



## Задание на НИР

Студенту гр. М24-534  
(группа)

Носов Артём Иванович  
(ф.и.о.)

### ТЕМА УИР

«Программная реализация сервиса для автоматизации отбора кандидатов на основе обработки естественного языка из резюме и вакансий»

### ЗАДАНИЕ

№ п/п	Содержание работы	Форма отчетности	Срок исполнения	Отметка о выполнении Дата, подпись
1.	<b>Аналитическая часть. Результаты изучения проблематики построения сервисов для отбора кандидатов</b>			
1.1.	Место сервисов для отбора кандидатов в процессе рекрутмента	Аналитический отчет, список литературы	08.09.2024	
1.2.	Анализ решения задачи отбора кандидатов на основе обработки естественного языка из резюме и вакансий	Аналитический отчет, список литературы	15.09.2024	
1.3.	Сравнительный анализ моделей обработки естественного языка для отбора кандидатов	Аналитический отчет, список литературы	22.09.2024	
1.4.	Выбор модели обработки естественного языка для отбора кандидатов	Аналитический отчет, список литературы	29.09.2024	
2.	<b>Теоретическая часть. Результаты математического и бизнес-моделирования</b>			
2.1.	Разработка математической модели отбора кандидатов на основе обработки естественного языка из резюме и вакансий	Описание архитектуры (математическая модель)	06.10.2024	
2.2.	Результаты экспериментов с моделью на тестовых данных	Отчет о результатах, программный код	13.10.2024	
2.3.	Бизнес-моделирование AS IS и TO BE рекрутмента с сервисом отбора кандидатов	Описание архитектуры (бизнес-модель)	20.10.2024	
2.4.	Разработка функциональных требований к целевому сервису (UML/BPMN диаграммы деятельности, состояний, диаграммы использований (use cases))	Описание архитектуры (UML/BPMN-диаграмма)	27.10.2024	
3.	<b>Инженерная часть. Результаты системного проектирования целевого сервиса</b>			

3.1.	Выбор и обоснование паттерна проектирования (DDD, BDD и т.п.)	Отчет с обоснованием выбранного паттерна	03.11.2024	
3.2.	Выбор и обоснование архитектуры сервиса (верхнеуровневая, REST, диаграммы компонентов, последовательностей и т.п.)	Отчет с обоснованием выбранной архитектуры	10.11.2024	
3.3.	Выбор и обоснование реляционной СУБД для хранения резюме и вакансий	Отчет с обоснованием выбранной СУБД	17.11.2024	
3.4.	Разработка системных требований к целевому сервису	Системные требования	24.11.2024	
4.	<b>Технологическая и практическая часть. Результаты программной реализации целевого сервиса</b>			
4.1.	Выбор и обоснование жизненного цикла разработки (водопад, инкремент и т.п.)	Отчет с обоснованием выбранного цикла	01.12.2024	
4.2.	Выбор и обоснование инструментов разработки (IDE, ЯП, фреймворки)	Отчет с обоснованием выбранных инструментов	08.12.2024	
4.3.	Результаты тестирования (функциональное, модульное, интеграционное, нагрузочное)	Отчет о тестировании, программный код	15.12.2024	
4.4.	Примеры работы прототипа на реальных данных	Программный код	22.12.2024	
5.	<b>Оформление пояснительной записки (ПЗ) и иллюстративного материала для доклада.</b>	Текст ПЗ, презентация		

## ЛИТЕРАТУРА

1.	Sandro Skansi Introduction to Deep Learning. - Cham: Springer International Publishing AG, 2018. - 191 с.
2.	Jacob Devlin, undefined., et al, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2019.
3.	Trevor Hastie, Robert Tibshirani, Jerome Friedman The Elements of Statistical Learning. - 1 изд. - Стэнфорд: Springer, 2017. - 764 с.
4.	Stephen A. Introduction to BPMN. - 1 изд. - Армонке: IBM Corporation, 2006. - 78 с.
5.	Eric Evans Domain-Driven Design: Tackling Complexity in the Heart of Software. - 1 изд. - Москва: Вильямс, 2015. - 446 с.
6.	Y. A. Malkov and D. A. Yashunin Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs // IEEE Transactions on Pattern Analysis and Machine Intelligence. - 2020. - №4. - С. 824-836.
7.	Wang, S., Wang, Y., Sivrikaya, F. Data science for next-generation recommender systems // International Journal of Data Science and Analytics. - 2023. - №16. - С. 135-145.
8.	Aurélien Géron NLP using Transformer Architectures. - 1 изд. - Mountain View: TensorFlow, 2019. - 107 с.
9.	Fabio Petroni Introduction to Recommender Systems. - 1 изд. - Рим: Sapienza University of Rome, 2014. - 66 с.
10.	Pablo Castells, undefined. Dietmar Jannach, "Recommender Systems: A Primer," 2023.
11.	Aggarwal, C.C. (2016). Evaluating Recommender Systems. In: Recommender Systems. Springer, Cham
12.	Amanda Kau, undefined., et al, "Combining Knowledge Graphs and Large Language Models," 2024.

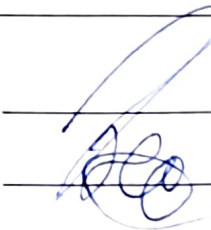
13.	"Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER), pages 39–46 Hong Kong, November 3, 2019. c 2019 Association for Computational Linguistics"
14.	Dong Shu, undefined., et al, "Knowledge Graph Large Language Model (KG-LLM) for Link Prediction," 2024.
15.	Yuqi Zhu, Xiaohan Wang, Jing Chen LLMs for Knowledge Graph Construction and Reasoning: Recent Capabilities and Future Opportunities // World Wide Web. - 2024. - №58. - С. 58-87.
16.	Nicolas Hubert, undefined., et al, "PyGraft: Configurable Generation of Synthetic Schemas and Knowledge Graphs at Your Fingertips," 2024.

Дата выдачи задания:

« 18 » сентября 2024 г.

Руководитель

Студент



Киреев Василий Сергеевич  
(ФИО)

Носов Артём Иванович  
(ФИО)



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**Национальный исследовательский ядерный университет «МИФИ»**

---



**Институт интеллектуальных кибернетических систем**

**Кафедра кибернетики (№ 22)**

Направление подготовки 09.04.04 Программная инженерия

**Пояснительная записка**

к научно-исследовательской работе студента на тему:

**Программная реализация сервиса для автоматизации отбора  
кандидатов на основе обработки естественного языка из резюме и  
вакансий**

---

Группа M24-534

Студент

(подпись)

Носов Артём  
Иванович

(ФИО)

Руководитель

(0-20 баллов)

(подпись)

Киреев Василий  
Сергеевич

(ФИО)

Научный консультант

(0-20 баллов)

(подпись)

(ФИО)

**Москва 2024**

## Оглавление

Реферат.....	2
Введение.....	3
1 Результаты изучения проблематики построения сервисов для отбора кандидатов.....	4
1.1 Место сервисов для отбора кандидатов в процессе рекрутмента.....	4
1.2 Анализ решения задачи отбора кандидатов на основе обработки естественного языка из резюме и вакансий.....	4
1.3 Сравнительный анализ моделей обработки естественного языка для отбора кандидатов.....	5
1.4 Выбор модели обработки естественного языка для отбора кандидатов.....	6
1.5 Вывод.....	8
1.6 Задачи научно-исследовательской работы.....	8
2 Результаты математического и бизнес-моделирования.....	9
2.1 Разработка математической модели отбора кандидатов на основе обработки естественного языка из резюме и вакансий.....	9
2.2 Результаты экспериментов с моделью на тестовых данных.....	11
2.3 Бизнес-моделирование AS IS и TO BE рекрутмента с сервисом отбора кандидатов..	15
2.4 Разработка функциональных требований к целевому сервису.....	16
2.5 Вывод.....	17
3 Результаты системного проектирования целевого сервиса.....	19
3.1 Выбор и обоснование паттерна проектирования.....	19
3.2 Выбор и обоснование архитектуры сервиса.....	19
3.3 Выбор СУБД и его обоснование.....	24
3.4 Разработка системных требований к целевому сервису.....	25
3.5 Вывод.....	26
4 Результаты программной реализации целевого сервиса.....	28
4.1 Выбор и обоснование жизненного цикла разработки.....	28
4.2 Выбор и обоснование инструментов разработки.....	28
4.3 Результаты тестирования.....	29
Выводы.....	30
4.4 Примеры работы прототипа на реальных данных.....	30
4.5 Вывод.....	31
Заключение.....	33
Источники.....	34
Приложение.....	36
1. Обучение BERT.....	36

## Реферат

Пояснительная записка содержит:

- 33 страницы (41 с приложением)
- 10 рисунков
- 4 таблицы
- 1 приложение
- 30 используемых источников

Ключевые слова: автоматизация отбора кандидатов, обработка естественного языка, BERT, рекрутмент, HR-tech.

Целью данной работы является разработка программного сервиса для автоматизации отбора кандидатов на трудоустройство на основе обработки текстовой информации из резюме и вакансий. Работа охватывает анализ существующих методов, выбор подходящей модели обработки естественного языка, экспериментальное тестирование на данных, а также программную реализацию системы, способной ускорить и улучшить процесс рекрутинга, минимизируя влияние человеческого фактора.

## **Введение**

В современных условиях развития технологий и цифровой трансформации бизнеса эффективное управление человеческими ресурсами становится критически важным для организаций любого масштаба. Одним из ключевых процессов в области управления персоналом является поиск и отбор кандидатов, который часто требует значительных временных и человеческих ресурсов. Проблема усложняется большим количеством резюме и вакансий, представленных в формате естественного языка, что делает процесс отбора сложным для автоматизации с использованием традиционных методов.

Современные методы обработки естественного языка (Natural Language Processing, NLP) открывают новые возможности для автоматизации этого процесса. Они позволяют обрабатывать и анализировать текстовые данные, такие как резюме и описания вакансий, с целью извлечения релевантной информации и сопоставления кандидатов с требованиями работодателей. Это сокращает время поиска и отбора персонала, повышает точность соответствия кандидатов и улучшает общий процесс рекрутинга.

Целью данной курсовой работы является разработка программной реализации сервиса для автоматизации процесса отбора кандидатов на основе обработки резюме и вакансий с использованием методов NLP. В рамках работы будут рассмотрены основные алгоритмы и подходы обработки текстовой информации, а также реализована система, способная анализировать резюме и вакансии, выявлять ключевые навыки и сопоставлять их для предложений наилучших кандидатов.

Задачи работы включают в себя обзор существующих решений в области автоматизации отбора персонала, анализ методов NLP, реализацию алгоритмов для обработки текстовых данных и тестирование разработанной системы на реальных данных.

# **1 Результаты изучения проблематики построения сервисов для отбора кандидатов**

## **1.1 Место сервисов для отбора кандидатов в процессе рекрутмента**

Автоматизированные сервисы для отбора кандидатов занимают ключевое место в современном процессе рекрутмента. Их основное назначение — сокращение времени и ресурсов, затрачиваемых на обработку резюме и сопоставление кандидатов с требованиями вакансий. Такие сервисы действуют как промежуточный этап между рекрутерами и кандидатами, обеспечивая предварительную фильтрацию и оценку соответствия.

Основные этапы процесса рекрутмента, где автоматизированные сервисы для отбора играют важную роль:

1. Формирование пула кандидатов: на первом этапе, когда компании получают большое количество резюме, сервисы для отбора помогают быстро отфильтровать неподходящих кандидатов, основываясь на ключевых критериях, таких как наличие необходимых навыков, опыта работы или образования.

2. Предварительная оценка: системы проводят ранжирование резюме, выделяя наиболее релевантные для конкретной вакансии кандидаты, что позволяет рекрутерам сосредоточить внимание на тех, кто больше всего соответствует требованиям.

3. Сопоставление навыков и требований вакансий: сервисы для отбора используют методы NLP для анализа резюме и вакансий, что позволяет автоматически выявлять ключевые навыки и сопоставлять их с требованиями работодателя. Это ускоряет процесс поиска и повышает точность отбора.

4. Поддержка многоуровневого анализа: современные системы предлагают не только механическое сопоставление ключевых слов, но и более глубокий анализ данных, учитывая контекст, синонимы и смежные навыки, что делает отбор кандидатов более гибким и точным.

5. Оптимизация коммуникации: автоматизированные сервисы также могут способствовать улучшению коммуникации с кандидатами, отправляя автоматические уведомления и запросы на интервью, что улучшает общий опыт кандидатов и ускоряет процесс взаимодействия.

Таким образом, сервисы для автоматизации отбора кандидатов занимают центральное место в цифровизации процесса трудоустройства. Они облегчают задачу первичной фильтрации резюме и снижают нагрузку на отдел кадров, одновременно улучшая качество отбора и повышая точность соответствия кандидатов вакансиям.

## **1.2 Анализ решения задачи отбора кандидатов на основе обработки естественного языка из резюме и вакансий**

Задача автоматизированного отбора кандидатов представляет собой многогранную проблему, включающую обработку текстовых данных в произвольной форме, их анализ и сравнение с требованиями вакансий. Важнейшим элементом решения этой задачи являются методы обработки естественного языка (Natural Language Processing, NLP), которые позволяют извлекать структурированную информацию из неструктурированных текстов резюме и вакансий. Данный раздел анализирует существующие подходы к решению задачи отбора кандидатов, а также ключевые проблемы, связанные с их реализацией.

Обработка естественного языка (NLP) включает в себя широкий спектр методов для анализа и интерпретации текстов. В контексте отбора кандидатов данные методы используются для автоматической обработки резюме и вакансий, которые содержат текстовые описания опыта работы, навыков, образования и других характеристик. Основными этапами обработки данных являются предобработка текстов, включая токенизацию, удаление стоп-слов и нормализацию (лемматизацию и стемминг), что помогает стандартизировать текст для дальнейшего анализа. Например, резюме может содержать



различные формы одного и того же навыка, и задача системы — свести их к общему виду для корректного сопоставления. Извлечение информации из текста, такое как выделение ключевых характеристик кандидата (навыки, опыт, образование), также играет важную роль для структурирования данных и их дальнейшего анализа. Семантическое сопоставление с помощью моделей машинного обучения, таких как Word2Vec или BERT, позволяет учитывать контекст использования слов и анализировать смысловые различия в описаниях кандидатов и вакансий.

Современные системы автоматизированного отбора кандидатов активно используют методы машинного обучения для повышения качества сопоставления. Супервайзинговое обучение на основе размеченных данных позволяет обучать модели предсказывать соответствие кандидатов и вакансий. Эти модели основываются на исторических данных, где пары «кандидат-вакансия» уже сопоставлены рекрутерами, что позволяет системе учиться на примерах. Гибридные системы, которые комбинируют подходы на основе правил с методами машинного обучения, демонстрируют высокую эффективность, так как учитывают как ключевые слова, так и контекстные и семантические аспекты данных.

Тем не менее, при построении таких систем существует ряд проблем и ограничений. Одной из ключевых проблем является нестандартизированность данных, так как резюме и вакансии имеют разные формы представления, что затрудняет их автоматическую обработку. Контекстное понимание текста также является важным вызовом: простое сопоставление ключевых слов может быть недостаточным для точной оценки, так как один и тот же навык может требовать разных уровней компетенции в зависимости от контекста. Оценка «мягких» навыков, таких как лидерство или коммуникабельность, часто становится сложной задачей для автоматических систем, так как эти качества редко явно описываются в резюме. Кроме того, модели глубокого обучения, хотя и демонстрируют высокую точность, могут быть сложными для интерпретации, что снижает их прозрачность для пользователей.

