy – это инструмент для статического анализа кода Python, который позволяет обнаруживать ошибки на этапе написания кода. Он проверяет типы переменных и аргументов функций и выдает ошибки, если они не соответствуют ожидаемым типам.

Также для автоматического форматирования кода использовался инструмент pre-commit, который позволяет форматировать кодовую базу перед отправкой в репозиторий с кодом.

4.2. Тестирование системы

Для тестирования системы были написаны скрипты Github actions, которые запускали тестирование системы при отправки кода в репозиторий. Для тестирования использовалась библиотека pytest [46], также для анализа покрытия кода использовалась библиотека `pytest-cov`.

4.2.1. Тестирование модуля работы с базой данных

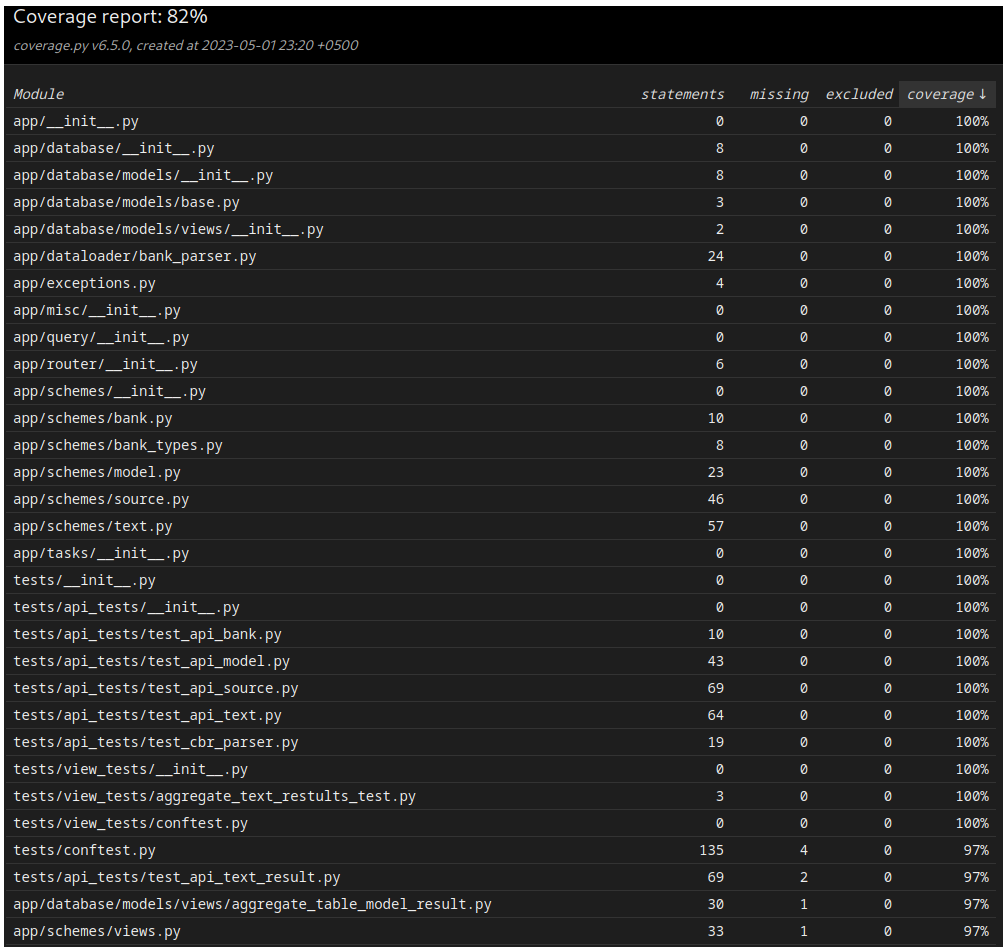
При запуске каждого теста создавалась новая база данных и к ней применялись миграции. В зависимости от нужного теста в базу данных помещались нужные данные. Каждый метод API тестировался на верные данные, не верные данные и пустые данные, затем проверялось, что записалось в базу данных. В зависимости от требований каждого теста в базу данных помещались соответствующие данные. Затем создавался клиент для тестирования API с помощью библиотеки httpx.

Для каждого метода API тестировалось его поведение на правильные, неправильные и пустые данные. Затем проверялось, что записалось в базу данных и соответствует ожиданиям.

4.2.2. Тестирование модуля сбора данных

Тестирование сборщиков данных заключается в проверке корректности извлечения и обработки данных, получаемых из HTML-страниц и json. Также важно проверять сбор данных на актуальных данных сайтов. Для имитации запросов к API и сайтам использовалась библиотека requests-mock, для сохранения данных страниц использовалась библиотека `vcrpy`. Данная библиотека при первом запросе сайта сохраняет данные запроса (заголовки, параметры и так далее), а тело запроса кодирует в байты, что позволяет сократить объем. При последующих запросов вместо реального ответа будут подставляться данные предыдущего ответа. Такой способ работы с запросами позволяет ускорить процесс тестирования. Также при запуске каждого теста создавалась новая база данных.

При написании тестирования были реализованы mock-данные для API. Для подмены запросов на тестовые сначала делались запросы (или брались данные предыдущих



