

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **Прикладной информатики**

Направление подготовки **45.03.04 Интеллектуальные системы в гуманитарной сфере**

Образовательная программа **Языковые модели и искусственный интеллект**

КУРСОВОЙ ПРОЕКТ

Тема: «Разработка сервиса локализации мобильного приложения»

Обучающийся: Калинин Марк Алексеевич, группа K3161

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ.....	3
1.1 Актуальность рассматриваемой темы.....	3
1.2 Цель проекта.....	4
1.3 Задачи проекта.....	4
2 РАБОТА НАД ПРОЕКТОМ.....	5
2.1 Суть проекта.....	5
2.2 Процессы работы над проектом.....	5
2.2.1 Определение целевых языков.....	5
2.2.2 Определение нужных сторонних сервисов.....	6
2.2.3 Обучение моделей.....	6
2.2.4 Размещение облачного сервера.....	6
2.2.5 Создание сервисного слоя мобильного приложения...	8
2.3 Мои задачи.....	10
2.4 Анализ моей работы.....	12
2.5 Взаимодействие с командой.....	12
2.6 Взаимодействие с руководителем.....	13
ЗАКЛЮЧЕНИЕ.....	14
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	16
ПРИЛОЖЕНИЕ. ТЕХНИЧЕСКОЕ ЗАДАНИЕ.....	17

1 ВВЕДЕНИЕ

1.1 Актуальность рассматриваемой темы

Компании и бизнес-проекты всегда стремятся максимизировать прибыль, поэтому нередко ставят перед собой цель создать продукт, который может выходить на международные рынки. В связи с этим у компаний нередко появляется проблема локализации: в разных странах разговаривают на разных языках, поэтому приложения приходится адаптировать и переводить. Это можно сделать несколькими способами: как правило, компании переводят навигационные компоненты на разные языки и в коде самого приложения хранят переводы, что неоптимально: это называется хардкод (от англ. *hard code* – «жёсткое кодирование», практика, заключающаяся в встраивании переменных и их значений в сам код программы – это считается неоптимальным, так как использует дополнительные ресурсы и увеличивает вес программы). Кроме того, это не решает проблему пользовательских строк: например, для маркетплейса такая стратегия не сработает, поскольку строки (названия товаров) в приложении вводят сами пользователи. Второй вариант – динамический перевод, плюс которого заключается в том, что для адаптации к изменениям не требуется перезагрузки приложения, но он, как правило, платный: например, переводчик DeepL требует 11 евро за перевод 1 миллиона знаков, причём заплатить там нельзя не только с российских банковских карт, но даже и с казахстанских. Ещё одна проблема динамического перевода кроется в том, что для этого используются алгоритмы машинного обучения, которые не всегда точны из-за множества факторов: переносные значения слов, особенности построений предложений, различная плотность передачи информации в языках и многое другое. Например, часто высмеиванию подвергаются переводы в китайском маркетплейсе AliExpress, специалист по локализации которого объяснял, что некоторые ошибки могут случаться из-за того, что плотность информации в китайском языке очень высокая: язык стремится к двусложным словам, т. е. к двум иероглифам, а условный

«корень» слова может выражаться одним символом, при этом в китайском нет пробелов, даже знаки препинания (точки, запятые) появились в нём только в XX веке. Кроме того, название товара на AliExpress, как правило, очень длинное, хаотичное и похоже на SEO, т. е. продавец добавляет туда всё, что имеет хотя бы косвенное отношение к товару, из принципа «больше напишу — по большому количеству запросов покажусь». Именно поэтому из 61 знака в китайском названии может получиться около 300 на русском, что в пять раз больше оригинала. [1]

1.2 Цель проекта

Мы поставили перед собой цель: разработать MVP сервиса для локализации, который может принимать нужные строки приложения, а затем выводить их переведёнными на нужные языки.