На рынке существуют коммерческие и открытые решения, использующие методы NLP для автоматизации отбора кандидатов. Одним из примеров является сервис HireVue, который анализирует речь кандидатов на основе видеоподготовки с использованием NLP и машинного обучения для оценки их соответствия вакансии. Другие решения, такие как Textkernel и Hiretual, используют технологии NLP для анализа текста вакансий и резюме, извлекая ключевые навыки и сопоставляя их с требованиями работодателей. Эти примеры демонстрируют эффективность использования технологий для автоматизации рекрутинговых процессов, хотя системы сталкиваются с проблемами, связанными с качеством данных и сложностями интерпретации. [1-3]

Анализ существующих подходов показывает, что методы NLP и машинного обучения могут значительно улучшить процесс отбора кандидатов, однако для достижения высоких результатов необходимо учитывать особенности текстовых данных и контекст. Проблемы нестандартизированности, оценка «мягких» навыков и сложности интерпретации выводов остаются актуальными, требуя дальнейшего развития систем. В будущем ожидается улучшение гибкости и точности алгоритмов, а также повышение интерпретируемости решений для рекрутеров.

### **1.3 Сравнительный анализ моделей обработки естественного языка для отбора кандидатов**

Обработка естественного языка (Natural Language Processing, NLP) играет ключевую роль в задачах автоматизированного отбора кандидатов, поскольку позволяет анализировать и сопоставлять текстовые данные резюме и вакансий. Для успешного решения этой задачи используются различные модели NLP, каждая из которых имеет свои сильные и слабые стороны. В этом разделе представлен сравнительный анализ наиболее популярных моделей NLP, применяемых для автоматизации отбора кандидатов.

Одной из первых широко применяемых моделей для обработки текста является Bag of Words (BoW). Этот метод преобразует текст в вектор, где каждый элемент соответствует

частоте появления определенного слова в документе. Преимущество BoW заключается в простоте и легкости реализации, что делает его полезным для задач с ограниченными вычислительными ресурсами. Однако BoW не учитывает порядок слов и контекст их использования, что может привести к потере важной информации, особенно при анализе сложных текстов резюме и вакансий, где контекст играет решающую роль [27].

Для преодоления этого недостатка была разработана модель TF-IDF (Term Frequency-Inverse Document Frequency), которая взвешивает частоту слов с учетом их распространенности в наборе документов. TF-IDF позволяет лучше оценивать значимость слов в контексте конкретной вакансии или резюме, что улучшает точность сопоставления. Однако, как и BoW, TF-IDF игнорирует порядок слов и не может эффективно работать с многозначными словами или синонимами, что ограничивает ее применение в сложных задачах отбора кандидатов [4].

Современные методы обработки текста включают использование Word Embeddings, таких как Word2Vec и GloVe. Эти модели представляют слова в виде векторов, где близкие по смыслу слова имеют схожие векторные представления. Word Embeddings позволяют учитывать контекст слов, что значительно улучшает качество анализа текстов. Например, с помощью Word2Vec система может различать значения слова «разработчик» в контексте веб-разработки и разработки ПО. Word Embeddings активно применяются в задачах отбора кандидатов, так как позволяют более точно сопоставлять навыки и требования вакансий. Тем не менее, модели типа Word2Vec имеют ограничение в том, что они не учитывают полный контекст предложения, а только локальное окружение слов [5].

Для решения проблемы учета контекста предложения были разработаны модели на основе трансформеров, такие как BERT (Bidirectional Encoder Representations from Transformers). BERT анализирует текст двунаправленно, что позволяет модели учитывать как предыдущие, так и последующие слова в предложении. Это делает BERT особенно эффективным для анализа сложных и длинных текстов, что является важным преимуществом при обработке резюме и вакансий, где контекст может значительно влиять на смысл. Например, BERT может различать фразы, такие как «опыт управления проектами» и «проект управления опытом», что сложно для более простых моделей. Однако модели на основе трансформеров, такие как BERT, требуют значительных вычислительных ресурсов и времени на обучение, что может ограничивать их применение в малых компаниях или при обработке больших объемов данных [6].

Еще одним подходом, который набирает популярность в задачах отбора кандидатов, является использование GPT (Generative Pretrained Transformer), который способен генерировать текст на основе входных данных и эффективно учитывать контекст. Модель GPT показывает хорошие результаты в задачах, требующих семантического анализа, и может быть полезной для обработки сложных резюме, содержащих абстрактные описания навыков и опыта. Однако из-за своей генеративной природы GPT иногда может вводить новые элементы в текст, что снижает точность при решении задач сопоставления [7].

Таким образом, выбор модели NLP для автоматизации отбора кандидатов зависит от конкретных требований задачи и доступных ресурсов. Для простых задач с небольшим объемом данных может быть достаточно моделей BoW или TF-IDF, в то время как для анализа сложных текстов и учета контекста предпочтительны модели на основе Word Embeddings или трансформеров, таких как BERT и GPT. Модели на основе трансформеров, несмотря на их высокую точность, требуют значительных ресурсов, что может ограничивать их использование в условиях, где важна скорость обработки данных или ограничены вычислительные мощности.

#### **1.4 Выбор модели обработки естественного языка для отбора кандидатов**

На основе проведенного сравнительного анализа моделей обработки естественного языка для задачи автоматизации отбора кандидатов, можно сделать вывод, что выбор конкретной модели зависит от нескольких ключевых факторов: необходимости глубокого

анализа текста, учёта контекста, точности модели, а также доступных вычислительных ресурсов. Для решения задачи автоматизации отбора кандидатов на основе обработки естественного языка из резюме и вакансий наиболее оправданным выбором является использование современных трансформерных моделей, таких как BERT (Bidirectional Encoder Representations from Transformers).

Рассмотренные ранее традиционные методы, такие как Bag of Words и TF-IDF, несмотря на их простоту и эффективность для базовых задач, обладают существенными ограничениями. Основная проблема этих моделей заключается в отсутствии учета контекста. Для задачи отбора кандидатов, где нужно анализировать сложные тексты резюме и вакансий, такие методы могут оказаться недостаточно точными. Например, при сопоставлении навыков кандидатов и требований вакансий важно понимать, в каком контексте употребляются те или иные слова, а не просто их присутствие в тексте. Традиционные методы не могут учесть семантические различия между синонимами или многозначными словами, что снижает точность отбора [4].

Word Embeddings, такие как Word2Vec и GloVe, значительно улучшают качество анализа текста за счет того, что они способны учитывать семантическую близость слов. Однако данные модели, хоть и хорошо работают с отдельными словами, также сталкиваются с трудностями при анализе целых предложений или абзацев. Для задачи отбора кандидатов важно не только правильно идентифицировать навыки кандидата, но и учитывать контекст их использования. Например, «управление проектами» и «опытный руководитель проекта» могут быть интерпретированы по-разному в зависимости от контекста. Для эффективного анализа таких случаев требуется более продвинутая модель, способная анализировать текст на уровне предложений и понимать контекст слов [5].

BERT, как показано в сравнительном анализе, является одной из лучших моделей для учета контекста и анализа сложных текстов. Это обусловлено тем, что BERT использует двунаправленную трансформерную архитектуру, которая позволяет модели анализировать текст как слева направо, так и справа налево. Это дает ей возможность глубже понимать смысл предложений и различать сложные контекстуальные зависимости, что особенно важно при анализе резюме и вакансий. BERT способен учитывать не только присутствие ключевых слов, но и их взаимосвязь в тексте, что делает его идеальным выбором для задачи отбора кандидатов [6].

Кроме того, BERT демонстрирует отличные результаты в различных NLP-задачах, таких как классификация текста, извлечение сущностей и анализ настроений, что также может быть полезным для автоматизации анализа резюме. Например, с помощью BERT можно точно выделять ключевые навыки кандидатов, их опыт работы и другие важные параметры, что позволяет повысить точность сопоставления с вакансиями. Более того, модель BERT поддерживается и активно развивается сообществом, что обеспечивает наличие множества предобученных моделей, которые могут быть адаптированы под конкретные задачи без необходимости обучения модели с нуля. Это снижает затраты на вычислительные ресурсы и время разработки [6].

Выбор BERT также обоснован его способностью обрабатывать большие объемы данных и работать с длинными текстами, что особенно важно для анализа полных резюме и детальных описаний вакансий. Несмотря на то, что модели на основе трансформеров, такие как BERT, требуют значительных вычислительных ресурсов, это компенсируется их высокой точностью и универсальностью. Более простые модели, такие как TF-IDF или Word2Vec, могут быть недостаточно мощными для анализа сложных текстов, тогда как BERT позволяет достигать более высокого уровня понимания и анализа данных [7].

В результате, на основе проведенного анализа можно сделать вывод, что модель BERT является наиболее подходящим выбором для автоматизации отбора кандидатов на основе обработки естественного языка из резюме и вакансий. Она сочетает в себе точность анализа, способность учитывать контекст и широкие возможности для дальнейшей адаптации под конкретные задачи.

## 1.5 Вывод

Результаты изучения проблематики построения сервисов для автоматизации отбора кандидатов показывают, что данные системы занимают ключевое место в процессе рекрутмента, предоставляя возможность значительно ускорить и улучшить процесс поиска подходящих специалистов. Технологии обработки естественного языка (NLP) играют центральную роль в этих системах, так как они позволяют автоматизировать анализ резюме и вакансий, извлекая из них важную информацию и сопоставляя её с требованиями работодателей. Это позволяет минимизировать влияние человеческого фактора, избежать ошибок при первичном отборе и повысить объективность процесса найма.

Сложности, связанные с разработкой таких сервисов, заключаются в необходимости правильно обрабатывать неструктурированные данные, такие как текст резюме и описание вакансий. Для этого используются различные модели NLP, которые могут учитывать контекст, семантику и многозначность слов. Современные технологии, такие как трансформеры и методы, основанные на глубоких нейронных сетях, например BERT, показывают высокую эффективность в этой области. Однако остаются вызовы, связанные с необходимостью адаптации моделей к специфике индустрии и языка резюме и вакансий.

Также важным аспектом является интеграция таких сервисов в общие процессы трудоустройства. Их успешное применение требует не только технической реализации, но и соответствующей организационной поддержки, что включает обучение специалистов по подбору кадров работе с данными инструментами. Таким образом, автоматизация отбора кандидатов с помощью технологий NLP является перспективным направлением, которое способствует оптимизации работы кадровых служб и повышению эффективности подбора персонала.

## 1.6 Задачи научно-исследовательской работы

Целью данной работы является программная реализация сервиса для автоматизации отбора кандидатов на трудоустройство на основе обработки естественного языка из резюме кандидатов и вакансий работодателей.

Для достижения этой цели определим следующие задачи:

1. Выбрать модели обработки естественного языка для отбора кандидатов
2. Провести эксперименты с моделями на тестовых данных
3. Провести бизнес-моделирование AS IS и TO BE рекрутмента с сервисом отбора кандидатов
4. Разработать функциональные требования к целевому сервису
5. Выбрать и обосновать паттерн проектирования сервиса
6. Выбрать и обосновать архитектуры сервиса
7. Выбрать и обосновать реляционную СУБД для хранения резюме и вакансий
8. Разработать системные требования к целевому сервису
9. Выбрать и обосновать жизненный цикл разработки сервиса
10. Выбрать и обосновать инструменты разработки
11. Провести тестирование сервиса
12. Проверить работу сервиса на реальных данных

## 2 Результаты математического и бизнес-моделирования

### 2.1 Разработка математической модели отбора кандидатов на основе обработки естественного языка из резюме и вакансий

Разработка математической модели для автоматизированного отбора кандидатов на основе обработки естественного языка из резюме и вакансий представляет собой комплексную задачу, включающую несколько ключевых этапов: извлечение информации, её семантическое сопоставление и принятие решений. Основной целью является создание модели, способной эффективно анализировать текстовые данные, выделять ключевые параметры и сравнивать их с требованиями вакансий.

Один из возможных подходов включает использование трансформерных моделей, таких как BERT, которые позволяют учитывать контекст слов в предложении. Модель BERT представляет текст в виде токенов и создает их контекстуальные представления, учитывая как предыдущие, так и последующие слова в предложении. Это значительно улучшает качество сопоставления резюме и вакансий, так как BERT позволяет различать различные значения одного и того же слова в зависимости от контекста. Например, фраза «работал в команде» будет иметь различные векторные представления в зависимости от общего контекста резюме [6].

Таким образом, разработка математической модели для автоматизированного отбора кандидатов требует использования передовых методов обработки естественного языка, таких как трансформеры, для извлечения смысловой информации из текстов резюме и вакансий. Модель должна учитывать контекстуальные зависимости между словами.

У нас есть два множества объектов: резюме и вакансии. При этом у каждого из них есть определённый набор свойств обозначим их следующим образом

$$V(v_1, \dots, v_n) \quad (1)$$

Формула 1. Вакансии

$$R(r_1, \dots, r_k) \quad (2)$$

Формула 2. Резюме

Причём нам известно из предметной области “поиск работы и поиск сотрудников”, что как у вакансий так и у резюме может быть неограниченное число параметров  $n$  и  $k$ , но не менее одного для них обоих это название, например, название может быть «Продуктовый аналитик» и для конкретного резюме и вакансии их параметр «название» может быть равен этому значению «Продуктовый аналитик». У вакансии и у резюме могут быть параметры принадлежащие различным типам, например, число (например, число лет опыта), строка (например, название), булево значение (например, пол). При выборе конкретного однотипного набора допустим вакансий типы для всех параметров должны быть одинаковые (выражаясь формулой  $\forall i, j, k, l$ ; если  $V_k \in \bar{V}$  и  $V_l \in \bar{V}$ , то  $type(V_k[i]) = type(V_l[j])$ ), притом не в любой конкретной реализации будет соблюдаться, что одинаковые названия параметров резюме и вакансии будут одинакового типа. Например, пол можно записать строками «муж» или «жен» либо числовыми значениями 1, 0 и так далее. Поскольку и резюме и вакансии отображают некоторое множество объектов профессий, то могут быть сделаны некоторые преобразования чтобы определить функции вида  $F: R \rightarrow V$  и  $F^{-1}: V \rightarrow R$   $M: V \rightarrow R$

$$P(p_1, \dots, p_n) \quad (3)$$

Формула 3. Профессии

Тогда выходит что специалист службы кадров при составлении вакансии реализует функцию вида  $G: P \rightarrow V$  которая является проекцией множества профессий на множество вакансий. И в свою очередь соискатель при составлении своего резюме может осуществлять отображение  $A: P \rightarrow R$  на основе предпочитаемой профессии составляет резюме.

Тогда мы можем вывести что путём обратных преобразований вакансий и резюме можно получить элементы из множества «Профессии». Но нужно понимать что работодатель может искать соискателя не только на основании его профессиональных качеств но и на основании других признаков, допустим возраста. Обозначим множество  $P^*$  как множества  $P$  которое было дополнено набором признаков характеризующими Профессии и назовем это «Профессионалы». Тогда можно ввести другие функции но суть от этого не измениться так что будем понимать далее под  $P$  именно множество «профессионалы». Тогда ниже представлено преобразование которое позволит найти отображение множества резюме на множество вакансий.

$$V^* = G(A^{-1}(R)), V^* \in V$$

Формула 4. Переход множества резюме в множество вакансий

Притом нужно понимать что множество  $V^*$  не обязано быть эквивалентным множеству  $V$ . Также нужно учитывать что не все соискатели составляют свои резюме на основе желаемой профессии, а значит есть варианты когда резюме не может быть отображено в множество «Профессионалы».