Coverage report: 82%

coverage.py v6.5.0, created at 2023-05-01 23:20 +0500

Module	statements	missing	excluded	coverage ↓
app/__init__.py	0	0	0	100%
app/database/__init__.py	8	0	0	100%
app/database/models/__init__.py	8	0	0	100%
app/database/models/base.py	3	0	0	100%
app/database/models/views/__init__.py	2	0	0	100%
app/dataloader/bank_parser.py	24	0	0	100%
app/exceptions.py	4	0	0	100%
app/misc/__init__.py	0	0	0	100%
app/query/__init__.py	0	0	0	100%
app/router/__init__.py	6	0	0	100%
app/schemes/__init__.py	0	0	0	100%
app/schemes/bank.py	10	0	0	100%
app/schemes/bank_types.py	8	0	0	100%
app/schemes/model.py	23	0	0	100%
app/schemes/source.py	46	0	0	100%
app/schemes/text.py	57	0	0	100%
app/tasks/__init__.py	0	0	0	100%
tests/__init__.py	0	0	0	100%
tests/api_tests/__init__.py	0	0	0	100%
tests/api_tests/test_api_bank.py	10	0	0	100%
tests/api_tests/test_api_model.py	43	0	0	100%
tests/api_tests/test_api_source.py	69	0	0	100%
tests/api_tests/test_api_text.py	64	0	0	100%
tests/api_tests/test_cbr_parser.py	19	0	0	100%
tests/view_tests/__init__.py	0	0	0	100%
tests/view_tests/aggregate_text_restults_test.py	3	0	0	100%
tests/view_tests/conftest.py	0	0	0	100%
tests/conftest.py	135	4	0	97%
tests/api_tests/test_api_text_result.py	69	2	0	97%
app/database/models/views/aggregate_table_model_result.py	30	1	0	97%
app/schemes/views.py	33	1	0	97%

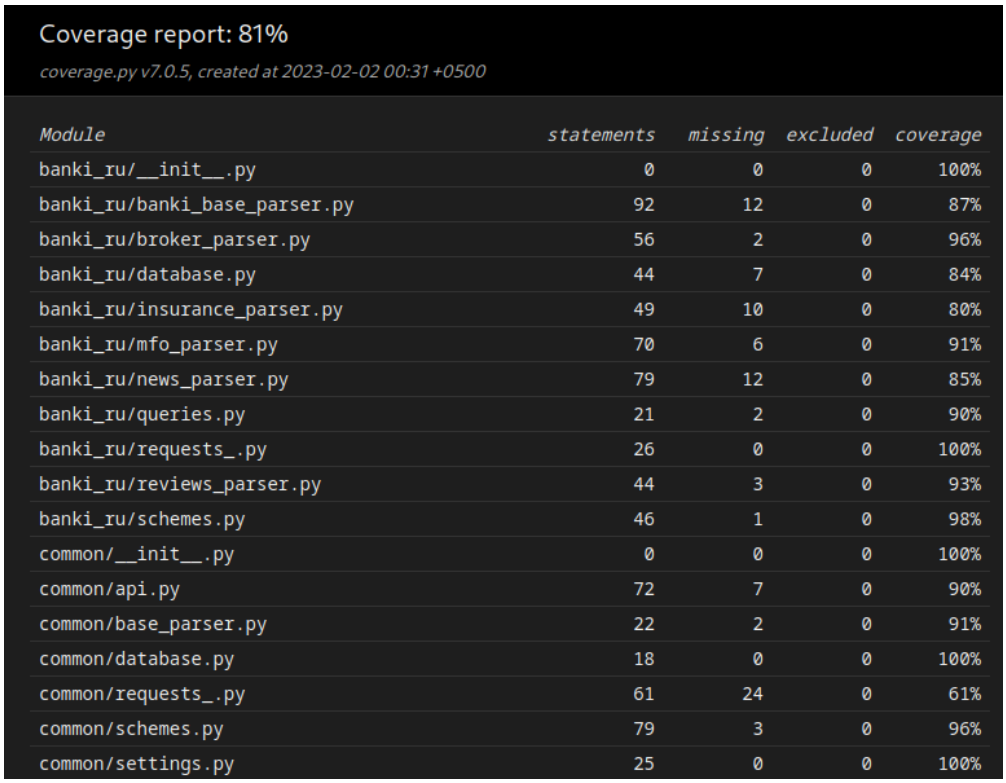
Рисунок 4.1 – Покрытие кода модуля работы с базой данных тестами

запросов), затем с помощью библиотеки `requests-mock` данные запроса подменялись на тестовые. Потом запускались тесты, которые проверяли корректности работы функций.

4.3. Выводы по главе

В данной главе были описаны различные виды тестирования, которые использовались при тестировании модулей для работы базой данных и сбора данных.

Были описаны тесты для сборщиков данных, которые позволяют убедиться в корректности работы при различных сценариях. Также были рассмотрены тесты для HTTP API, которые проверяют работу API на правильную обработку запросов и на корректное взаимодействие с базой данных.



Coverage report: 81%

coverage.py v7.0.5, created at 2023-02-02 00:31 +0500

Module	statements	missing	excluded	coverage
banki_ru/__init__.py	0	0	0	100%
banki_ru/banki_base_parser.py	92	12	0	87%
banki_ru/broker_parser.py	56	2	0	96%
banki_ru/database.py	44	7	0	84%
banki_ru/insurance_parser.py	49	10	0	80%
banki_ru/mfo_parser.py	70	6	0	91%
banki_ru/news_parser.py	79	12	0	85%
banki_ru/queries.py	21	2	0	90%
banki_ru/requests_.py	26	0	0	100%
banki_ru/reviews_parser.py	44	3	0	93%
banki_ru/schemes.py	46	1	0	98%
common/__init__.py	0	0	0	100%
common/api.py	72	7	0	90%
common/base_parser.py	22	2	0	91%
common/database.py	18	0	0	100%
common/requests_.py	61	24	0	61%
common/schemes.py	79	3	0	96%
common/settings.py	25	0	0	100%

Рисунок 4.2 – Покрытие кода модуля сборщиков данных тестами

Заключение

В ходе анализа предметной области были рассмотрены особенности алгоритмов генерации игровых уровней, включая алгоритмы машинного обучения. Были определены критерии для сравнения генерируемых уровней, и результаты показали, что уровни, созданные с использованием генеративных сетей, являются наилучшими. Среди генеративных нейронных сетей была произведена оценка трех архитектур, показывающих наибольший потенциал при генерации уровней: GAN, VAE и RNN. Среди них GAN была выбрана как наиболее подходящая для данной задачи.