1.3 Задачи проекта

Для достижения цели мы разделили проект на следующие задачи между участниками:

- обучить модели
- разработать API для доступа к модели
- разместить облачный сервер
- разработать MVP (собрать всё воедино)

2 РАБОТА НАД ПРОЕКТОМ

2.1 Суть проекта

Проект представляет собой сервис для динамической локализации приложений. Разработчики других приложений могут подключаться к нашему сервису, при помощи API обращаться к модели, написанной на основе алгоритмов машинного обучения, которая может переводить строки и возвращать их на другом языке. Интеграция с нашим сервисом поможет разработчикам приложений решить проблему адаптации для другой страны удобно и просто, даже не нужно будет перезагружать приложение для применения перевода.

2.2 Процессы работы над проектом

Ввиду специфики проекта у нас было три глобальные задачи: подготовить backend, который поддерживает API (этим занимались Тимур Толкачёв и Бахадыр Ахмедов), frontend (этим занимался Алексей Данилевский) и обучить модель для перевода (эта задача была возложена на меня и Дениса Скворцова). Периодически мы созванивались для синхронизации нашей работы, обсуждения промежуточных итогов, получения фидбэка от Игоря: в среднем мы это делали раз в неделю, иногда два раза, иногда раз в 2 недели.

2.2.1 Определение целевых языков

Вместе с командой мы обсудили, с какими языками будет работать наш сервис, поскольку под каждый язык нужно искать параллельный корпус (большие собрания текстов с выравниванием по предложениям с сопоставлением одного языка с другим) и обучать модель, что занимает много времени и вычислительных ресурсов. Исходя из того, в каких языках у нас самих есть познания, а также того, какие теоретически могут быть полезны, мы выделили следующие языки как целевые: английский, русский, немецкий, французский, китайский и турецкий. При этом основным языком всё равно

является английский: все изначальные строки вводятся именно на английском, а с него переводятся на все остальные.

2.2.2 Определение нужных сторонних ресурсов

Поскольку не все задачи мы можем решить своими силами, нам нужно было использовать некоторые сторонние сервисы. Для обучения мы использовали модели Marian Helsinki-NLP и Google T5 (для увеличения вычислительной мощности мы покупали подписку Google), для бэкенда мы использовали подписки Amazon EC2 (аренда облачных вычислений, для того, чтобы сервис мог работать на удалённом сервере) и Amazon S3 (хранение картинок для приложения).

Из open-source датасетов для обучения мы использовали: KazParC (английский-турецкий, английский-русский) [2], Opus Books (английский-французский) [3], WMT14 (английский-немецкий) [4] и Wlhb/Translation-Chinese-2-English (английский-китайский) [5]. Кроме того, для дообучения мы сгенерировали перевод датасета из книг: мы взяли датасет ISBNDB с данными о книгах, который был переведён через переводчик DeepL.

2.2.3 Обучение моделей

Одной из главных веток работы над проектом является работа с моделями, основанных на алгоритмах машинного обучения. Модели обучались при помощи вышеуказанных датасетов и моделей при помощи фреймворка Google Colab. Поскольку эта задача лежала на мне и Денисе Скворцове, я об этом расскажу более подробно далее.

2.2.4 Размещение облачного сервера

Главными задачами backend/серверной части проекта было взаимодействие с БД, взаимодействие с моделью и интерфейс программирования приложения. Для этого использовались библиотеки SQLAlchemy с psycopg2, uvicorn, PyTorch и fastapi. Программа была размещена в docker-контейнере и расположена на удалённом сервере от