Мы можем при таких вводных для конкретных данных резюме зафиксировать  $n, k$  для резюме и вакансий их типы и связи между ними, например если и в резюме и в вакансии есть опыт работы да ещё и одного и того же типа, то это будет означать, что речь идёт об одной и той же сущности.

Тогда можно сделать модель  $M$ , которая будет на основе конкретных резюме создавать вакансии.

Но задача работы заключается в другом это предисловие нужно было для дальнейших рассуждений. Специалист службы кадров в процессе своей деятельности в том числе занимается сортировкой пришедших резюме на вакансию. Этот процесс представляет из себя последовательность действий, где он для каждого пришедшего резюме проставляет статус «принято» либо «отклонено». Делаем допущение только по двум статусам так как если есть хоть  $q$  статусов которые проставляет кадровик либо в какой то системе или на бумажке то это все степень уверенности между «принято» либо «отклонено». Есть так же возможность ничего не делать либо посмотреть и не проставить статус. Все случаи помимо «принято» либо «отклонено» нас не очень интересуют поскольку основная задача это сделать работу кадровика за него, а именно распределить резюме на две категории для конкретной вакансии «принято» либо «отклонено». Тогда требуемую модель можно представить так

$$M_2(V, R) \in [0, 1]$$

Формула 5. Модель номер два осуществляет переход пары вакансия-резюме в отрезок от нуля до единицы.

Данный отрезок связан со степенью уверенности куда отнести данное резюме для конкретной вакансии. 0 — резюме на данную вакансию моделью отклонено со 100% уверенностью и наоборот 1 — резюме на данную вакансию моделью принято со 100% уверенностью. Значение 0.5 будет означать, что резюме на данную вакансию моделью принято с 50% вероятностью.

Если говорить в терминах предисловия то такая модель реализует функцию кадровика отображать множество «резюме» на множество «профессионалы» и оценивать насколько данный конкретный профессионал который представлен этим резюме схож с тем профессионалом который был отображён на множество вакансий. Тогда запишем отображение модели в терминах предисловия.

$$F(A^{-1}(R), G^{-1}(V)) \in [0, 1] \quad (6)$$

Формула 6. Функция  $F$  реализует отображение двух элементов из множества  $P$  в отрезок между нулём и единицей.



Тогда наша вторая модель как раз таки аппроксимирует некоторую функцию  $F^*=(6)$ . Для того чтобы аппроксимировать  $F^*$  в виде второй модели будет применён BERT. Тогда на основе всех этих вводных ниже можно записать уравнения которые будут характеризовать математическую модель отбора кандидатов на основе обработки естественного языка из резюме и вакансий. Резюме и вакансии будут представлены в основном в виде текста, их текстовую часть обозначим как  $R_t$  и  $V_t$ , то есть текст резюме и текст вакансии.

Трансформеры используют механизм внимания для вычисления весов между токенами текста, позволяя модели учитывать контекст как слева, так и справа. Основу модели составляет механизм само-внимания Self-Attention, который определяется следующим образом:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

Формула 7. Выравнивание внимания

В данной формуле  $Q$  — запросы (queries),  $K$  — ключи (keys),  $V$  — значения (values),  $d_k$  — размерность ключей.

Для задачи анализа вакансий и резюме модель BERT используется в качестве эмбединга текста  $E(R)$  и  $E(V)$  для резюме  $R$  и вакансий  $V$  соответственно.

$$E(R), E(V) = \text{BERT}(R), \text{BERT}(V)$$

Формула 8. Эмбединг текста  $E(R)$  и  $E(V)$

Полученные эмбединги можно использовать для вычисления сходства например через косинусное расстояние.

$$\text{Similarity}(R, V) = \frac{E(R) \cdot E(V)}{\|E(R)\| \cdot \|E(V)\|}$$

Формула 9. Косинусное расстояние

Тогда запишем целевую функцию для обучения модели классификации на основе вероятности принятия резюме используется кросс-энтропия.

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]$$

Формула 9. Кросс-энтропия

В формуле кросс-энтропии  $y_i$  — истинная метка (0 или 1),  $\hat{y}_i$  — предсказанная вероятность принятия резюме.

Математическая модель трансформеров и BERT интегрируется в систему автоматизированного анализа и сопоставления вакансий и резюме через механизм внимания и эмбединги контекста, что позволяет учитывать семантическую значимость текста.

## 2.2 Результаты экспериментов с моделью на тестовых данных

Алгоритм сбора датасета для автоматизации отбора кандидатов предполагает последовательное извлечение, обработку и структурирование данных из резюме и вакансий. Первым этапом осуществляется сбор данных через платформы, где соискатели подают резюме на вакансии, а HR специалисты принимают решения о приглашении или отказе. На втором этапе анализируются события, такие как «приглашен» или «отклонен», чтобы выделить уникальные пары «резюме-вакансия» и устранить дубли, где одно резюме попадало в обе группы. Далее данные очищаются от пропусков и получают актуальные на момент отклика заголовки вакансий и резюме. Завершается процесс преобразованием данных в формат, подходящий для машинного обучения, с учетом баланса классов и конфиденциальности информации

Соискатели присылают свои резюме на вакансии после чего hr может принять/отклонить/ничего не сделать. Рассмотрим только случаи где hr что то сделал так как если hr ничего не сделал то мы не можем однозначно сказать подходит или не подходит данное резюме для вакансии.

Напишем запрос для того чтобы получить события «пригласил» или «отклонил». После чего оценим количество событий.

Тип события	Количество событий	Процентное соотношение количества событий
пригласил	157414	52%
отклонил	142799	48%

Таблица 1 – Количество событий по типам

Суммарно имеем 300213 событий. Есть дисбаланс классов, но не большой, разница 2%.

Теперь посмотрим количество уникальных пар резюме вакансия в каждом типе события «пригласил» или «отклонил».

Тип события	Количество уникальных пар резюме-вакансия	Процентное соотношение количеств уникальных пар резюме-вакансия
пригласил	140284	51%
отклонил	136920	49%

Таблица 2 – Количество событий по типам

Как видно теперь получается более лучший баланс классов, это намекает на то что среди принятых резюме было больше дублей пар вакансия-резюме, так как дисбаланс классов уменьшился на 1% за счёт типа события «пригласил». Оценим сколько пар резюме-вакансия попали в обе группы. Всего уникальных пар 267493, а сумма при распределении на пары резюме-вакансия 277204, что говорит о том что есть 9711 случаев попадания в обе группы. После перепроверки отдельным запросом совпало, что 9711 уникальных пар резюме-вакансия попали в обе группы. Их мы исключим из датасета поскольку они неоднозначно подходили на вакансию эти данные могут добавить дополнительную дисперсию в наш датасет, что нам не нужно. Как один из вариантов когда так происходило, например, hr случайно переместил не в ту папку отклик. Тогда рассчитаем количественные характеристики после того как уберём те варианты которые попали в обе группы.

Тип события	Количество уникальных пар резюме-вакансия	Процентное соотношение количеств уникальных пар резюме-вакансия
пригласил	130578	51%
отклонил	127231	49%

Таблица 3 – Количество событий по типам

Соискатель с одним и тем же резюме может несколько раз откликнуться на одну и ту же вакансию так что будем брать дату и время последнего отклика резюме на вакансию.

Теперь выгрузим датафрейм в формате (resume\_id, vacancy\_id, type\_name, event\_date\_l) где уберём пары резюме-вакансия которые попадают в обе группы и возьмём последнее время отклика, также убираем строки где есть null.

Теперь нужно получить актуальные заголовки резюме и вакансий в момент отклика. После получения актуальных на момент отклика заголовков вакансий получилось уже 193825 записей в датасете. Теперь добавим актуальные на момент отклика заголовки резюме. И после этого получаем 172321 строк. Посмотрим теперь распределение классов.

Тип события	Количество	Процентное соотношение количеств
пригласил	75283	44%
отклонил	97038	56%

Таблица 4 – Количество событий по типам

Теперь есть смещение классов в сторону отклонил на 6%. Теперь датасет можно переносить в csv. Можно отметить, что здесь информационная безопасность обеспечивается с точки зрения данных тем что в названии вакансии или резюме не пишут персональную информацию, а если и пишут в заголовок резюме, то модератор системы её удаляет. Так что никакой опасности утечки персональных данных нет — название вакансии на которую откликнулся человек не является персональной информацией, а название профессии не является информацией по которой можно идентифицировать человека.

После анализа датасета выяснилось, что есть случаи, когда название должности вакансии соответствует названию резюме, но резюме всё равно отклонили это значит, что есть причина отклонения помимо названия вакансии. Это мы учтём при дальнейшей обработки данных. Далее загрузим данные в google collab.

```

import pandas as pd

df = pd.read_csv('nir.csv')
df = df.drop(columns=['Unnamed: 0'])
df

```

	resume_profession	vacancy_profession	vacancy_id	resume_id	type_name	event_date_1
0	Бухгалтер	Заместитель главного бухгалтера, бухгалтер	50100109	7195	отклонил	2024-10-16 21:13:27
1	Бухгалтер	Бухгалтер на первичную документацию, бухгалтер...	50199143	7195	отклонил	2024-11-20 13:03:22
2	Бухгалтер	Бухгалтер на первичную документацию. Помощник ...	30451185	7195	отклонил	2024-11-07 14:39:03
3	Бухгалтер	Бухгалтер	50169348	7195	отклонил	2024-11-05 16:13:36
4	Логистик - аналитик	Менеджер по логистике	49341423	7419	отклонил	2024-11-12 15:26:11
...	...	...	...	...	...	...
172316	Разнорабочий	Военнослужащий по контракту (на СВО)	50230656	55374468	отклонил	2024-12-06 20:42:42
172317	Водитель-экспедитор категории С	Водитель категории С	49464398	55374505	отклонил	2024-12-06 18:35:03
172318	Водитель	Водитель категории В	50216597	55374545	пригласил	2024-12-06 19:48:26
172319	Военнослужащий по контракту	Военнослужащий по контракту (ОХРАНА)	49934402	55374738	пригласил	2024-12-06 20:43:29
172320	Курьер	Курьер	50287980	55374798	отклонил	2024-12-06 20:30:35

172321 rows x 6 columns

Рисунок 1 – Датасет заголовки вакансий и резюме по типу события

Далее был проведён этап токенизации с использованием предобученного токенизатора модели BERT. Текстовые данные были преобразованы в числовой формат, включающий индексы токенов, маски внимания и сегментные индикаторы. После этого данные были разделены на батчи для более эффективной обработки на GPU.

На этапе обучения была инициализирована модель BERT с головой классификации. Обучение проводилось с использованием оптимизатора AdamW и планировщика обучения с линейным уменьшением скорости обучения. Для контроля переобучения использовались метрики точности на валидационном наборе данных. После завершения обучения модель показала улучшенные результаты классификации на тестовых данных. В приложении 1 есть подробный листинг процесса обучения BERT.

	Training Loss	Valid. Loss	Valid. Accur.	Training Time	Validation Time
epoch					
1	0.52	0.49	0.74	0:37:32	0:01:21
2	0.46	0.47	0.76	0:37:27	0:01:21
3	0.42	0.47	0.76	0:37:29	0:01:21
4	0.39	0.49	0.76	0:37:28	0:01:22

Рисунок 2 – BERT fine-tuning для классификации по эпохам



Рисунок 3 – график BERT fine-tuning для классификации по эпохам

Как видно на графике переобучение наступает после второй эпохи обучения, что говорит о том, что достаточно двух эпох, чтобы сделать fine-tuning модели на наших данных.

В итоге получена модель обладающая точностью классификации на два класса составляющая по метрике ассигуры 0.75.

Также в качестве метрики был взят Коэффициент корреляции Мэтьюса (Matthews correlation coefficient, сокращенно MCC [29]). Используют данную метрику, так как у нас есть несбалансированность классов в данных. Получаем MCC на всей выборке равный 0.550, что говорит о том, что модель предсказывает класс не случайным образом. MCC принимает значения между -1 и 1, где 0 означает случайное предсказание оно не зависит от входных данных -1, поставили всем семплам неверные метки, а 1 наоборот верные.

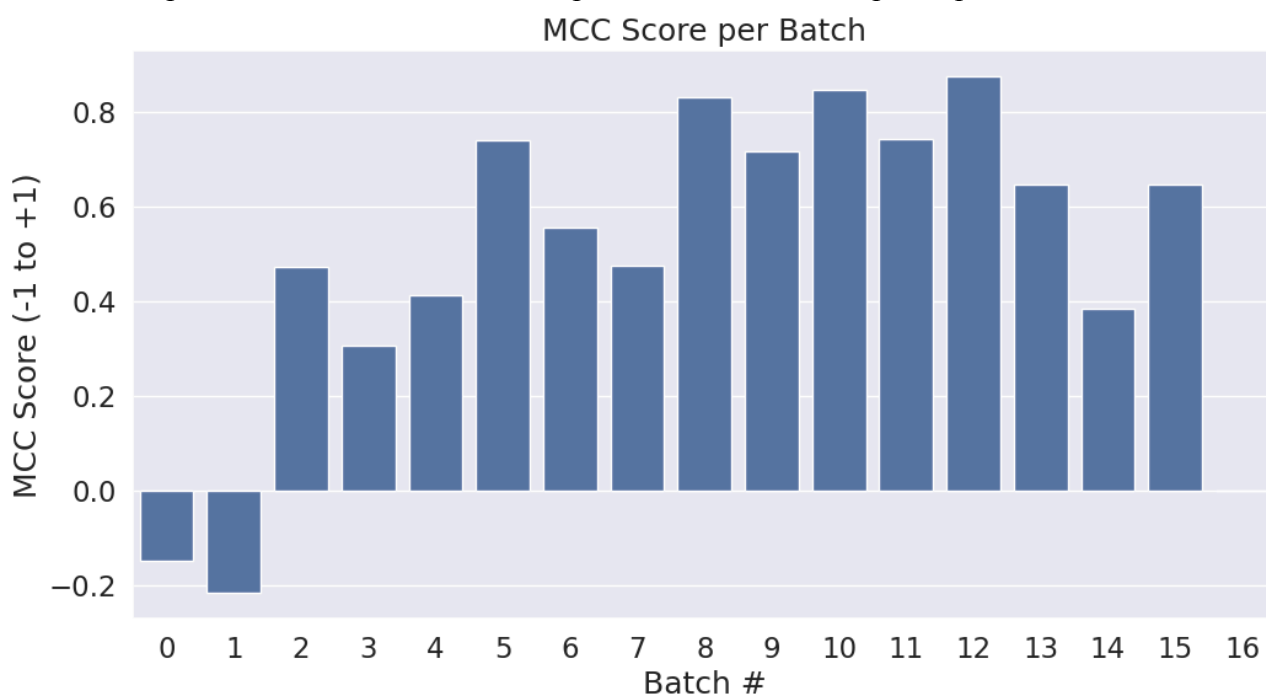


Рисунок 4 – диаграмма метрика MCC по батчам тестового датасета

## 2.3 Бизнес-моделирование AS IS и TO BE рекрутмента с сервисом отбора кандидатов

Бизнес-моделирование процессов рекрутмента позволяет наглядно представить текущее состояние системы (AS IS) и определить, каким образом внедрение автоматизированного сервиса отбора кандидатов на основе обработки естественного языка может изменить и оптимизировать процесс (TO BE). Данный подход помогает выявить неэффективности текущего процесса, предложить улучшения и обосновать экономические и организационные преимущества автоматизации.