Дальнейший анализ выявил ограничения каждого из этих алгоритмов. Было установлено, что текущие нейронные сети не способны генерировать уровни, удовлетворяющие заданным пользователем ограничениям, а также не могут использовать объекты из одной игры при генерации уровней для другой игры. С учетом выбора GAN для дальнейшего использования были выявлены ограничения и проблемы, с которыми сталкивается эта архитектура, включая коллапс режима, исчезающий градиент и нестабильность процедуры обучения. Анализ метрик оценки моделей и уровней привел к выделению базовых метрик, таких как качество генерации, разнообразие уровней и вычислительная эффективность.

При определении функциональных требований было решено модифицировать GAN с помощью RL-агента для улучшения проходимости сгенерированных уровней. Все требования, выявленные в ходе анализа, были формализованы в техническом задании на разрабатываемую систему в соответствии с ГОСТ 34.602-2020.

В качестве основы для GAN была выбрана модель WGAN, которая обладает большей стабильностью при обучении по методике PPO. На основании технического задания и спецификациях прецедентов было выполнено проектирование архитектуры приложения. Были определены компоненты системы, такие как модуль машинного обучения, преобразователь уровней, модуль дообучения, модуль оценки проходимости уровней и редактор уровней, а также спроектированы их взаимосвязи.

В качестве источника данных для обучения модели были выбраны пользовательские уровни из игры Super Mario Maker 2, которые в основном представляют собой

уровни «под открытым небом». Однако из-за большого объема данных и необходимости выбора подходящих уровней для обучения модели, был разработан модуль оценки проходимости уровней.

Также была построена схема базы данных для хранения сопутствующей информации, связанной с файлами уровней. Кроме того, были составлены прототипы окон пользовательского интерфейса, установлена их иерархия, построены диаграммы активности.

Для реализации поставленной задачи был осуществлен сбор наиболее полного набора данных с использованием специализированного алгоритма сбора, использующего Nintendo Clients. Был разработан модуль для преобразования игровых уровней с учетом текущих ограничений работы с форматами игровых уровней Super Mario Maker 2 и используемой кодировки игровых объектов на изображении.

Также был реализован модуль редактирования уровней с обширными функциональными возможностями, включая эмулятор. Разработан и применен модуль RL GAN, а также были устранены ошибки хаотичности начального блока, определены оптимальные значения гиперпараметров. С учетом проведенного проектирования и требований к удобству использования, был реализован пользовательский интерфейс.

В процессе работы программы была обнаружена предвзятость в генерации уровней “под открытым небом” и структур, характерных для пользовательских уровней, но не типичных для классических уровней. Были выявлены преимущества по сравнению с другими решениями, такие как контроль над генерацией, отсутствие тупиков и непроходимых участков, а также правильное применение объектов. Была проведена оценка работы моделей на 1000 сгенерированных уровнях, которая продемонстрировала показатели проходимости, разнообразия и комплексности уровней выше 90\

Дальнейшие исследования могут быть связаны с интеграцией в модуль машинного обучения LSTM или VAE, улучшением среды в которой работает RL-агент, а также использование дополнительной информации, предоставляемой с файлом уровня.

В заключении можно отметить, что проект является успешным и достигнуты поставленные цели и задачи. Результаты работы могут быть использованы в различных

областях, в том числе для исследовательской работы и применения в игровой индустрии.

Библиографический список

1. *Murè, P.* ESG and Reputation: The Case of Sanctioned Italian Banks / P. Murè, M. Spallone, F. Mango, S. Marzioni, L. Bittucci // Corporate Social Responsibility and Environmental Management. — 2021. — Vol. 28, no. 1. — P. 265–277.
2. *Семенко, И. Е.* Корпоративная Социальная Ответственность И Бизнес-Этика Компании / И. Е. Семенко // Экономические науки: актуальные вопросы теории и практики. — Наука и Просвещение, 2022. — С. 43—45.
3. *Кудрявцева, Ю. А.* Корпоративно-Социальная Ответственность В Контексте Этики Банковского Дела / Ю. А. Кудрявцева, Г. Г. Чахкиев. — 2016.
4. *Climent, F.* Ethical Versus Conventional Banking: A Case Study / F. Climent // Sustainability. — 2018. — July. — Vol. 10, issue 7, no. 7. — P. 2152.
5. *Harvey, B.* Ethical Banking: The Case of the Co-operative Bank / B. Harvey // Journal of Business Ethics. — 1995. — Dec. 1. — Vol. 14, no. 12. — P. 1005–1013.
6. *Brunk, K. H.* Exploring Origins of Ethical Company/Brand Perceptions — A Consumer Perspective of Corporate Ethics / K. H. Brunk // Journal of Business Research. — 2010. — Mar. 1. — Vol. 63, no. 3. — P. 255–262.
7. *Mitchell, W. J.* Bank Ethics: An Exploratory Study of Ethical Behaviors and Perceptions in Small, Local Banks / W. J. Mitchell, P. V. Lewis, N. L. Reinsch // Journal of Business Ethics. — 1992. — Mar. 1. — Vol. 11, no. 3. — P. 197–205.
8. *López, M. V.* Sustainable Development and Corporate Performance: A Study Based on the Dow Jones Sustainability Index / M. V. López, A. Garcia, L. Rodriguez // Journal of Business Ethics. — 2007. — Oct. 1. — Vol. 75, no. 3. — P. 285–300.
9. *Collison, D. J.* The Financial Performance of the FTSE4Good Indices / D. J. Collison, G. Cobb, D. M. Power, L. A. Stevenson // Corporate Social Responsibility and Environmental Management. — 2008. — Vol. 15, no. 1. — P. 14–28.
10. *Harris, Z. S.* Distributional Structure / Z. S. Harris // WORD. — 1954. — Aug. 1. — Vol. 10, no. 2/3. — P. 146–162.

11. *Jones, Karen Sparck.* A Statistical Interpretation of Term Specificity and Its Application in Retrieval / Jones, Karen Sparck // Journal of Documentation. — 1972. — Jan. 1. — Vol. 28, no. 1. — P. 11–21.

12. *Mikolov, T.* Distributed Representations of Words and Phrases and Their Compositionality / T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean // Advances in Neural Information Processing Systems. Vol. 26. — Curran Associates, Inc., 2013.