Amazon. Выглядит это так, как показано на Рисунке 1.

```
__py:1154: UserWarning: "translation" task was used, instead of "translation_XX_to_YY", defaulting to "translation_en_to_de"
2024-12-17 00:44:13 server | warnings.warn(
2024-12-17 00:44:13 server | Device set to use cpu
2024-12-17 00:44:13 server | INFO:server.src.migrations.migration:Проверяем существование таблиц...
2024-12-17 00:44:13 server | INFO:server.src.migrations.migration:Таблицы не найдены, запускаем миграции...
2024-12-17 00:44:13 server | INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
2024-12-17 00:44:13 server | INFO [alembic.runtime.migration] Will assume transactional DDL.
2024-12-17 00:44:13 server | INFO [alembic.runtime.migration] Running upgrade 19d5fac3ae0b -> a1078f7452c2, Changing field 'price' back to number
2024-12-17 00:44:19 server | Device set to use cpu
2024-12-17 00:44:21 server | Device set to use cpu
2024-12-17 00:44:24 server | Device set to use cpu
2024-12-17 00:44:26 server | Device set to use cpu
2024-12-17 00:44:26 server | /usr/local/lib/python3.10/site-packages/transformers/pipelines/__init__
__py:1154: UserWarning: "translation" task was used, instead of "translation_XX_to_YY", defaulting to "translation_en_to_de"
2024-12-17 00:44:26 server | warnings.warn(
2024-12-17 00:44:26 server | Device set to use cpu
2024-12-17 00:44:26 server | INFO:server.src.migrations.migration:Проверяем существование таблиц...
2024-12-17 00:44:26 server | INFO:server.src.migrations.migration:Таблицы не найдены, запускаем миграции...
2024-12-17 00:44:26 server | INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
2024-12-17 00:44:26 server | INFO [alembic.runtime.migration] Will assume transactional DDL.
2024-12-17 00:46:28 server | INFO: Stopping reloader process [1]
2024-12-17 16:05:35 server | INFO: Will watch for changes in these directories: ['/app']
2024-12-17 16:05:35 server | INFO: Uvicorn running on http://0.0.0.0:8080 (Press CTRL+C to quit)
2024-12-17 16:05:35 server | INFO: Started reloader process [1] using StatReload
2024-12-17 16:05:45 server | Device set to use cpu
2024-12-17 16:05:47 server | Device set to use cpu
2024-12-17 16:05:53 server | Device set to use cpu
```

Рисунок 1 – Ответы сервера на действия пользователей

Кроме того, нам нужно было создать интерфейс API для взаимодействия с моделями, чтобы получать json и обращаться на виртуальный выделенный сервер (instance). Информация о нашем сервере предоставлена на Рисунке 2.

Instance summary for i-0ab6dc9d51b9dc5b2 (LocSer) [Info](#)

[Connect](#)[Instance state ▼](#)[Actions ▼](#)

Updated less than a minute ago

Instance ID

i-0ab6dc9d51b9dc5b2

Private IPv4 addresses

172.31.33.49

Instance state

Running

Hostname type

IP name: ip-172-31-33-49.eu-central-1.compute.internal

Instance type

t2.micro

Public IPv4 address



3.120.132.186 | [open address](#)

IPv6 address

–

Public IPv4 DNS



ec2-3-120-132-186.eu-central-1.compute.amazonaws.com
| [open address](#)

Private IP DNS name (IPv4 only)



ip-172-31-33-49.eu-central-1.compute.internal

Answer private resource DNS name

IPv4 (A)

Elastic IP addresses

–

Рисунок 2 – Виртуальный выделенный сервер нашего проекта

2.2.5 Создание сервисного слоя мобильного приложения

Для демонстрации работы продукта мы создали мобильное приложение для iOS: для этого использовались Xcode, Swift, SwiftUI, MVVM+Combine и

SwiftPM. Таким образом, мы создали визуальную оболочку для презентации работы нашего продукта в виде мобильного приложения, которое сможет выводить картинки и тексты списком, как на Рисунке 3.