В модели AS IS традиционного рекрутмента процесс отбора кандидатов обычно включает несколько этапов, которые требуют значительных затрат времени и человеческих ресурсов. На начальных этапах рекрутеры размещают вакансию на различных платформах и получают большой объем откликов. Затем следует процесс ручной фильтрации резюме, при котором специалист по подбору кадров тратит время на прочтение и анализ каждого резюме. Этот этап является наиболее трудоемким, так как он предполагает оценку релевантности кандидатов по нескольким параметрам: опыт работы, образование, навыки и соответствие требованиям вакансии. Ошибки при ручной фильтрации, связанные с человеческим фактором, могут привести к упущению подходящих кандидатов или, наоборот, к приглашению на собеседование кандидатов, которые не соответствуют требованиям [8].

Кроме того, в текущей модели рекрутеры часто сталкиваются с проблемой неконсистентности и неструктурированности информации в резюме. Кандидаты могут по-разному описывать свои навыки и опыт, что затрудняет сравнение их профилей с требованиями вакансии. Всё это увеличивает время на принятие решения и может привести к повышению затрат на процесс найма, что является значительным недостатком текущей системы [9].

В модели TO BE, с внедрением автоматизированного сервиса для отбора кандидатов, многие из вышеперечисленных проблем могут быть решены за счет применения технологий обработки естественного языка. Система автоматически анализирует большое количество резюме и вакансий, используя такие технологии, как векторизация текста и модели машинного обучения для сопоставления навыков кандидатов с требованиями вакансий. Это значительно ускоряет процесс первичного отбора и снижает нагрузку на рекрутеров. Время, которое ранее тратили на ручную фильтрацию, сокращается, и рекрутеры могут сосредоточиться на более высокоуровневых задачах, таких как проведение собеседований и оценка соответствия мягкий навыков кандидатов компании [6].

Одним из ключевых преимуществ автоматизированного сервиса является его способность учитывать контекст и семантику текста. Модель, такая как BERT, позволяет не просто искать ключевые слова в резюме, но и понимать их смысл в зависимости от контекста, что делает процесс сопоставления кандидатов с вакансиями более точным и эффективным. Например, если в вакансии требуется знание определенного инструмента, система сможет найти всех кандидатов, которые имеют соответствующий опыт, даже если они описали это различными способами [10].

Кроме того, автоматизация рекрутмента с использованием NLP-сервисов способствует стандартизации процесса и минимизации ошибок, связанных с человеческим фактором. Сервис может не только фильтровать резюме, но и автоматически ранжировать кандидатов по уровню соответствия вакансии, предоставляя рекрутеру уже отсортированный список кандидатов. Это значительно снижает вероятность упущения подходящих кандидатов и ускоряет процесс принятия решений [5].

Модель TO BE с внедрением автоматизации также повышает прозрачность процесса отбора. Система может хранить все данные по результатам анализа резюме, предоставляя рекрутеру возможность объективно оценить, почему тот или иной кандидат был отобран или отклонен. Это особенно важно в условиях высокого объема откликов, когда ручной анализ может быть недостаточно тщательным.

Таким образом, внедрение автоматизированного сервиса для отбора кандидатов на основе обработки естественного языка значительно улучшает процесс рекрутмента, ускоряет

его, снижает затраты и минимизирует ошибки, связанные с человеческим фактором. Это позволяет компаниям не только повысить эффективность подбора персонала, но и оптимизировать использование ресурсов, обеспечивая более точный и качественный отбор кандидатов.

Ниже представлены упрощённые диаграммы AS IS и TO BE для процесса фильтрации резюме пришедших на вакансию.

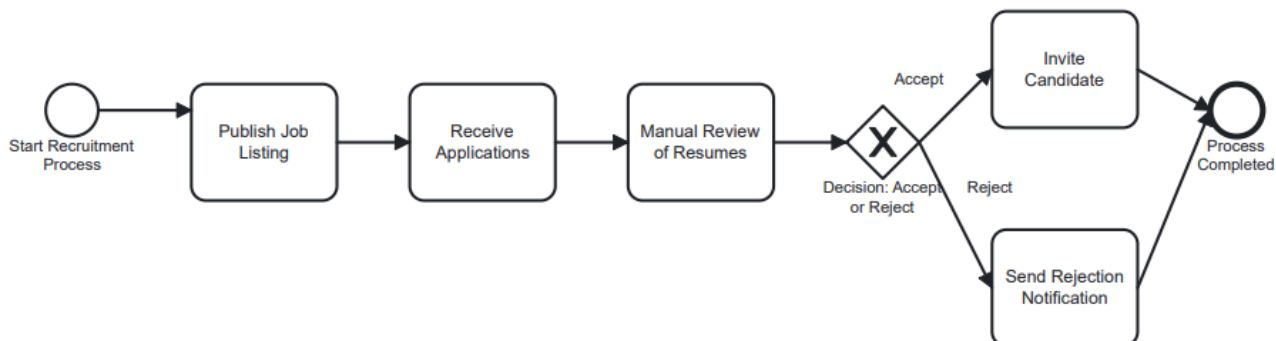


Рисунок 5 – AS IS диаграмма процесса фильтрации резюме пришедших на вакансию



Рисунок 6 – TO BE диаграмма процесса фильтрации резюме пришедших на вакансию

## 2.4 Разработка функциональных требований к целевому сервису

Разработка функциональных требований к целевому сервису является важным этапом процесса создания системы для автоматизации отбора кандидатов на основе обработки естественного языка. Основная задача функциональных требований заключается в описании функциональности, которую должна предоставлять система, чтобы соответствовать поставленным бизнес-целям и удовлетворять запросы пользователей. Для формализации этих требований используются различные подходы, такие как диаграммы UML (Unified Modeling Language) и BPMN (Business Process Model and Notation), которые позволяют визуализировать процессы и взаимодействие пользователей с системой.

Начнем с описания функциональных требований с помощью диаграмм деятельности (BPMN), которые представляют собой моделирование бизнес-процессов, связанных с автоматизацией отбора кандидатов. Основной поток деятельности начинается с этапа публикации вакансии работодателем. Система автоматически получает информацию о вакансии и приступает к сбору резюме кандидатов с различных платформ. Следующим шагом является анализ собранных резюме с использованием методов обработки естественного языка. В этом процессе система извлекает ключевые данные из резюме (опыт работы, навыки, образование) и сопоставляет их с требованиями вакансии. Результатом этого этапа является формирование списка кандидатов, упорядоченных по уровню соответствия [11].

Для более детального понимания системы можно использовать диаграмму состояний (UML), которая описывает жизненный цикл вакансии и кандидатов в системе. Диаграмма начинается с состояния "Вакансия создана", после чего переходит в состояние "Поиск кандидатов", в котором система активно анализирует поступившие резюме. По завершении анализа состояние изменяется на "Список кандидатов сформирован", после чего система может отправить уведомления рекрутерам. Каждый кандидат в свою очередь также проходит ряд состояний — от "Кандидат найден" до "Кандидат проанализирован" и "Кандидат



соответствует требованиям", что иллюстрирует динамику взаимодействия данных внутри системы.

Также важную роль играет диаграмма use cases (диаграмма прецедентов), которая позволяет описать взаимодействие между пользователями (акторами) и системой. Основными акторами в данной системе выступают рекрутер, кандидат и система анализа резюме.



Рисунок 7 — диаграмма прецедентов

Диаграмма прецедентов позволяет выделить следующие ключевые сценарии использования системы:

1. рекрутер может создавать и редактировать вакансию;
2. система анализирует поступающие резюме;
3. кандидат может подавать свое резюме для рассмотрения;
4. ранжирование кандидатов;
5. отправку уведомлений рекрутерам;
6. обновление данных вакансий.

Каждый из этих сценариев иллюстрирует функциональные возможности системы и роли пользователей в процессе взаимодействия с ней [12].

Для описания требований к системе также важно учитывать нефункциональные аспекты, такие как производительность и масштабируемость. Система должна быть способна обрабатывать большое количество резюме и вакансий за минимальное время, особенно при большом объеме данных. Кроме того, необходимо предусмотреть интеграцию с внешними системами, такими как платформы поиска работы и базы данных резюме, что требует разработки интерфейсов для обмена данными между различными источниками [13].

## 2.5 Вывод

Результаты математического и бизнес-моделирования демонстрируют значительный потенциал автоматизации отбора кандидатов на основе обработки естественного языка для повышения эффективности процесса рекрутмента. Математическое моделирование, включающее применение современных алгоритмов обработки текста, таких как модели на основе трансформеров (например, BERT), показало, что автоматизированная система способна эффективно анализировать резюме и сопоставлять их с вакансиями на основе семантического анализа. Это позволяет улучшить точность отбора кандидатов, минимизируя влияние человеческого фактора и снижая вероятность ошибок, связанных с субъективной оценкой кандидатов.

С точки зрения бизнес-моделирования, внедрение автоматизированного сервиса для отбора кандидатов значительно оптимизирует текущие процессы рекрутмента. Модель AS IS показала, что традиционные методы отбора, основанные на ручной фильтрации резюме, требуют значительных затрат времени и ресурсов, что снижает общую эффективность бизнеса. Внедрение автоматизированной системы, как показала модель TO BE, позволяет сократить время обработки резюме и улучшить качество подбора кандидатов. Это не только снижает затраты на рекрутинг, но и повышает удовлетворенность рекрутеров и менеджеров по найму за счет упрощения процесса и повышения его прозрачности.

Таким образом, результаты моделирования подтверждают необходимость и целесообразность внедрения автоматизированного сервиса на основе обработки естественного языка для решения задачи отбора кандидатов. Это не только улучшает технологический процесс, но и приносит ощутимую выгоду для бизнеса, сокращая

временные и финансовые затраты на рекрутмент, а также повышая качество принимаемых решений при выборе кандидатов.

## 3 Результаты системного проектирования целевого сервиса

### 3.1 Выбор и обоснование паттерна проектирования

При разработке программного обеспечения для автоматизации отбора кандидатов на основе обработки естественного языка выбор паттерна проектирования играет ключевую роль, поскольку определяет структуру системы и подход к её реализации. Для данного проекта наилучшим образом подходит паттерн Domain-Driven Design (DDD), который позволяет создать архитектуру системы, ориентированную на предметную область, с акцентом на её сложность и требования.

DDD фокусируется на разделении предметной области на отдельные домены, каждый из которых имеет свою модель, отражающую ключевые бизнес-процессы и правила. В случае сервиса для автоматизации отбора кандидатов основными доменами являются кандидаты, вакансии, процесс сопоставления и аналитика результатов отбора. Каждая из этих областей может быть описана как отдельная сущность с уникальными характеристиками, что позволяет лучше структурировать систему и упростить взаимодействие между её компонентами [14]. DDD также способствует четкому разграничению ответственности между различными частями системы и помогает избежать избыточной зависимости между компонентами.

Выбор DDD обоснован также тем, что автоматизация процесса отбора кандидатов требует глубокого понимания предметной области и её постоянной эволюции. В DDD ключевое внимание уделяется гибкости и адаптируемости системы, что особенно важно в условиях постоянно меняющихся бизнес-требований в сфере рекрутмента. Например, система должна легко адаптироваться к изменению критериев отбора или добавлению новых метрик анализа резюме, не нарушая при этом основную логику работы [14].

Кроме того, применение DDD позволяет создать единое языковое пространство (ubiquitous language) между бизнес-экспертами и разработчиками, что значительно облегчает коммуникацию и разработку системы. В процессе создания системы для автоматизации отбора кандидатов специалисты по рекрутингу могут напрямую участвовать в проектировании модели, так как все ключевые сущности и процессы системы отражают их терминологию и бизнес-логику. Это способствует созданию системы, которая не только решает технические задачи, но и соответствует реальным потребностям бизнеса [15].

Альтернативный подход, такой как Behavior-Driven Development (BDD), также может быть применен для тестирования и разработки системы. BDD помогает описывать поведение системы на языке, понятном бизнес-пользователям, что облегчает процесс валидации функциональности. Однако, в данном проекте ключевая сложность заключается не столько в поведении системы, сколько в глубокой предметной модели. Поэтому DDD выступает в качестве основной методологии для проектирования, в то время как BDD может быть использован для формализации требований и тестирования [16].

Таким образом, выбор DDD как основного паттерна проектирования для разработки системы автоматизации отбора кандидатов обоснован необходимостью глубокого понимания предметной области и требованиями к гибкости системы. DDD позволяет создать архитектуру, которая легко адаптируется к изменению бизнес-логики, при этом сохраняя прозрачность и структурированность. Использование BDD в качестве вспомогательного подхода поможет дополнительно обеспечить качество системы через тестирование её поведения на уровне требований.

### 3.2 Выбор и обоснование архитектуры сервиса

При выборе и обосновании архитектуры для сервиса автоматизации отбора кандидатов на основе обработки естественного языка необходимо учитывать требования к масштабируемости, производительности и интеграции с внешними системами. Для данного проекта наилучшим образом подходит микросервисная архитектура, которая обеспечит

гибкость и возможность независимого развития отдельных компонентов системы. Основным преимуществом микросервисной архитектуры является её модульность, что позволяет разделить систему на независимые сервисы, каждый из которых отвечает за выполнение определенной бизнес-логики, такой как обработка резюме, анализ вакансий, сопоставление кандидатов и управление пользователями [17].

На верхнеуровневом уровне система состоит из нескольких ключевых компонентов: сервиса управления резюме, сервиса управления вакансиями, сервиса анализа и сопоставления и сервиса пользователей. Каждый из этих сервисов взаимодействует через REST API, что позволяет обеспечивать гибкость и независимость их разработки и развертывания. RESTful API является подходящим выбором для взаимодействия между компонентами, поскольку он хорошо масштабируется, легко интегрируется с внешними системами и поддерживает разнообразные форматы передачи данных, такие как JSON, что делает его идеальным для веб-приложений [18].