13. *Joulin, A.* Bag of Tricks for Efficient Text Classification / A. Joulin, E. Grave, P. Bojanowski, T. Mikolov. — 08/09/2016. — URL: `http://arxiv.org/abs/1607.01759` (visited on 05/24/2023).

14. *Devlin, J.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin, M.-W. Chang, K. Lee, K. Toutanova. — 05/24/2019.

15. *Vaswani, A.* Attention Is All You Need / A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin // Advances in Neural Information Processing Systems. Vol. 30. — 2017.

16. *Peters, M. E.* Deep Contextualized Word Representations / M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer. — 03/22/2018.

17. *Radford, A.* Language Models Are Unsupervised Multitask Learners / A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever. — 2019.

18. *Fan, R.-E.* LIBLINEAR: A Library for Large Linear Classification / R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin // The Journal of Machine Learning Research. — 2008. — June 1. — Vol. 9. — P. 1871–1874.

19. *Platt, J.* Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods / J. Platt // Adv. Large Margin Classif. — 2000. — June 23. — Vol. 10.

20. *Joachims, T.* Text Categorization with Support Vector Machines: Learning with Many Relevant Features / T. Joachims // Machine Learning: ECML-98. — Berlin, Heidelberg : Springer, 1998. — P. 137–142.

21. *Breiman, L.* Random Forests / L. Breiman // Machine Learning. — 2001. — Oct. 1. — Vol. 45, no. 1. — P. 5–32.

22. *Friedman, J. H.* Greedy Function Approximation: A Gradient Boosting Machine / J. H. Friedman // The Annals of Statistics. — 2001. — Vol. 29, no. 5. — P. 1189–1232.

23. *Chen, T.* XGBoost: A Scalable Tree Boosting System / T. Chen, C. Guestrin // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. — New York, NY, USA : Association for Computing Machinery, 08/13/2016. — P. 785–794.

24. *Mitchell, R.* Accelerating the XGBoost Algorithm Using GPU Computing / R. Mitchell, E. Frank // PeerJ Computer Science. — 2017. — July 24. — Vol. 3. — e127.

25. *Prokhorenkova, L.* CatBoost: Unbiased Boosting with Categorical Features / L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, A. Gulin // Advances in Neural Information Processing Systems. Vol. 31. — Curran Associates, Inc., 2018.

26. *Dorogush, A. V.* CatBoost: Gradient Boosting with Categorical Features Support / A. V. Dorogush, V. Ershov, A. Gulin. — 10/24/2018.

27. *Van Rossum, G.* Python 3 Reference Manual / G. Van Rossum, F. L. Drake. — Scotts Valley, CA : CreateSpace, 2009.

28. *Paszke, A.* PyTorch: An Imperative Style, High-Performance Deep Learning Library / A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala // Advances in Neural Information Processing Systems. Vol. 32. — Curran Associates, Inc., 2019.

29. *Wolf, T.* Transformers: State-of-the-Art Natural Language Processing / T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, A. Rush // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. — Online : Association for Computational Linguistics, 10/2020. — P. 38–45.

30. Encyclopaedia of Mathematics (Set).

31. *Korogodina, O.* Evaluation of Vector Transformations for Russian Word2Vec and FastText Embeddings / O. Korogodina, O. Karpik, E. Klyshinsky // Proceedings of the 30th International Conference on Computer Graphics and Machine Vision (GraphiCon 2020). Part 2. — 2020. — Dec. 17. — paper18-1-paper18–12.

32. *Kuratov, Y.* Adaptation of Deep Bidirectional Multilingual Transformers for Russian Language / Y. Kuratov, M. Arkhipov. — 05/17/2019.

33. *Defazio, A.* SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives / A. Defazio, F. Bach, S. Lacoste-Julien. — 12/16/2014. — URL: `http://arxiv.org/abs/1407.0202` (visited on 05/29/2023).

34. *Merkel, D.* Docker: Lightweight Linux Containers for Consistent Development and Deployment / D. Merkel // Linux journal. — 2014. — Vol. 2014, no. 239. — P. 2.

35. *Boyd, A.* Explosion/spaCy: V3.2.6: Bug Fixes for Pydantic and Pip / A. Boyd. — Zenodo, 05/25/2023.

36. *Kutuzov, A.* WebVectors: A Toolkit for Building Web Interfaces for Vector Semantic Models / A. Kutuzov, E. Kuzmenko // Analysis of Images, Social Networks and Texts. — Cham : Springer International Publishing, 2017. — P. 155–161.

37. *Řehůřek, R.* Software Framework for Topic Modelling with Large Corpora / R. Řehůřek, P. Sojka //. — 05/17/2010. — P. 45–50.

38. *Kingma, D. P.* Adam: A Method for Stochastic Optimization / D. P. Kingma, J. Ba. — 01/29/2017. — URL: `http://arxiv.org/abs/1412.6980` (visited on 05/10/2023).

39. *Chen, A.* Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle / A. Chen, A. Chow, A. Davidson, A. DCunha, A. Ghodsi, S. A. Hong, A. Konwinski, C. Mewald, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, A. Singh, F. Xie, M. Zaharia, R. Zang, J. Zheng, C. Zumar // Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning. — New York, NY, USA : Association for Computing Machinery, 06/17/2020. — P. 1–4.

40. *Akiba, T.* Optuna: A Next-generation Hyperparameter Optimization Framework / T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama // Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. — New York, NY, USA : Association for Computing Machinery, 07/25/2019. — P. 2623–2631.

41. *Bayer, Michael.* The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks / Bayer, Michael // The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks. — Mountain View : aosabook.org, 2012.

42. *Richardson, Leonard.* Beautiful Soup Documentation / Richardson, Leonard // April. — 2007.

43. *team, T. pandas development.* Pandas-Dev/Pandas: Pandas / T. pandas development team. — Zenodo, 02/20/2023.

44. *Bird, S.* Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit / S. Bird, E. Klein, E. Loper. — O'Reilly Media, Inc., 2009.

45. *Chandra, R. V.* Python Requests Essentials / R. V. Chandra, B. S. Varanasi. — Packt Publishing Ltd, 2015.

46. *Krekel, H.* Pytest 7.3 / H. Krekel, B. Oliveira, R. Pfannschmidt, F. Bruynooghe, B. Laugher, F. Bruhin. — 2004.

ПРИЛОЖЕНИЕ А Техническое задание на разрабатываемую систему

УТВЕРЖДЕНО

А.В.00001-01 ТЗ 01

СИСТЕМА ДЛЯ АВТОМАТИЧЕСКОГО СБОРА, АНАЛИЗА И ВИЗУАЛИЗАЦИИ ИНФОРМАЦИИ ПО ЭТИЧНОСТИ КОМПАНИЙ

Техническое задание

Лист утверждения

Подпись и дата	
Инв. № дубл.	
Взам. инв. №	
Подпись и дата	
Инв. № подл.	

Руководитель разработки

_____ Бузмаков А.В.

«___»_____ 2023

Исполнитель

_____ Соломатин Р.И.

«___»_____ 2023

1. Общие сведения

Наименование программы – «Система для автоматического сбора, анализа и визуализации информации по этичности компаний» (далее – «Система»). Основная функция системы - сбор и анализ данных из различных источников, включая новостные сайты, социальные сети, отзывы о компаниях и другие открытые источники данных. Система использует алгоритмы машинного обучения и обработки естественного языка для автоматической обработки данных и определения этичности компаний.

Система также предоставляет визуализацию данных в виде графиков и диаграмм, позволяя пользователям легко понять и сравнивать данные по разным компаниям. Кроме того, система может предоставлять аналитические отчеты и рекомендации по улучшению этичности компаний на основе собранных данных.

Система разрабатывается в рамках выполнения выпускной квалификационной работы. Основанием для разработки являются:

- Положение о курсовой и выпускной квалификационной работе студентов, обучающихся по программам бакалавриата, специалитета и магистратуры в Национальном исследовательском университете «Высшая школа экономики», утвержденным ученым советом НИУ ВШЭ (протокол от 28.11.2014 № 08), с изменениями от 29.03.2016;
- Правила подготовки выпускной квалификационной работы студентов основной образовательной программы бакалавриата «Программная инженерия» по направлению подготовки 09.03.04. Программная инженерия, утвержденные протоколом ученого совета НИУ ВШЭ – Пермь от 19.11.2020 № 8.2.1.7-10/10.

2. Цели и назначение создания автоматизированной системы

2.1. Цели создания АС

Целью создания системы является получение инструмента который позволит анализировать компании на основании их этичности, соответствующего следующим требованиям:

- Показывать историю изменений индекса с возможностью фильтровать по

  - годам;
  - отраслям компаний, с возможностью множественного выбора;
  - компаниям, с возможностью множественного выбора;
  - моделям, с возможностью множественного выбора;
  - источникам, с возможностью множественного выбора.
- Агрегировать значения индекса по годам и кварталам;
- Анализировать тексты для построения индекса этичности;
- Иметь возможность добавления анализа текста несколькими вариантами;
- Сохранять тексты для последующего анализа другими методами;
- Система должна собирать данные с сайтов banki.ru, sravni.ru и комментарии из групп «вконтаке»;
- На сайте должен быть график, который показывать изменение индекса этичности компаний и количества собранных отзывов по разным источникам.
- Для расчета индекса этичности компаний на основании рецензий должна использоваться формулаА.1:

$$\begin{gathered}\text{Base index} = \frac{\text{positive} - \text{negative}}{\text{positive} + \text{negative}} \\ \text{Std index} = \sqrt{\frac{\text{positive}}{\text{negative} \cdot (\text{positive} + \text{negative})^3} + \frac{\text{negative}}{\text{positive} \cdot (\text{positive} + \text{negative})^3}} \\ \text{Index} = (2 \cdot (\text{Base index} - \text{Mean index} > 0) - 1) \cdot \\ max\left(|\text{Base index} - \text{Mean index}| - \text{Std index}, 0\right)\end{gathered} \tag{A.1}$$

positive – количество позитивных предложений,

negative – количество негативных предложений,

Mean index – среднее значения для пар источник сбора данных и модели, которая обрабатывала предложения.

2.2. Назначение АС

Система предназначена для собора и анализа отзывы потребителей с различных веб-сайтов, с помощью алгоритмов обработки естественного языка.

3. Характеристика объекта автоматизации

Система автоматизирует процесс анализа этичности компаний.

4. Требования к автоматизированной системе

Требования к АС:

1. построение графика не должно занимать больше секунды;
2. данные должны собираться автоматически;
3. данные должны обрабатываться автоматически;
4. система должны способна работать с большим объемом информации;
5. система должна быть стабильна.

4.1. Требования к структуре АС в целом

Должно быть несколько модулей, которые общаются между собой с помощью HTTP API:

- сборщики данных;
- взаимодействие с базой данных(API);
- сайт;
- модели для обработки данных.

Все подсистемы должны быть в Docker контейнерах.

4.2. Требования к функциям (задачам), выполняемым АС

4.2.1. Требования к API

Система взаимодействия с базой данных должна выполнять функции:

- хранить информацию о компаниях;
- модель должна отдавать информацию о компаниях из разных сфер;
- хранить информацию о разных источниках;
- добавлять различные источники;
- получать отзывы и разбивать их на предложения;
- иметь возможность отдавать необработанные предложения в зависимости от модели;

- сохранять информацию о моделях;
- сохранять результат обработки предложений;
- агрегировать результат обработки моделей для каждого квартала;
- хранить информацию о состоянии сборщиков данных, если у них возникнут проблемы;
- отдавать информацию об индексе менее, чем за минуту;
- рассчитывать индекс на основе полученных результатов обработки предложений.

4.2.2. Требования к сборщикам данных

Сборщики данных должны выполнять функции:

- собирать отзывы пользователей ежедневно;
- собранные отзывы отправлять на API.

4.2.3. Требования к моделям

Модели должны выполнять функции:

- обрабатывать отзывы пользователей ежедневно;
- обработанные отзывы отправлять на API.