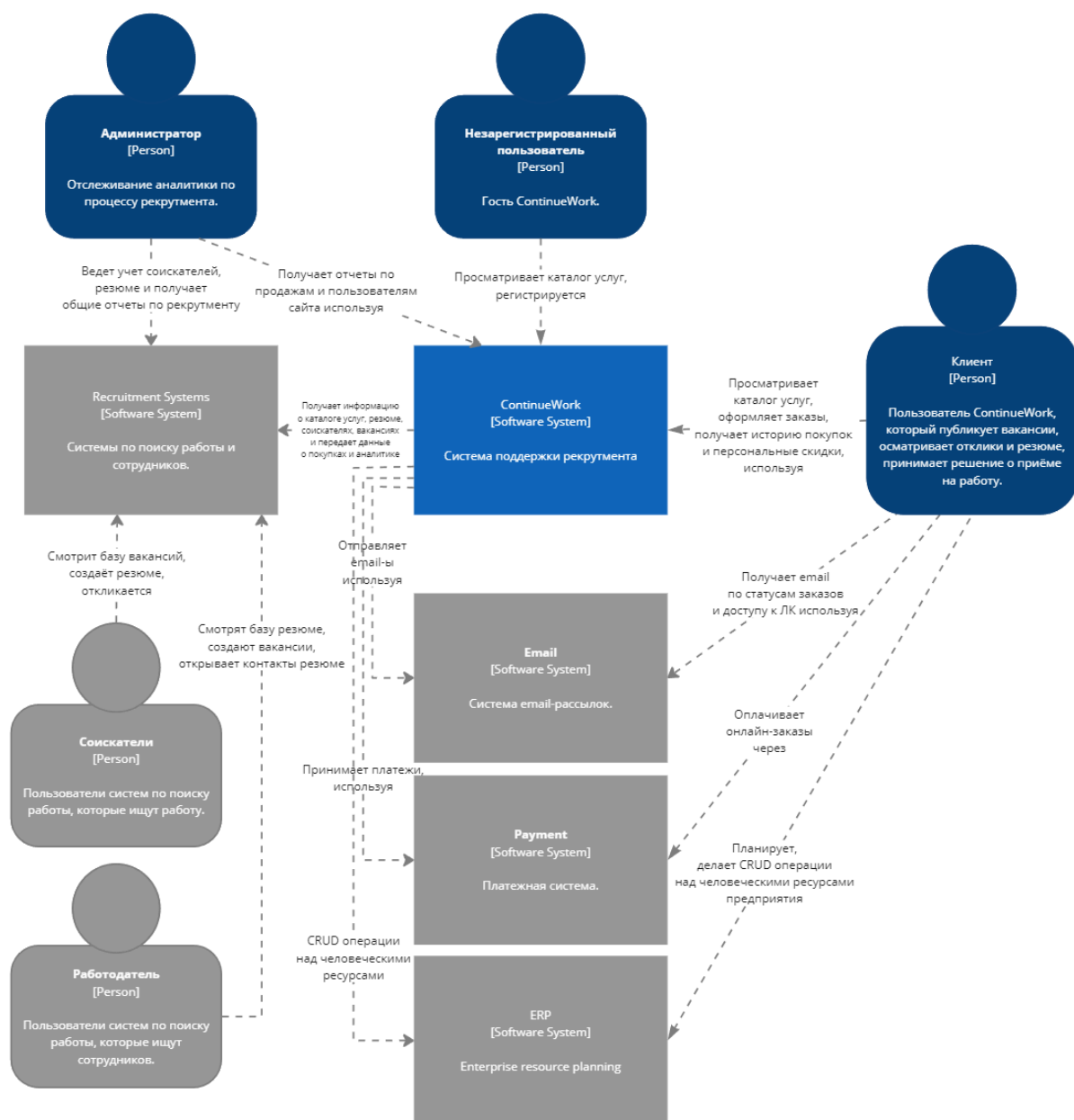
Важным элементом архитектуры является диаграмма компонентов, которая описывает основные части системы и их взаимодействие. Основные компоненты включают серверных компонентов, выполняющих бизнес-логику, такие как модуль анализа резюме и модуль сопоставления с вакансиями. Эти компоненты взаимодействуют через API-шлюз, который контролирует потоки запросов и маршрутизирует их к соответствующим сервисам. В случае необходимости компоненты могут обращаться к внешним сервисам, таким как базы данных или системы машинного обучения для анализа текстов [19].

Для более детального описания процессов взаимодействия компонентов системы можно использовать диаграмму последовательностей. Например, процесс сопоставления кандидата с вакансией начинается с того, что вызывается сервис вакансий через REST API. Далее система отправляет запрос к сервису анализа резюме для получения списка подходящих кандидатов. Модуль анализа на основе моделей обработки естественного языка анализирует данные и возвращает список кандидатов, упорядоченных по уровню соответствия, после чего сервис вакансий сохраняет результат и предоставляет его рекрутеру через пользовательский интерфейс [20].

Модульная структура, основанная на микросервисной архитектуре и RESTful API, также позволяет системе быть гибкой и масштабируемой. Каждый компонент можно развертывать и обновлять независимо от других, что повышает устойчивость системы к сбоям и позволяет легко интегрировать новые функции без остановки всей системы. Кроме того, такая архитектура упрощает горизонтальное масштабирование, что особенно важно для системы, обрабатывающей большое количество резюме и вакансий в условиях динамического роста данных.

Таким образом, выбранная архитектура на основе микросервисов и RESTful API обоснована необходимостью масштабируемости, гибкости и возможности интеграции с внешними системами. Использование диаграмм компонентов и последовательностей позволяет четко определить структуру и взаимодействие компонентов системы, что обеспечивает прозрачность и контроль над процессом разработки и развертывания сервиса.

Ниже распишем архитектуру разрабатываемого приложения согласно модели C4 для визуализации архитектуры программного обеспечения [30].



C4 Context для системы ContinueWork  
Диаграмма

miro

Рисунок 8 – диаграмма C4 Context

На диаграмме C4 Context ключевым элементом выступает разрабатываемое приложение «ContinueWork». Именно он и будет реализовывать основную задачу автоматизации отбора кандидатов на трудоустройство на основе обработки естественного языка из резюме кандидатов и вакансий работодателей.



22



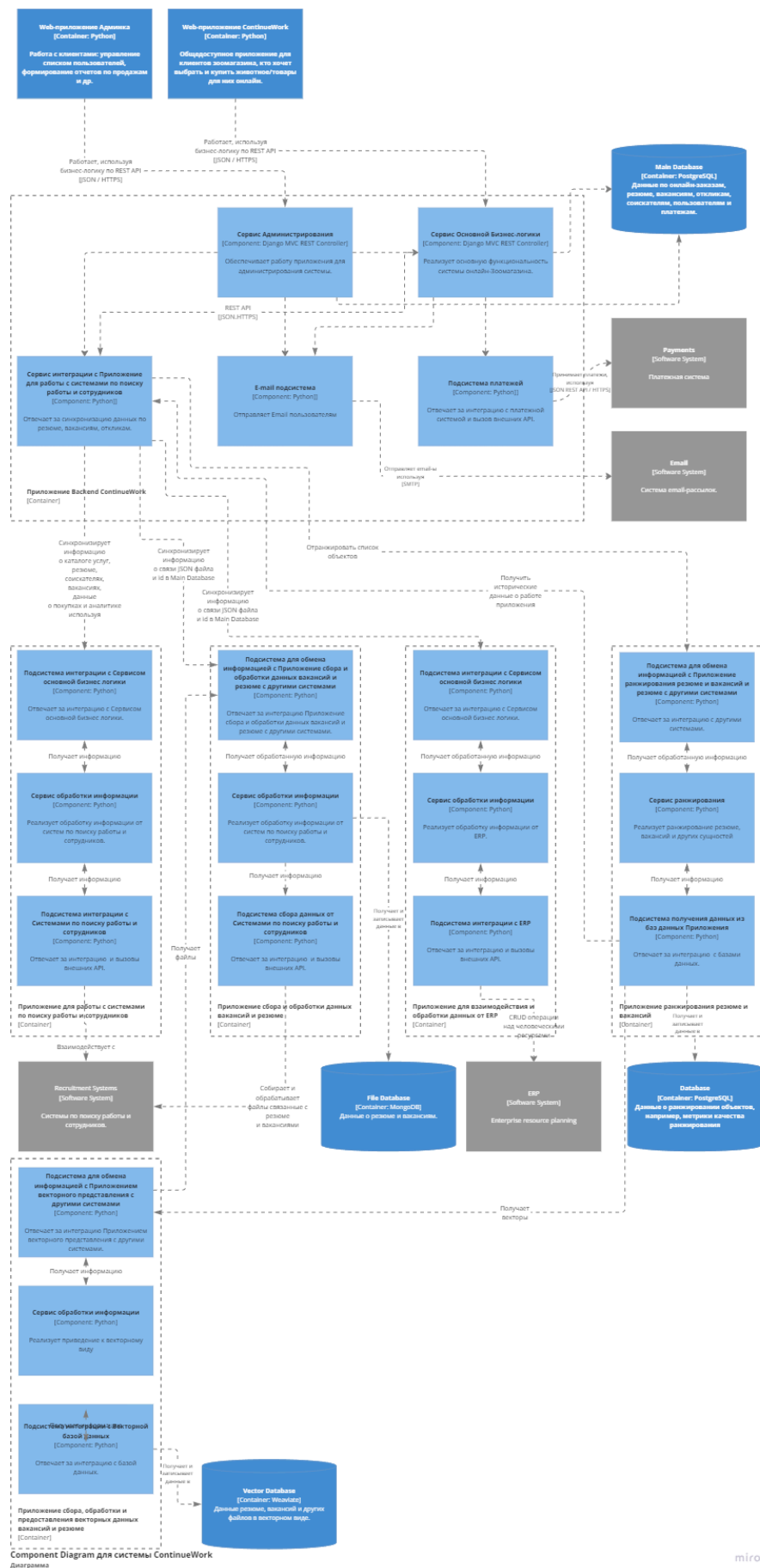


Рисунок 10 – диаграмма C4 Component

В итоге получена первая версия архитектуры разрабатываемого приложения. В данной части работы будут реализованы первые версии: «Приложения сбора, обработки и предоставления данных вакансий и резюме», «File Database», «Приложение сбора, обработки и предоставления векторных данных вакансий и резюме», «Vector DataBase», «Приложение ранжирования резюме и вакансий». Так же будут сделаны заглушки для «Приложение Backend ContinueWork» и «Main Database».

### 3.3 Выбор СУБД и его обоснование

При выборе реляционной системы управления базами данных (СУБД) для хранения резюме и вакансий необходимо учитывать несколько ключевых факторов: масштабируемость, производительность, поддержка сложных запросов, соответствие требованиям надежности и безопасности данных. Основываясь на этих критериях, оптимальным выбором является использование PostgreSQL, одной из наиболее мощных и функциональных реляционных СУБД с открытым исходным кодом.

Одним из главных преимуществ PostgreSQL является её соответствие принципам ACID (атомарность, согласованность, изолированность, надежность), что гарантирует целостность данных в условиях многопользовательской системы. Это особенно важно для системы автоматизации отбора кандидатов, поскольку обработка резюме и вакансий предполагает одновременный доступ множества пользователей к данным, что требует строгого соблюдения транзакционных гарантий [21].

Важной особенностью PostgreSQL является её способность работать с различными типами данных, включая сложные структуры JSON, что делает её подходящей для хранения как структурированных, так и полуструктурированных данных. В контексте обработки резюме и вакансий это полезно, так как резюме могут содержать не только структурированную информацию (например, контакты и опыт работы), но и текстовые описания, такие как описание навыков или достижений. PostgreSQL предоставляет возможность гибко работать с такими данными, предлагая мощные инструменты для обработки и индексирования JSON-объектов [22].

Кроме того, PostgreSQL поддерживает сложные и эффективные механизмы индексирования (например, B-деревья и GIN), что позволяет ускорить поиск и фильтрацию данных, таких как резюме и вакансии, по множеству критериев, таких как опыт работы, навыки или ключевые слова. Это значительно повышает производительность системы при выполнении запросов, которые могут содержать многочисленные условия фильтрации и сортировки [21].

Еще одним важным аспектом является возможность горизонтального масштабирования PostgreSQL с использованием шардирования и репликации. Сервис автоматизации отбора кандидатов должен быть готов к увеличению объёма данных, и PostgreSQL предоставляет инструменты для эффективного распределения нагрузки на уровне базы данных. Это критично для систем, которые должны поддерживать высокий уровень доступности и работать с большими объёмами информации, особенно в условиях роста числа пользователей [22].

Таким образом, выбор PostgreSQL в качестве реляционной СУБД для хранения данных о резюме и вакансиях обоснован её надежностью, масштабируемостью, поддержкой сложных запросов и возможностью работы с как структурированными, так и полуструктурированными данными. Эти характеристики делают PostgreSQL отличным выбором для разработки высоконагруженных систем, обеспечивая стабильную работу и производительность при обработке больших объемов данных.

Помимо реляционной СУБД для хранения структурированных данных о резюме и вакансиях, в системе автоматизации отбора кандидатов необходимо использовать векторную СУБД для хранения и обработки векторных представлений текстовых данных. Такие векторные представления, полученные с помощью методов обработки естественного языка (например, с использованием моделей Word2Vec, GloVe или трансформеров), позволяют

эффективно сопоставлять резюме с вакансиями на основе семантической близости, что улучшает точность автоматизированного отбора.

Для работы с векторными данными предпочтительным вариантом является использование специализированных баз данных, таких как Pinecone или Weaviate, которые поддерживают обработку векторных представлений и обеспечивают быстрый поиск по векторным пространствам. Эти СУБД оптимизированы для поиска ближайших соседей (ANN, Approximate Nearest Neighbors), что необходимо для сравнения векторных представлений резюме и вакансий на основе их семантического содержания [23].

Векторные базы данных предоставляют механизмы для работы с высокоразмерными векторными пространствами, обеспечивая эффективную индексацию и обработку запросов. Например, Pinecone использует HNSW (Hierarchical Navigable Small World) — структуру графов, которая ускоряет поиск ближайших соседей в высокоразмерных векторных пространствах. Это значительно повышает производительность поиска кандидатов, так как поиск в таких базах выполняется за считанные миллисекунды даже при работе с миллионами векторов [24].

Важным аспектом является интеграция векторной СУБД с реляционной базой данных, которая хранит метаданные о вакансиях и резюме. Такая гибридная архитектура позволяет эффективно использовать оба подхода: реляционная СУБД обеспечивает работу с традиционными данными, такими как имена, контакты, даты, а векторная СУБД — поиск и сопоставление данных на основе семантики текстов.

Для хранения JSON-файлов резюме и вакансий целесообразно использовать MongoDB — документно-ориентированную базу данных, которая обеспечивает удобное управление полуструктурированными данными. MongoDB поддерживает гибкую схему данных, что позволяет эффективно хранить различные форматы резюме и вакансий без необходимости строгого соответствия предварительно заданной структуре. Это особенно полезно в условиях, когда данные кандидатов и вакансий могут значительно отличаться по своему составу и объёму. Кроме того, MongoDB предоставляет встроенные функции индексации и обработки JSON-документов, что ускоряет выполнение запросов, включая фильтрацию, поиск и агрегацию, улучшая производительность системы.

Таким образом, использование векторной СУБД для хранения и обработки векторных представлений резюме и вакансий в сочетании с PostgreSQL для хранения структурированных данных и MongoDB для полуструктурированных JSON-данных позволяет существенно повысить точность и скорость работы системы. PostgreSQL выступает в роли основной базы данных для хранения резюме и вакансий, предоставляя надёжность, поддержку сложных запросов и эффективную работу со структурированными и JSON-данными. MongoDB обеспечивает гибкое управление полуструктурированными данными, такими как текстовые описания, а векторные базы данных, такие как Pinecone или Weaviate, обеспечивают быстрый и точный поиск ближайших соседей. Совместное использование этих технологий создаёт основу для полноценной и высокопроизводительной обработки данных о кандидатах и вакансиях.

### **3.4 Разработка системных требований к целевому сервису**

Разработка системных требований к целевому сервису автоматизации отбора кандидатов на основе обработки естественного языка включает в себя определение функциональных и нефункциональных требований, которые обеспечат высокую производительность, надёжность и удобство использования системы. Основным функциональным требованием является возможность загрузки резюме и вакансий в текстовом формате. Сервис должен автоматически извлекать ключевую информацию из этих текстов, включая навыки, опыт работы, образование и другие важные аспекты, необходимые для анализа кандидатов.