4.3. Требования к видам обеспечения АС

4.3.1. Требования к лингвистическому обеспечению

Система должна соответствовать следующим требованиям:

- Программный код должен быть реализован на языке Python;
- Документация к программе должна быть на русском языке. Других языков не планируется.

4.3.2. Требования к программного обеспечению

Система должна использовать:

- Для разработки API следует использовать библиотеку FastAPI;
- Для взаимодействия с базой данных должна использоваться библиотека SQLAlchemy, а для миграций Alembic;

– Сборщики данных должны собирать информацию с помощью библиотеки запросов requests и для работы с HTML Beautiful Soup;
– Для нейросетевых моделей должен использоваться Pytorch;
– Для клиентской части будет использоваться библиотека React.

4.3.3. Требования к техническому обеспечению

Для работы приложения необходим сервер, который обладает следующими параметрами:

– Процессор с тактовой частотой не ниже 2,5 ГГц, при количестве ядер не менее 4;
– Графическая карта с объемом памяти не менее 4 Гб;
– ОЗУ не менее 16 Гб;
– Не менее 100 Гб свободного места на жестком диске для хранения собранных данных;
– Скорость интернета не менее 100 Мб/с;
– ОС Ubuntu 20.04 и выше;
– Docker 20.10.23 и выше.

4.3.4. Требования к информационному обеспечению

Система должна соответствовать следующим требованиям:

– Система должна использовать PostgreSQL;
– Сервисы между собой должны взаимодействовать при помощи HTTP;
– Для базы данных должен всегда быть резервная копия данных.

4.4. Общие технические требования к АС

4.4.1. Требования к численности и квалификации персонала

Для разработки системы требуется программист со средней квалификацией. Для работы с конечной системой (сайтом), не требуется высокой квалификации, поэтому пользователь с ней справится пользователь, который пользуется сайтами.

4.4.2. Требования к надежности

Надежность системы зависит от надежности функционирования сервера. Устойчивое функционирование программы будет обеспечено с помощью:

- Бесперебойное питание сервера;
- Использованием лицензионного программного обеспечения, необходимого для запуска приложения, включая лицензионную операционную систему;
- Регулярным выполнением рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;
- Регулярным выполнением требований ГОСТ 51188-98 «Защита информации. Испытания программных средств на наличие компьютерных вирусов»;

Время восстановление системы будет до 10 минут с момента сбоя.

4.4.3. Требования по сохранности информации при авариях

Отказы как самой системы, так и ее отдельных функций, могут привести к аварийному завершению работы программы, однако при перезапуске программы ее функциональность не должна пострадать. При таких сбоях программы база данных не должна пострадать. Дополнительно все данные будут резервно копироваться на дополнительную базу данных.

5. Состав и содержание работ по созданию автоматизированной системы

Таблица А.1 – Этапы реализации, контрольные точки проекта

Основной этап	Подэтап	**Крайний срок**
Анализ	Литературный обзор	26.12.2022
Анализ	Сравнительный анализ существующих решений	08.01.2023

(Продолжение на следующей странице)

Продолжение таблицы А.1 – Этапы реализации, контрольные точки проекта

Основной этап	Подэтап	**Крайний срок**
Анализ	Анализ сценариев использования	14.01.2023
Анализ	Написание тех задания	21.01.2023
Проектирование	Проектирование архитектуры приложения	14.02.2023
Проектирование	Проектирование базы данных	20.03.2023
Проектирование	Проектирование графического интерфейса	20.03.2023
Проектирование	Проектирование алгоритмов машинного обучения	20.03.2023
Разработка	Разработка алгоритмов для анализа текста	01.05.2023
Разработка	Реализация серверной части	01.05.2023
Разработка	Реализация клиентской части	01.05.2023
Тестирование	Подготовка тестовых сценариев	15.05.2023
Тестирование	Функциональное тестирование	15.05.2023
Тестирование	Системное тестирование	15.05.2023
Завершение	Сдача проекта	22.05.2023

6. Порядок разработки автоматизированной системы

В разделе «Порядок разработки автоматизированной системы» приводят следующее:

- порядок организации разработки АС;
- перечень документов и исходных данных для разработки АС;
- перечень документов, предъявляемых по окончании соответствующих этапов работ;
- порядок проведения экспертизы технической документации;
- перечень макетов (при необходимости), порядок их разработки, изготовления, испытаний, необходимость разработки на них документации, программы и методик испытаний;
- порядок разработки, согласования и утверждения плана совместных работ по разработке АС;
- порядок разработки, согласования и утверждения программы работ по стандартизации;
- требования к гарантийным обязательствам разработчика;
- порядок проведения технико-экономической оценки разработки АС;
- порядок разработки, согласования и утверждения программы метрологического обеспечения, программы обеспечения надежности, программы эргономического обеспечения.

7. Порядок контроля и приемки автоматизированной системы

Осуществление приемо-сдаточных испытаний для всей системы осуществляется на основе Программы и методики испытаний и включает:

- Функциональное тестирование;
- Тестирование удобства эксплуатации;
- Оценка сгенерированных уровней.

8. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу автоматизированной системы в действие

Таблица А.2 – Требования к программной документации

Название документа	**Краткое содержание**
Текст программы (ГОСТ 19.401–78)	Программный код всех модулей программы с необходимыми комментариями.
Программа и методика испытаний (ГОСТ 19.301–79)	Требования, подлежащие проверке при испытании программы, а также порядок и методы их контроля.
Техническое задание (34.602-2020)	Назначение и область применения программы, технические, технико-экономические и специальные требования, предъявляемые к программе, необходимые стадии и сроки разработки, виды испытаний.

ПРИЛОЖЕНИЕ Б Схема базы данных

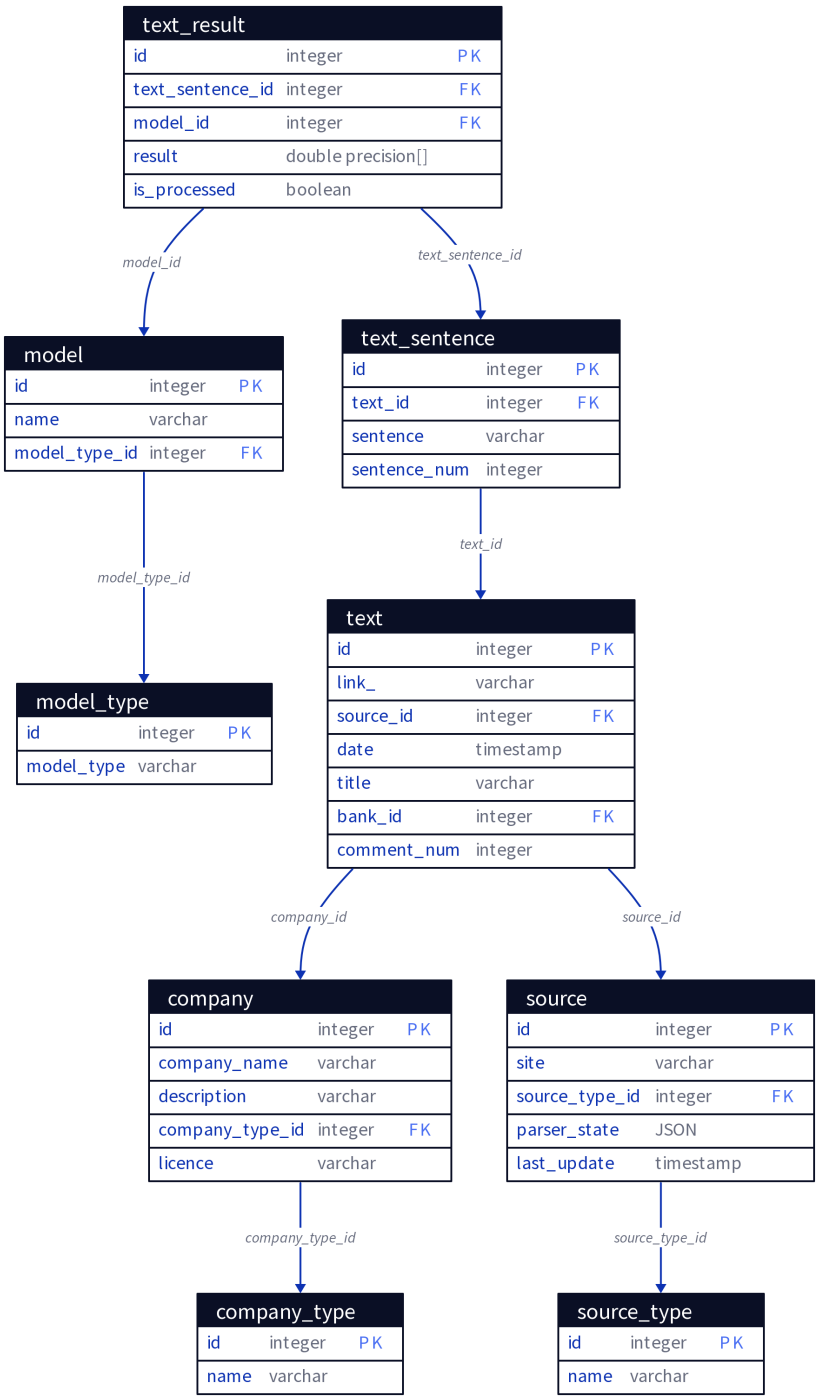


Рисунок Б.1 – Схема базы данных

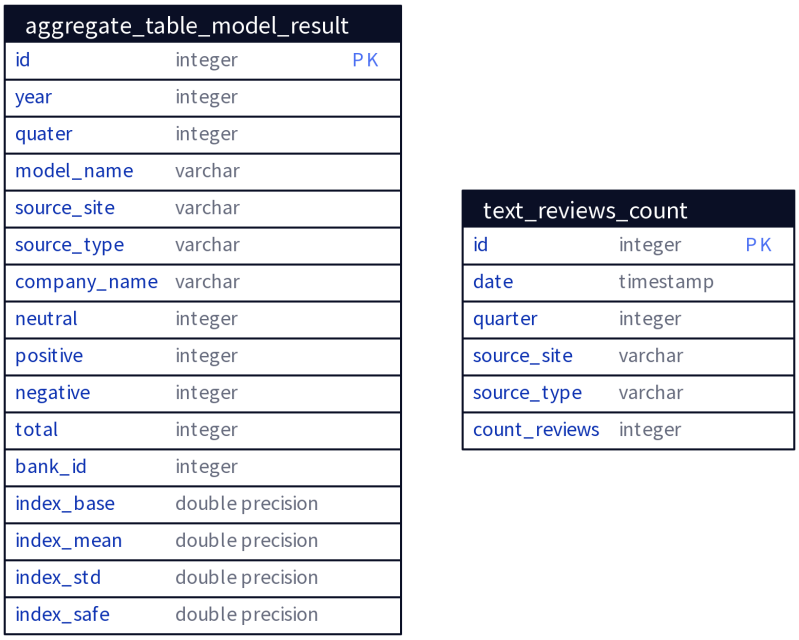


Рисунок Б.2 – Схема базы данных для агрегаций

Рисунок Б.3 – Схема базы данных сайта banki.ru

Рисунок Б.4 – Схема базы данных сайта sravni.ru

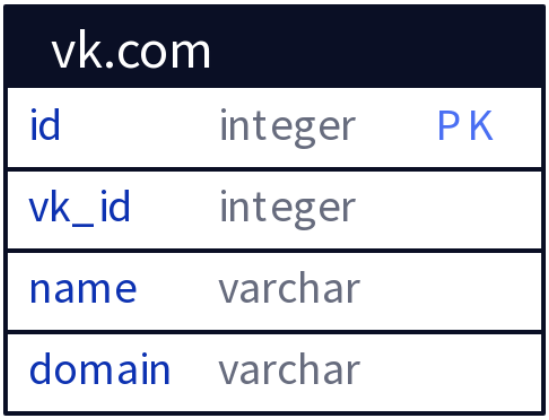


Рисунок Б.5 – Схема базы данных сайта vk.com

ПРИЛОЖЕНИЕ В Запросы API

Таблица В.1 – Запросы API

Путь	**Параметры запроса**	**Тело запроса**	**Результат запроса**

GET
/text/sentences
	sources: array
	model_id: integer
	limit: integer
			{
			items: [{
			sentence_id: integer
			sentence: string
			}]
			}