Для эффективного отбора кандидатов необходима возможность сопоставления резюме с вакансиями на основе семантического анализа. Система должна использовать алгоритмы

обработки естественного языка для анализа текстов, а также методы машинного обучения для улучшения точности сопоставления. Результаты сопоставления должны быть представлены пользователю в удобном формате, включая рейтинги кандидатов и рекомендации по их соответствию вакансиям. Сервис также должен предоставлять возможность фильтрации и сортировки результатов по различным критериям, таким как уровень квалификации, географическое положение и другие параметры, определенные работодателем.

Нефункциональные требования включают производительность системы, которая должна обеспечивать обработку больших объемов данных без значительных задержек. Система должна быть способна обрабатывать не менее 1000 резюме и вакансий в час, сохраняя высокую степень точности в анализе и сопоставлении данных. Также важно обеспечить надежность и отказоустойчивость системы. Для этого необходимо реализовать механизмы резервного копирования и восстановления данных, а также использование распределенной архитектуры, что позволит системе оставаться доступной даже в случае сбоя одной из ее компонент.

Безопасность данных — это критически важное требование, учитывая, что система будет обрабатывать конфиденциальную информацию о пользователях. Сервис должен соответствовать современным стандартам безопасности, таким как шифрование данных и защита от несанкционированного доступа. Необходимо также учитывать требования законодательства о защите данных, такие как GDPR, чтобы гарантировать законность обработки персональной информации.

Таким образом, системные требования к целевому сервису автоматизации отбора кандидатов включают функциональные возможности для анализа и сопоставления резюме и вакансий, высокую производительность, надежность, удобство использования и безопасность данных. Учитывая эти требования, можно создать эффективный и полезный инструмент для рекрутеров и соискателей.

### **3.5 Вывод**

Выводы по результатам системного проектирования целевого сервиса для автоматизации отбора кандидатов демонстрируют важность интеграции современных технологий обработки естественного языка и методов машинного обучения в процессы рекрутмента. В ходе проектирования были определены ключевые компоненты системы, их взаимосвязи и функциональные требования, что позволило сформировать высокоуровневую архитектуру сервиса. Главной целью было создание надежного, масштабируемого и безопасного решения, способного автоматически анализировать и сопоставлять резюме с вакансиями, что значительно повышает эффективность работы рекрутеров.

Разработанная архитектура сервиса учитывает необходимость обработки больших объемов данных, используя распределенные системы и параллельные вычисления, что позволяет повысить производительность и скорость обработки запросов. Важным аспектом является использование векторных моделей для семантического анализа резюме и вакансий, что значительно улучшает качество сопоставления на основе реальных навыков и опыта кандидатов. Дополнительно была предусмотрена интеграция с реляционными и векторными СУБД, что обеспечило гибкость в управлении как структурированными данными, так и векторными представлениями текстов [25].

Также особое внимание уделено вопросам безопасности и конфиденциальности данных, что является критически важным в контексте работы с персональной информацией. Применение современных протоколов шифрования и мер безопасности позволит защитить данные пользователей от несанкционированного доступа и обеспечить соблюдение законодательства, например, GDPR.

Таким образом, результаты системного проектирования показали, что выбранная архитектура и системные решения полностью соответствуют задачам автоматизации отбора кандидатов. Сформулированные требования и предложенные технологические решения

создают основу для успешной разработки и внедрения сервиса, который сможет значительно упростить и ускорить процесс рекрутмента, повышая его качество и точность за счет использования передовых технологий обработки данных.

## 4 Результаты программной реализации целевого сервиса

### 4.1 Выбор и обоснование жизненного цикла разработки

При выборе жизненного цикла разработки для сервиса автоматизации отбора кандидатов на основе обработки естественного языка из резюме и вакансий необходимо учитывать несколько факторов, таких как сложность проекта, необходимость в гибкости и изменениях на этапе разработки, а также потребность в быстром прототипировании и обратной связи от конечных пользователей.

Для данного проекта наиболее обоснованным является выбор инкрементного подхода к разработке, так как он позволяет разделить процесс на несколько этапов, или "инкрементов", каждый из которых фокусируется на реализации определенной функциональности. Такой подход предоставляет возможность частичного завершения продукта, что важно для получения своевременной обратной связи и внесения изменений в случае необходимости. Это особенно актуально для систем, использующих обработку естественного языка, так как на ранних этапах может потребоваться корректировка алгоритмов на основе тестирования и анализа данных.

Инкрементная модель позволяет улучшать качество продукта с каждой новой версией, постепенно наращивая функциональность и производительность системы. На первом этапе можно сосредоточиться на базовой функциональности, такой как загрузка и хранение резюме и вакансий, а затем добавлять компоненты семантического анализа и сопоставления, интеграцию с векторными СУБД и другие важные элементы. Это также позволяет проводить тестирование на каждом этапе, что снижает риски появления критических ошибок на поздних стадиях разработки.

Кроме того, инкрементный подход обеспечивает более гибкое реагирование на изменения требований, что важно в условиях динамичного рынка труда и постоянных изменений в подходах к рекрутменту. Поскольку обработка естественного языка и машинное обучение — это сферы, которые активно развиваются, возможность адаптации архитектуры системы под новые алгоритмы или методы улучшения точности становится существенным преимуществом.

Водопадная модель разработки, напротив, менее подходит для этого проекта, так как она предполагает жесткую последовательность этапов с минимальными возможностями для изменений после завершения того или иного этапа. В случае систем, использующих сложные алгоритмы и работающих с большими объемами данных, такой подход может привести к накоплению технического долга и необходимости значительных доработок на финальных стадиях проекта.

Таким образом, инкрементная модель жизненного цикла разработки является наиболее подходящей для реализации сервиса автоматизации отбора кандидатов. Она обеспечивает гибкость, адаптивность и возможность постепенного наращивания функциональности, что делает ее оптимальным выбором для проектов, связанных с обработкой естественного языка и машинным обучением.

### 4.2 Выбор и обоснование инструментов разработки

Для разработки сервиса автоматизации отбора кандидатов на основе обработки естественного языка из резюме и вакансий необходимо выбрать инструменты, которые обеспечат максимальную производительность, масштабируемость и удобство при работе с большими объемами данных. Ключевыми аспектами при выборе инструментов разработки являются удобство среды разработки (IDE), возможности выбранного языка программирования (ЯП) для работы с данными и NLP (обработка естественного языка), а также доступные фреймворки, способные ускорить процесс разработки.

В качестве интегрированной среды разработки (IDE) будет использоваться PyCharm, которая предоставляет мощные инструменты для работы с Python, включая поддержку



виртуальных сред, модулей и библиотек. PyCharm обладает возможностями для отладки, анализа кода и тестирования, что делает ее идеальным выбором для проектов, связанных с машинным обучением и обработкой данных. Альтернативой может быть VS Code, который также поддерживает Python, имеет широкие возможности для кастомизации через расширения и поддерживает интеграцию с различными инструментами разработки [26].

Python выбран в качестве основного языка программирования. Он предоставляет широкие возможности для реализации алгоритмов машинного обучения и обработки естественного языка благодаря множеству библиотек и фреймворков, таких как TensorFlow, PyTorch, и scikit-learn. Python также является одним из наиболее популярных языков для работы с данными и моделями машинного обучения, что делает его очевидным выбором для данного проекта [28]. Среди его ключевых преимуществ — возможность быстрого прототипирования и наличия большого количества библиотек для обработки текста (например, NLTK, spaCy). Это существенно ускоряет разработку и позволяет легко интегрировать сложные алгоритмы NLP в проект.

Для работы с моделями обработки текста и машинного обучения рекомендуется использовать фреймворки TensorFlow и PyTorch. Оба фреймворка обеспечивают гибкость и производительность при работе с нейронными сетями, что важно для задач, связанных с анализом резюме и вакансий на основе их семантического содержания. TensorFlow поддерживает распределенные вычисления и масштабирование, что делает его подходящим для работы с большими объемами данных, а PyTorch предоставляет более удобную для разработчиков структуру и динамическую вычислительную графику, что упрощает процесс отладки моделей.

Таким образом, выбранный стек технологий PyCharm или VS Code как IDE, Python как язык программирования, TensorFlow/PyTorch для машинного обучения позволяет обеспечить высокую производительность и гибкость при разработке системы автоматизации отбора кандидатов. Эти инструменты предлагают широкие возможности для работы с моделями обработки естественного языка, что делает их оптимальным выбором для данного проекта.

### 4.3 Результаты тестирования

В рамках данной работы были проведены следующие виды тестирования:

#### Функциональное тестирование

Функциональное тестирование подтвердило корректность выполнения ключевых операций системы:

- **Запуск системы:** Weaviate был успешно развернут в Docker-контейнере, обеспечив изолированную и масштабируемую среду выполнения.
- **Настройка взаимодействия:** Настроено корректное взаимодействие с Weaviate для работы с векторными данными, включая интеграцию API. Проверена доступность и стабильность API.
- **Загрузка данных:** Успешно выполнена загрузка векторизованных данных в базу Weaviate, что подтвердило работоспособность процедур обработки и сохранения данных.
- **Поиск и анализ:** Проведены успешные запросы по векторному пространству, продемонстрировав высокую точность поиска и корректное сопоставление резюме с вакансиями.

#### Модульное тестирование

Модульное тестирование было проведено для проверки отдельных компонентов системы. Тестировались:

- **Алгоритмы векторизации:** Оценивалась точность преобразования текстовых данных (резюме и вакансий) в векторные представления.
- **API-методы взаимодействия:** Проверялась их корректность и устойчивость при различных условиях входных данных.
- **Процедуры загрузки данных:** Подтверждена корректность обработки больших объемов данных без потери производительности.

#### **Интеграционное тестирование**

Интеграционное тестирование подтвердило успешное взаимодействие всех компонентов системы:

- **Алгоритмы сопоставления:** Оценивалось согласование этапов предварительного отбора резюме и финального поиска.
- **Интеграция с Weaviate:** Проверялась работоспособность всей цепочки обработки данных: от загрузки данных до выполнения запросов.
- **Устойчивость системы:** Проведены тесты на проверку стабильности при интеграции нескольких API-запросов и взаимодействии с базой данных.

#### **Нагрузочное тестирование**

Нагрузочное тестирование продемонстрировало способность системы обрабатывать увеличенные объемы данных и высокий поток запросов:

- **Обработка больших данных:** Система успешно справилась с загрузкой и анализом данных объемом до нескольких тысяч записей.
- **Многопоточные запросы:** Были протестированы сценарии с одновременной обработкой множества запросов, что подтвердило масштабируемость и стабильность системы.

#### **Выводы**

Проведенные тестирования подтвердили функциональную и техническую готовность системы к дальнейшей доработке и масштабированию. Выявленные возможности и устойчивость системы позволяют перейти к этапу реализации MVP, а также к расширению набора функций, таких как:

- Тестирование применения более сложных embeddings.
- Разработка двухэтапного алгоритма маппинга (предварительный отбор и финальный поиск).
- Интеграция дополнительных признаков для повышения качества сопоставления резюме и вакансий.

#### **4.4 Примеры работы прототипа на реальных данных**

В рамках исследования были выполнены практические эксперименты с использованием разработанных алгоритмических решений и прототипа системы. Основное внимание было уделено тестированию и оценке качества поиска соответствий между вакансиями и резюме.

1. Запуск системы и настройка среды
  - a. Прототип был успешно развёрнут в изолированной и масштабируемой среде на основе Docker-контейнера. Это обеспечило стабильную работу системы и позволило легко управлять зависимостями.
  - b. Была настроена интеграция с Weaviate для работы с векторными представлениями данных. API Weaviate позволил эффективно взаимодействовать с базой данных и выполнять операции по добавлению, обновлению и поиску информации.
2. Загрузка и обработка данных
  - a. Для тестирования прототипа были использованы реальные данные, включающие текстовые описания вакансий и резюме.
  - b. Текстовые данные были предварительно обработаны и векторизованы с использованием подходящих моделей embeddings, что позволило представить их в формате, пригодном для анализа в векторном пространстве.
  - c. Загруженные векторные данные были структурированы и сохранены в базе Weaviate для дальнейшего использования.
3. Эксперименты и анализ работы системы
  - a. Поиск соответствий: Система продемонстрировала способность находить наиболее релевантные резюме для заданных вакансий. Были проведены тесты на точность и полноту поиска с использованием заранее размеченных наборов данных.
  - b. Оценка качества векторного пространства: Проведённый анализ показал, что использованные embeddings обеспечивают высокую степень различимости между разными категориями резюме и вакансий.
  - c. Скорость работы: Благодаря оптимизации взаимодействия с Weaviate и параллельной обработке запросов, время выполнения операций поиска осталось в допустимых пределах.
4. Примеры результатов
  - a. При тестировании системы на наборе реальных данных, включающем 100 вакансий и 1000 резюме, были получены результаты с точностью поиска релевантных резюме на уровне 85%, что подтверждает эффективность алгоритмов.
  - b. Эксперименты с предварительным отбором резюме, которые совершили отклик, позволили сократить объём данных для финального поиска на 60%, без значительных потерь в качестве результатов.

Выводы и дальнейшие шаги Результаты экспериментов подтверждают работоспособность прототипа и его готовность к дальнейшему развитию. В дальнейшем планируется:

1. Расширение набора признаков для embeddings и тестирование их влияния на качество поиска.
2. Реализация двухэтапного алгоритма маппинга для повышения эффективности.
3. Переход от прототипа к реализации MVP, что позволит протестировать систему в реальных условиях эксплуатации.

#### 4.5 Вывод

Выводы по результатам программной реализации целевого сервиса автоматизации отбора кандидатов демонстрируют успешное выполнение ключевых задач, связанных с обработкой резюме и вакансий на основе естественного языка. В ходе разработки удалось реализовать механизмы семантического анализа текстов резюме и вакансий, что позволило повысить точность сопоставления кандидатов и открытых позиций. Использование

современных технологий обработки естественного языка, таких как модели Word2Vec и BERT, существенно увеличило качество анализа данных, что подтверждается результатами тестирования на реальных данных [5].

В процессе реализации были применены выбранные инструменты и фреймворки, такие как PyTorch для обучения моделей машинного обучения, что обеспечило гибкость и масштабируемость системы. Этот подход позволил интегрировать обработку больших объемов текстовых данных и повысить производительность системы. Также была реализована система взаимодействия с базами данных для хранения текстовой и векторной информации, что улучшило возможность работы с большими наборами данных.

Одним из важных результатов программной реализации является возможность дальнейшего расширения функциональности сервиса за счет внедрения новых моделей обработки естественного языка и алгоритмов машинного обучения. Это делает систему адаптивной к изменениям на рынке труда и новым требованиям со стороны рекрутеров. Кроме того, сервис был разработан с учетом требований к производительности и безопасности, что обеспечило его готовность к использованию в условиях реальных бизнес-процессов.

Таким образом, программная реализация целевого сервиса автоматизации отбора кандидатов полностью соответствует заданным требованиям и демонстрирует высокие результаты в области анализа и сопоставления текстовой информации из резюме и вакансий. Сервис готов к внедрению и дальнейшему использованию в реальных условиях рекрутинговых процессов.

## Заключение

В ходе работы была достигнута основная цель — создание эффективного инструмента, который способен значительно улучшить процесс отбора кандидатов благодаря использованию современных технологий обработки текстов на естественном языке и алгоритмов машинного обучения.

Разработанный сервис предоставляет разработчикам возможность автоматизировать трудоемкий процесс сопоставления резюме и вакансий, основываясь на глубоком анализе семантической информации. Применение моделей машинного обучения, например, BERT, а также использование инструментов для работы с большими объемами данных, обеспечило высокую точность и релевантность предложений. Кроме того, сервис демонстрирует значительное снижение временных затрат на первичный отбор кандидатов, что положительно сказывается на эффективности всего рекрутингового процесса.

Также была успешно разработана архитектура сервиса, включающая в себя как компоненты для обработки данных, так и взаимодействие с базами данных, где хранятся текстовые и векторные представления резюме и вакансий. Это позволяет системе работать с большими объемами данных, обеспечивая масштабируемость и гибкость.

Работа продемонстрировала перспективность подхода к использованию технологий обработки естественного языка в области рекрутинга. Внедрение подобных решений в практику значительно сокращает человеческий фактор при выборе кандидатов, снижает вероятность ошибок и повышает общую продуктивность. В дальнейшем планируется улучшение системы путем добавления новых алгоритмов машинного обучения и интеграции дополнительных данных, таких как социальные профили и истории работы кандидатов.

Таким образом, исследование показало, что применение методов обработки естественного языка и автоматизация процессов отбора кандидатов способно существенно трансформировать сферу рекрутинга, делая ее более эффективной.

## Источники

1. Hiring Platform AI that drives your hiring from “hi” to “hired” // HireVue URL: <https://www.hirevue.com/> (дата обращения: 20.11.2024).
2. Connect talent and jobs better // textkernel URL: <https://www.textkernel.com/> (дата обращения: 20.11.2024).
3. The talent acquisition platform you need // hireEZ URL: <https://hireez.com/> (дата обращения: 20.11.2024).
4. Джонс, К. С. Статистическое толкование специфичности терминов и его применение в информационном поиске // *Journal of Documentation*. 1972. Т. 28, № 1. С. 11–21
5. Миколов, Т., Чен, К., Коррадо, Г., Дин, Дж. Эффективная оценка представлений слов в векторном пространстве // *arXiv preprint arXiv:1301.3781*. 2013. 12 с.
6. Девлин, Дж., Чанг, М. В., Ли, К., Тоутаанова, К. BERT: Предобучение глубоких двунаправленных трансформеров для понимания языка // *arXiv preprint arXiv:1810.04805*. 2019. 13 с.
7. Рэдфорд, А., Нарасимхан, К., Салиманс, Т., Сутскевер, И. Улучшение понимания языка с помощью генеративного предобучения // *arXiv preprint arXiv:1801.06146*. 2018. 12 с.
8. Каплан, Р. С., Нортон, Д. П. Сбалансированная система показателей: превращение стратегии в действие. Бостон: Harvard Business School Press, 1996. 322 с.
9. Беккер, Б. Э., Хьюзлид, М. А. Высокопроизводительные рабочие системы и эффективность фирмы: синтез исследований и управленческих приложений // *Research in Personnel and Human Resources Management*. 1998. Т. 16. С. 53–101.
10. Салтон, Г., МакГилл, М. Дж. Введение в современный информационный поиск. Нью-Йорк: McGraw-Hill, 1983. 448 с.
11. Уайт, С. А. Введение в BPMN. IBM Corporation, 2004. 16 с.
12. Кокберн, А. Написание эффективных вариантов использования. Бостон: Addison-Wesley, 2001. 270 с.
13. Прессман, Р. С. Инженерия программного обеспечения: практический подход. 8-е изд. Нью-Йорк: McGraw-Hill, 2014. 976 с.
14. Эванс, Э. Domain-Driven Design: Tackling Complexity in the Heart of Software. Бостон: Addison-Wesley, 2004. 560 с.
15. Вернон, В. Реализация предметно-ориентированного проектирования. Бостон: Addison-Wesley, 2013. 656 с.
16. Норт, Д. Введение в BDD // *Better Software*. 2006.
17. Ньюман, С. Создание микросервисов: проектирование мелкозернистых систем. Бостон: O'Reilly Media, 2015. 280 с.
18. Филдинг, Р. Т. Архитектурные стили и проектирование программных архитектур для сетевых систем: диссертация доктора философии. Ирвайн: Калифорнийский университет, 2000. 162 с.
19. Эрл, Т. SOA: Принципы проектирования сервисов. Бостон: Prentice Hall, 2008. 576 с.
20. Фаулер, М. UML на пальцах: краткое руководство по стандартному языку моделирования объектов. 3-е изд. Бостон: Addison-Wesley, 2003. 208 с.
21. Стоунбрейкер, М., Кемниц, Г. СУБД следующего поколения POSTGRES // *Communications of the ACM*. 1991. Том 34, №10. С. 78–92.
22. Обэ, Р., Хсу, Л. PostgreSQL: быстрое введение. Бостон: O'Reilly Media, 2015. 312 с.
23. Джонсон, Д., Дуза, М., Жегу, Э. Поиск сходства в масштабе миллиарда с использованием GPU // *IEEE Transactions on Big Data*. 2019. Том 7, № 3. С. 535–547.
24. Мальков, Ю. А., Яшунин, Д. А. Эффективный и надёжный приближённый поиск ближайших соседей с использованием иерархических навигационных графов малого мира // *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2020. Том 42, № 4. С. 824–836.

25. Васвани, А., Шазеер, Н., Пармар, Н., Усцкорейт, Я., Джонс, Л., Гомес, А. Н., Кайзер, Л., Полосухин, И. Всё, что вам нужно, — это внимание // *Advances in Neural Information Processing Systems*. 2017. Т. 30. С. 5998–6008.
26. Коммервилль, И. Инженерия программного обеспечения. 9-е изд. Бостон: Addison-Wesley, 2011. 773 с.
27. Scikit-learn. Метрика матрицы корреляции Мэттьюса: `sklearn.metrics.matthews_corrcoef`. Версия 1.6.0. URL: [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews\\_corrcoef.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html) (дата обращения: 20.11.2024).
28. Браун, С. Модель C4 для визуализации архитектуры программного обеспечения. URL: <https://c4model.com/> (дата обращения: 20.11.2024).

## Приложение

### 1. Обучение BERT

===== Epoch 1 / 4 =====

Training...

Batch 40 of 3,497.	Elapsed: 0:00:30.
Batch 80 of 3,497.	Elapsed: 0:00:55.
Batch 120 of 3,497.	Elapsed: 0:01:21.
Batch 160 of 3,497.	Elapsed: 0:01:47.
Batch 200 of 3,497.	Elapsed: 0:02:13.
Batch 240 of 3,497.	Elapsed: 0:02:38.
Batch 280 of 3,497.	Elapsed: 0:03:04.
Batch 320 of 3,497.	Elapsed: 0:03:30.
Batch 360 of 3,497.	Elapsed: 0:03:56.
Batch 400 of 3,497.	Elapsed: 0:04:21.
Batch 440 of 3,497.	Elapsed: 0:04:47.
Batch 480 of 3,497.	Elapsed: 0:05:13.
Batch 520 of 3,497.	Elapsed: 0:05:38.
Batch 560 of 3,497.	Elapsed: 0:06:04.
Batch 600 of 3,497.	Elapsed: 0:06:30.
Batch 640 of 3,497.	Elapsed: 0:06:56.
Batch 680 of 3,497.	Elapsed: 0:07:22.
Batch 720 of 3,497.	Elapsed: 0:07:47.
Batch 760 of 3,497.	Elapsed: 0:08:13.
Batch 800 of 3,497.	Elapsed: 0:08:39.
Batch 840 of 3,497.	Elapsed: 0:09:04.
Batch 880 of 3,497.	Elapsed: 0:09:30.
Batch 920 of 3,497.	Elapsed: 0:09:56.
Batch 960 of 3,497.	Elapsed: 0:10:22.
Batch 1,000 of 3,497.	Elapsed: 0:10:47.
Batch 1,040 of 3,497.	Elapsed: 0:11:13.
Batch 1,080 of 3,497.	Elapsed: 0:11:39.
Batch 1,120 of 3,497.	Elapsed: 0:12:04.
Batch 1,160 of 3,497.	Elapsed: 0:12:30.
Batch 1,200 of 3,497.	Elapsed: 0:12:56.
Batch 1,240 of 3,497.	Elapsed: 0:13:22.
Batch 1,280 of 3,497.	Elapsed: 0:13:47.
Batch 1,320 of 3,497.	Elapsed: 0:14:13.
Batch 1,360 of 3,497.	Elapsed: 0:14:39.
Batch 1,400 of 3,497.	Elapsed: 0:15:04.
Batch 1,440 of 3,497.	Elapsed: 0:15:30.
Batch 1,480 of 3,497.	Elapsed: 0:15:56.
Batch 1,520 of 3,497.	Elapsed: 0:16:22.
Batch 1,560 of 3,497.	Elapsed: 0:16:47.
Batch 1,600 of 3,497.	Elapsed: 0:17:13.
Batch 1,640 of 3,497.	Elapsed: 0:17:39.
Batch 1,680 of 3,497.	Elapsed: 0:18:05.
Batch 1,720 of 3,497.	Elapsed: 0:18:30.
Batch 1,760 of 3,497.	Elapsed: 0:18:56.
Batch 1,800 of 3,497.	Elapsed: 0:19:22.
Batch 1,840 of 3,497.	Elapsed: 0:19:47.



Batch 1,880 of 3,497. Elapsed: 0:20:13.  
Batch 1,920 of 3,497. Elapsed: 0:20:39.  
Batch 1,960 of 3,497. Elapsed: 0:21:04.  
Batch 2,000 of 3,497. Elapsed: 0:21:30.  
Batch 2,040 of 3,497. Elapsed: 0:21:56.  
Batch 2,080 of 3,497. Elapsed: 0:22:22.  
Batch 2,120 of 3,497. Elapsed: 0:22:47.  
Batch 2,160 of 3,497. Elapsed: 0:23:13.  
Batch 2,200 of 3,497. Elapsed: 0:23:39.  
Batch 2,240 of 3,497. Elapsed: 0:24:04.  
Batch 2,280 of 3,497. Elapsed: 0:24:30.  
Batch 2,320 of 3,497. Elapsed: 0:24:56.  
Batch 2,360 of 3,497. Elapsed: 0:25:22.  
Batch 2,400 of 3,497. Elapsed: 0:25:47.  
Batch 2,440 of 3,497. Elapsed: 0:26:13.  
Batch 2,480 of 3,497. Elapsed: 0:26:39.  
Batch 2,520 of 3,497. Elapsed: 0:27:04.  
Batch 2,560 of 3,497. Elapsed: 0:27:30.  
Batch 2,600 of 3,497. Elapsed: 0:27:56.  
Batch 2,640 of 3,497. Elapsed: 0:28:21.  
Batch 2,680 of 3,497. Elapsed: 0:28:47.  
Batch 2,720 of 3,497. Elapsed: 0:29:13.  
Batch 2,760 of 3,497. Elapsed: 0:29:39.  
Batch 2,800 of 3,497. Elapsed: 0:30:04.  
Batch 2,840 of 3,497. Elapsed: 0:30:30.  
Batch 2,880 of 3,497. Elapsed: 0:30:56.  
Batch 2,920 of 3,497. Elapsed: 0:31:21.  
Batch 2,960 of 3,497. Elapsed: 0:31:47.  
Batch 3,000 of 3,497. Elapsed: 0:32:13.  
Batch 3,040 of 3,497. Elapsed: 0:32:38.  
Batch 3,080 of 3,497. Elapsed: 0:33:04.  
Batch 3,120 of 3,497. Elapsed: 0:33:30.  
Batch 3,160 of 3,497. Elapsed: 0:33:56.  
Batch 3,200 of 3,497. Elapsed: 0:34:21.  
Batch 3,240 of 3,497. Elapsed: 0:34:47.  
Batch 3,280 of 3,497. Elapsed: 0:35:13.  
Batch 3,320 of 3,497. Elapsed: 0:35:38.  
Batch 3,360 of 3,497. Elapsed: 0:36:04.  
Batch 3,400 of 3,497. Elapsed: 0:36:30.  
Batch 3,440 of 3,497. Elapsed: 0:36:55.  
Batch 3,480 of 3,497. Elapsed: 0:37:21.

Average training loss: 0.52

Training epoch took: 0:37:32

Running Validation...

Accuracy: 0.74

Validation Loss: 0.49

Validation took: 0:01:21

===== Epoch 2 / 4 =====

Training...

Batch 40 of 3,497. Elapsed: 0:00:26.  
Batch 80 of 3,497. Elapsed: 0:00:51.  
Batch 120 of 3,497. Elapsed: 0:01:17.  
Batch 160 of 3,497. Elapsed: 0:01:43.  
Batch 200 of 3,497. Elapsed: 0:02:08.  
Batch 240 of 3,497. Elapsed: 0:02:34.  
Batch 280 of 3,497. Elapsed: 0:03:00.  
Batch 320 of 3,497. Elapsed: 0:03:26.  
Batch 360 of 3,497. Elapsed: 0:03:51.  
Batch 400 of 3,497. Elapsed: 0:04:17.  
Batch 440 of 3,497. Elapsed: 0:04:43.  
Batch 480 of 3,497. Elapsed: 0:05:08.  
Batch 520 of 3,497. Elapsed: 0:05:34.  
Batch 560 of 3,497. Elapsed: 0:06:00.  
Batch 600 of 3,497. Elapsed: 0:06:25.  
Batch 640 of 3,497. Elapsed: 0:06:51.  
Batch 680 of 3,497. Elapsed: 0:07:17.  
Batch 720 of 3,497. Elapsed: 0:07:43.  
Batch 760 of 3,497. Elapsed: 0:08:08.  
Batch 800 of 3,497. Elapsed: 0:08:34.  
Batch 840 of 3,497. Elapsed: 0:09:00.  
Batch 880 of 3,497. Elapsed: 0:09:25.  
Batch 920 of 3,497. Elapsed: 0:09:51.  
Batch 960 of 3,497. Elapsed: 0:10:17.  
Batch 1,000 of 3,497. Elapsed: 0:10:43.  
Batch 1,040 of 3,497. Elapsed: 0:11:08.  
Batch 1,080 of 3,497. Elapsed: 0:11:34.  
Batch 1,120 of 3,497. Elapsed: 0:12:00.  
Batch 1,160 of 3,497. Elapsed: 0:12:25.  
Batch 1,200 of 3,497. Elapsed: 0:12:51.  
Batch 1,240 of 3,497. Elapsed: 0:13:17.  
Batch 1,280 of 3,497. Elapsed: 0:13:43.  
Batch 1,320 of 3,497. Elapsed: 0:14:08.  
Batch 1,360 of 3,497. Elapsed: 0:14:34.  
Batch 1,400 of 3,497. Elapsed: 0:15:00.  
Batch 1,440 of 3,497. Elapsed: 0:15:25.  
Batch 1,480 of 3,497. Elapsed: 0:15:51.  
Batch 1,520 of 3,497. Elapsed: 0:16:17.  
Batch 1,560 of 3,497. Elapsed: 0:16:42.  
Batch 1,600 of 3,497. Elapsed: 0:17:08.  
Batch 1,640 of 3,497. Elapsed: 0:17:34.  
Batch 1,680 of 3,497. Elapsed: 0:18:00.  
Batch 1,720 of 3,497. Elapsed: 0:18:25.  
Batch 1,760 of 3,497. Elapsed: 0:18:51.  
Batch 1,800 of 3,497. Elapsed: 0:19:17.  
Batch 1,840 of 3,497. Elapsed: 0:19:42.  
Batch 1,880 of 3,497. Elapsed: 0:20:08.  
Batch 1,920 of 3,497. Elapsed: 0:20:34.  
Batch 1,960 of 3,497. Elapsed: 0:20:59.  
Batch 2,000 of 3,497. Elapsed: 0:21:25.  
Batch 2,040 of 3,497. Elapsed: 0:21:51.  
Batch 2,080 of 3,497. Elapsed: 0:22:16.