POST
/text/
		{
		items: [{
		source_id: integer
		date: string
		title: string
		text: string
		bank_id: integer
		link: string
		comments_num:
		integer
		}]
		parser_state: string
		date: string
		}	

(Продолжение на следующей странице)

Продолжение таблицы В.1 – Запросы API

Путь	**Параметры запроса**	**Тело запроса**	**Результат запроса**

GET
/model/
			{
			items: [{
			id: integer
			name: string
			model_type_id:
			integer
			model_type: string
			}]
			}

POST
/model/
		{
		model_name: string
		model_type: string
		}
			{
			model_id: integer
			}

GET
/model/type/
			{
			items: [{
			id: integer
			model_type: string
			}]
			}

(Продолжение на следующей странице)

Продолжение таблицы B.1 – Запросы API

Путь	**Параметры запроса**	**Тело запроса**	**Результат запроса**

GET
/text_-
result/item/{text_-
id}
	text_id: integer		{
			items: [{
			id: integer
			text_sentence_id:
			integer
			result:
			array[number]
			model_id: integer
			}]
			}

POST
/text_result/
		{
		items: [{
		text_result:
		array[number]
		model_id: integer
		text_sentence_id:
		integer
		}]
		}	

(Продолжение на следующей странице)

Продолжение таблицы В.1 – Запросы API

Путь	**Параметры запроса**	**Тело запроса**	**Результат запроса**

GET
/source/
			{
			items: [{
			id: integer
			site: string
			source_type_id:
			integer
			parser_state: string
			last_update: string
			}]
			}

POST
/source/
		{
		site: string
		source_type: string
		}
			{
			id: integer
			site: string
			source_type_id:
			integer
			parser_state: string
			last_update: string
			}

GET
/source/item/{source_-
id}
	source_id: integer		{
			id: integer
			site: string
			source_type_id:
			integer
			parser_state: string
			last_update: string
			}

(Продолжение на следующей странице)

Продолжение таблицы B.1 – Запросы API

Путь	**Параметры запроса**	**Тело запроса**	**Результат запроса**
PATCH /source/item/{source_id}	source_id: integer	{ parser_state: string last_update: string }	{ id: integer site: string source_type_id: integer parser_state: string last_update: string }
GET /source/type/			{ items: [{ id: integer name: string }] }
GET /bank/			{ items: [{ id: integer bank_name: string licence: string description: string }] }

(Продолжение на следующей странице)

Продолжение таблицы B.1 – Запросы API

Путь	**Параметры запроса**	**Тело запроса**	**Результат запроса**

GET
/bank/broker
			{
			items: [{
			id: integer
			bank_name: string
			licence: string
			description: string
			}]
			}

GET
/bank/insurance
			{
			items: [{
			id: integer
			bank_name: string
			licence: string
			description: string
			}]
			}

GET
/bank/mfo
			{
			items: [{
			id: integer
			bank_name: string
			licence: string
			description: string
			}]
			}

(Продолжение на следующей странице)

Продолжение таблицы В.1 – Запросы API

Путь	**Параметры запроса**	**Тело запроса**	**Результат запроса**

GET /views/aggregate_-text_result	start_year: integer
	end_year: integer
	bank_ids: array
	model_names: array
	source_type: array
	aggregate_by_year: boolean
	index_type: None
			{
			items: [{
			year: integer
			quarter: integer
			date: string
			bank_name: string
			bank_id: integer
			model_name: string
			source_type: string
			index: number
			index_10_-percentile: number
			index_90_-percentile: number
			}]
			}

GET /views/reviews_-count	start_date: string
	end_date: string
	source_sites: array
	aggregate_by: None
			{
			items: [{
			date: string
			source_site: string
			source_type: string
			count: integer
			}]
			}

ПРИЛОЖЕНИЕ Г Диаграмма классов

Рисунок Г.1 – Схема классов сборщиков данных

97

ПРИЛОЖЕНИЕ Д Листинг программы

В данном документе представлено описание структуры репозитория, в котором находится исходный код системы, описанной и разработанной в работе.

Репозиторий программы находится по ссылке: `https://github.com/Samoed/EthicsAnalysis`.

ПРИЛОЖЕНИЕ Е Акт о внедрении

Национальный исследовательский университет
«Высшая школа экономики»

НИУ ВШЭ –
Санкт-Петербург

О внедрении
выпускной квалификационной работы

Справка о результатах внедрения
выпускной квалификационной работы

Выпускная квалификационная работа студента группы ПИ-19-1 НИУ ВШЭ – Пермь Соломатина Романа Игоревича по теме «Разработка системы для автоматического сбора, анализа и визуализации информации по этичности компаний» успешно внедрена в рамках выполнения научно-исследовательского проекта НИУ ВШЭ «Инструменты управления этикой бизнеса: измерение этичности и социальной ответственности компании, обучение этичности в компании и механизм саморегулирования этики на рынке» (ТЗ-142).

При выполнении выпускной квалификационной работы достигнуты научно-практические результаты, значимые для реализации проекта ТЗ-142:

1. Разработано программное обеспечение (ПО) для автоматического сбора данных и анализа этичности компаний.
2. Проведена интеграция разработанного ПО в результаты проекта ТЗ-142.
3. Проведено тестирование и подготовлена документация на ПО для автоматического сбора данных и анализа этичности компаний.

Руководитель проекта ТЗ-142 М.А. Сторчевой

Заместитель директора
по научной деятельности
НИУ ВШЭ – Санкт-Петербург Л.А. Цветкова

29.05.2023

Исп.: Тарасов А.М. amtarasov@hse.ru