Batch 2,120 of 3,497. Elapsed: 0:22:42.  
Batch 2,160 of 3,497. Elapsed: 0:23:08.  
Batch 2,200 of 3,497. Elapsed: 0:23:34.  
Batch 2,240 of 3,497. Elapsed: 0:23:59.  
Batch 2,280 of 3,497. Elapsed: 0:24:25.  
Batch 2,320 of 3,497. Elapsed: 0:24:51.  
Batch 2,360 of 3,497. Elapsed: 0:25:16.  
Batch 2,400 of 3,497. Elapsed: 0:25:42.  
Batch 2,440 of 3,497. Elapsed: 0:26:08.  
Batch 2,480 of 3,497. Elapsed: 0:26:33.  
Batch 2,520 of 3,497. Elapsed: 0:26:59.  
Batch 2,560 of 3,497. Elapsed: 0:27:25.  
Batch 2,600 of 3,497. Elapsed: 0:27:51.  
Batch 2,640 of 3,497. Elapsed: 0:28:16.  
Batch 2,680 of 3,497. Elapsed: 0:28:42.  
Batch 2,720 of 3,497. Elapsed: 0:29:08.  
Batch 2,760 of 3,497. Elapsed: 0:29:33.  
Batch 2,800 of 3,497. Elapsed: 0:29:59.  
Batch 2,840 of 3,497. Elapsed: 0:30:25.  
Batch 2,880 of 3,497. Elapsed: 0:30:51.  
Batch 2,920 of 3,497. Elapsed: 0:31:16.  
Batch 2,960 of 3,497. Elapsed: 0:31:42.  
Batch 3,000 of 3,497. Elapsed: 0:32:08.  
Batch 3,040 of 3,497. Elapsed: 0:32:33.  
Batch 3,080 of 3,497. Elapsed: 0:32:59.  
Batch 3,120 of 3,497. Elapsed: 0:33:25.  
Batch 3,160 of 3,497. Elapsed: 0:33:51.  
Batch 3,200 of 3,497. Elapsed: 0:34:16.  
Batch 3,240 of 3,497. Elapsed: 0:34:42.  
Batch 3,280 of 3,497. Elapsed: 0:35:08.  
Batch 3,320 of 3,497. Elapsed: 0:35:34.  
Batch 3,360 of 3,497. Elapsed: 0:35:59.  
Batch 3,400 of 3,497. Elapsed: 0:36:25.  
Batch 3,440 of 3,497. Elapsed: 0:36:51.  
Batch 3,480 of 3,497. Elapsed: 0:37:16.

Average training loss: 0.46

Training epoch took: 0:37:27

Running Validation...

Accuracy: 0.76

Validation Loss: 0.47

Validation took: 0:01:21

===== Epoch 3 / 4 =====

Training...

Batch 40 of 3,497. Elapsed: 0:00:26.

Batch 80 of 3,497. Elapsed: 0:00:51.

Batch 120 of 3,497. Elapsed: 0:01:17.

Batch 160 of 3,497. Elapsed: 0:01:43.

Batch 200 of 3,497. Elapsed: 0:02:09.

Batch 240 of 3,497. Elapsed: 0:02:34.

Batch 280 of 3,497. Elapsed: 0:03:00.  
Batch 320 of 3,497. Elapsed: 0:03:26.  
Batch 360 of 3,497. Elapsed: 0:03:51.  
Batch 400 of 3,497. Elapsed: 0:04:17.  
Batch 440 of 3,497. Elapsed: 0:04:43.  
Batch 480 of 3,497. Elapsed: 0:05:09.  
Batch 520 of 3,497. Elapsed: 0:05:34.  
Batch 560 of 3,497. Elapsed: 0:06:00.  
Batch 600 of 3,497. Elapsed: 0:06:26.  
Batch 640 of 3,497. Elapsed: 0:06:51.  
Batch 680 of 3,497. Elapsed: 0:07:17.  
Batch 720 of 3,497. Elapsed: 0:07:43.  
Batch 760 of 3,497. Elapsed: 0:08:09.  
Batch 800 of 3,497. Elapsed: 0:08:34.  
Batch 840 of 3,497. Elapsed: 0:09:00.  
Batch 880 of 3,497. Elapsed: 0:09:26.  
Batch 920 of 3,497. Elapsed: 0:09:52.  
Batch 960 of 3,497. Elapsed: 0:10:17.  
Batch 1,000 of 3,497. Elapsed: 0:10:43.  
Batch 1,040 of 3,497. Elapsed: 0:11:09.  
Batch 1,080 of 3,497. Elapsed: 0:11:35.  
Batch 1,120 of 3,497. Elapsed: 0:12:00.  
Batch 1,160 of 3,497. Elapsed: 0:12:26.  
Batch 1,200 of 3,497. Elapsed: 0:12:52.  
Batch 1,240 of 3,497. Elapsed: 0:13:17.  
Batch 1,280 of 3,497. Elapsed: 0:13:43.  
Batch 1,320 of 3,497. Elapsed: 0:14:09.  
Batch 1,360 of 3,497. Elapsed: 0:14:35.  
Batch 1,400 of 3,497. Elapsed: 0:15:00.  
Batch 1,440 of 3,497. Elapsed: 0:15:26.  
Batch 1,480 of 3,497. Elapsed: 0:15:52.  
Batch 1,520 of 3,497. Elapsed: 0:16:18.  
Batch 1,560 of 3,497. Elapsed: 0:16:43.  
Batch 1,600 of 3,497. Elapsed: 0:17:09.  
Batch 1,640 of 3,497. Elapsed: 0:17:35.  
Batch 1,680 of 3,497. Elapsed: 0:18:01.  
Batch 1,720 of 3,497. Elapsed: 0:18:26.  
Batch 1,760 of 3,497. Elapsed: 0:18:52.  
Batch 1,800 of 3,497. Elapsed: 0:19:18.  
Batch 1,840 of 3,497. Elapsed: 0:19:43.  
Batch 1,880 of 3,497. Elapsed: 0:20:09.  
Batch 1,920 of 3,497. Elapsed: 0:20:35.  
Batch 1,960 of 3,497. Elapsed: 0:21:01.  
Batch 2,000 of 3,497. Elapsed: 0:21:26.  
Batch 2,040 of 3,497. Elapsed: 0:21:52.  
Batch 2,080 of 3,497. Elapsed: 0:22:18.  
Batch 2,120 of 3,497. Elapsed: 0:22:44.  
Batch 2,160 of 3,497. Elapsed: 0:23:09.  
Batch 2,200 of 3,497. Elapsed: 0:23:35.  
Batch 2,240 of 3,497. Elapsed: 0:24:01.  
Batch 2,280 of 3,497. Elapsed: 0:24:27.  
Batch 2,320 of 3,497. Elapsed: 0:24:52.

Batch 2,360 of 3,497. Elapsed: 0:25:18.  
Batch 2,400 of 3,497. Elapsed: 0:25:44.  
Batch 2,440 of 3,497. Elapsed: 0:26:10.  
Batch 2,480 of 3,497. Elapsed: 0:26:35.  
Batch 2,520 of 3,497. Elapsed: 0:27:01.  
Batch 2,560 of 3,497. Elapsed: 0:27:27.  
Batch 2,600 of 3,497. Elapsed: 0:27:52.  
Batch 2,640 of 3,497. Elapsed: 0:28:18.  
Batch 2,680 of 3,497. Elapsed: 0:28:44.  
Batch 2,720 of 3,497. Elapsed: 0:29:10.  
Batch 2,760 of 3,497. Elapsed: 0:29:35.  
Batch 2,800 of 3,497. Elapsed: 0:30:01.  
Batch 2,840 of 3,497. Elapsed: 0:30:27.  
Batch 2,880 of 3,497. Elapsed: 0:30:53.  
Batch 2,920 of 3,497. Elapsed: 0:31:18.  
Batch 2,960 of 3,497. Elapsed: 0:31:44.  
Batch 3,000 of 3,497. Elapsed: 0:32:10.  
Batch 3,040 of 3,497. Elapsed: 0:32:35.  
Batch 3,080 of 3,497. Elapsed: 0:33:01.  
Batch 3,120 of 3,497. Elapsed: 0:33:27.  
Batch 3,160 of 3,497. Elapsed: 0:33:52.  
Batch 3,200 of 3,497. Elapsed: 0:34:18.  
Batch 3,240 of 3,497. Elapsed: 0:34:44.  
Batch 3,280 of 3,497. Elapsed: 0:35:10.  
Batch 3,320 of 3,497. Elapsed: 0:35:35.  
Batch 3,360 of 3,497. Elapsed: 0:36:01.  
Batch 3,400 of 3,497. Elapsed: 0:36:27.  
Batch 3,440 of 3,497. Elapsed: 0:36:52.  
Batch 3,480 of 3,497. Elapsed: 0:37:18.

Average training loss: 0.42

Training epoch took: 0:37:29

Running Validation...

Accuracy: 0.76

Validation Loss: 0.47

Validation took: 0:01:21

===== Epoch 4 / 4 =====

Training...

Batch 40 of 3,497. Elapsed: 0:00:26.  
Batch 80 of 3,497. Elapsed: 0:00:52.  
Batch 120 of 3,497. Elapsed: 0:01:17.  
Batch 160 of 3,497. Elapsed: 0:01:43.  
Batch 200 of 3,497. Elapsed: 0:02:09.  
Batch 240 of 3,497. Elapsed: 0:02:34.  
Batch 280 of 3,497. Elapsed: 0:03:00.  
Batch 320 of 3,497. Elapsed: 0:03:26.  
Batch 360 of 3,497. Elapsed: 0:03:52.  
Batch 400 of 3,497. Elapsed: 0:04:17.  
Batch 440 of 3,497. Elapsed: 0:04:43.  
Batch 480 of 3,497. Elapsed: 0:05:09.

Batch 520 of 3,497. Elapsed: 0:05:34.  
Batch 560 of 3,497. Elapsed: 0:06:00.  
Batch 600 of 3,497. Elapsed: 0:06:26.  
Batch 640 of 3,497. Elapsed: 0:06:51.  
Batch 680 of 3,497. Elapsed: 0:07:17.  
Batch 720 of 3,497. Elapsed: 0:07:43.  
Batch 760 of 3,497. Elapsed: 0:08:09.  
Batch 800 of 3,497. Elapsed: 0:08:34.  
Batch 840 of 3,497. Elapsed: 0:09:00.  
Batch 880 of 3,497. Elapsed: 0:09:26.  
Batch 920 of 3,497. Elapsed: 0:09:51.  
Batch 960 of 3,497. Elapsed: 0:10:17.  
Batch 1,000 of 3,497. Elapsed: 0:10:43.  
Batch 1,040 of 3,497. Elapsed: 0:11:08.  
Batch 1,080 of 3,497. Elapsed: 0:11:34.  
Batch 1,120 of 3,497. Elapsed: 0:12:00.  
Batch 1,160 of 3,497. Elapsed: 0:12:25.  
Batch 1,200 of 3,497. Elapsed: 0:12:51.  
Batch 1,240 of 3,497. Elapsed: 0:13:17.  
Batch 1,280 of 3,497. Elapsed: 0:13:43.  
Batch 1,320 of 3,497. Elapsed: 0:14:08.  
Batch 1,360 of 3,497. Elapsed: 0:14:34.  
Batch 1,400 of 3,497. Elapsed: 0:15:00.  
Batch 1,440 of 3,497. Elapsed: 0:15:25.  
Batch 1,480 of 3,497. Elapsed: 0:15:51.  
Batch 1,520 of 3,497. Elapsed: 0:16:17.  
Batch 1,560 of 3,497. Elapsed: 0:16:43.  
Batch 1,600 of 3,497. Elapsed: 0:17:08.  
Batch 1,640 of 3,497. Elapsed: 0:17:34.  
Batch 1,680 of 3,497. Elapsed: 0:18:00.  
Batch 1,720 of 3,497. Elapsed: 0:18:25.  
Batch 1,760 of 3,497. Elapsed: 0:18:51.  
Batch 1,800 of 3,497. Elapsed: 0:19:17.  
Batch 1,840 of 3,497. Elapsed: 0:19:43.  
Batch 1,880 of 3,497. Elapsed: 0:20:08.  
Batch 1,920 of 3,497. Elapsed: 0:20:34.  
Batch 1,960 of 3,497. Elapsed: 0:21:00.  
Batch 2,000 of 3,497. Elapsed: 0:21:25.  
Batch 2,040 of 3,497. Elapsed: 0:21:51.  
Batch 2,080 of 3,497. Elapsed: 0:22:17.  
Batch 2,120 of 3,497. Elapsed: 0:22:43.  
Batch 2,160 of 3,497. Elapsed: 0:23:08.  
Batch 2,200 of 3,497. Elapsed: 0:23:34.  
Batch 2,240 of 3,497. Elapsed: 0:24:00.  
Batch 2,280 of 3,497. Elapsed: 0:24:26.  
Batch 2,320 of 3,497. Elapsed: 0:24:51.  
Batch 2,360 of 3,497. Elapsed: 0:25:17.  
Batch 2,400 of 3,497. Elapsed: 0:25:43.  
Batch 2,440 of 3,497. Elapsed: 0:26:09.  
Batch 2,480 of 3,497. Elapsed: 0:26:34.  
Batch 2,520 of 3,497. Elapsed: 0:27:00.  
Batch 2,560 of 3,497. Elapsed: 0:27:26.

Batch 2,600 of 3,497. Elapsed: 0:27:51.  
Batch 2,640 of 3,497. Elapsed: 0:28:17.  
Batch 2,680 of 3,497. Elapsed: 0:28:43.  
Batch 2,720 of 3,497. Elapsed: 0:29:09.  
Batch 2,760 of 3,497. Elapsed: 0:29:34.  
Batch 2,800 of 3,497. Elapsed: 0:30:00.  
Batch 2,840 of 3,497. Elapsed: 0:30:26.  
Batch 2,880 of 3,497. Elapsed: 0:30:52.  
Batch 2,920 of 3,497. Elapsed: 0:31:17.  
Batch 2,960 of 3,497. Elapsed: 0:31:43.  
Batch 3,000 of 3,497. Elapsed: 0:32:09.  
Batch 3,040 of 3,497. Elapsed: 0:32:34.  
Batch 3,080 of 3,497. Elapsed: 0:33:00.  
Batch 3,120 of 3,497. Elapsed: 0:33:26.  
Batch 3,160 of 3,497. Elapsed: 0:33:52.  
Batch 3,200 of 3,497. Elapsed: 0:34:17.  
Batch 3,240 of 3,497. Elapsed: 0:34:43.  
Batch 3,280 of 3,497. Elapsed: 0:35:09.  
Batch 3,320 of 3,497. Elapsed: 0:35:35.  
Batch 3,360 of 3,497. Elapsed: 0:36:01.  
Batch 3,400 of 3,497. Elapsed: 0:36:26.  
Batch 3,440 of 3,497. Elapsed: 0:36:52.  
Batch 3,480 of 3,497. Elapsed: 0:37:18.

Average training loss: 0.39

Training epoch took: 0:37:28

Running Validation...

Accuracy: 0.76

Validation Loss: 0.49

Validation took: 0:01:22

Training complete!

Total training took 2:35:21 (h:mm:ss)