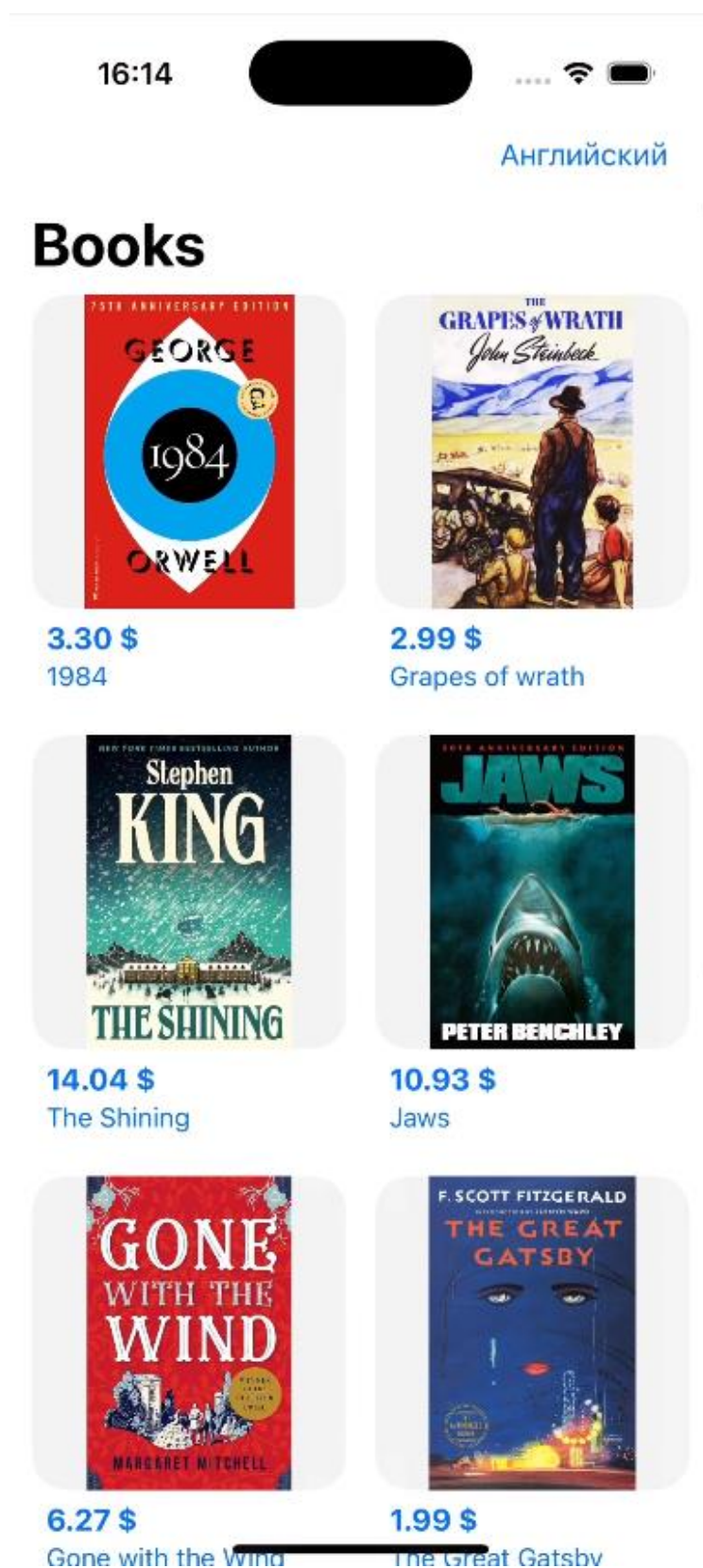


Рисунок 3 – Внешний вид прототипа мобильного приложения

2.3 Мои задачи

В проекте у меня была роль специалиста по Machine Learning и Data Science. Совместно с Денисом Скворцовым мы должны были обучить модель.

Первым делом мы должны были разобраться с Hugging Face, где располагаются миллионы моделей для машинного обучения. Мы изучили и настроили базовую модель [6] и попробовали её натренировать переводить пары английский-русский на датасетах Opus Books и KazParC (Kazakh Parallel Corpus), но результаты получались неудовлетворительными: показатель Loss был достаточно высоким (см. Рисунок 4), а переводы были неточными (см. Рисунок 5).

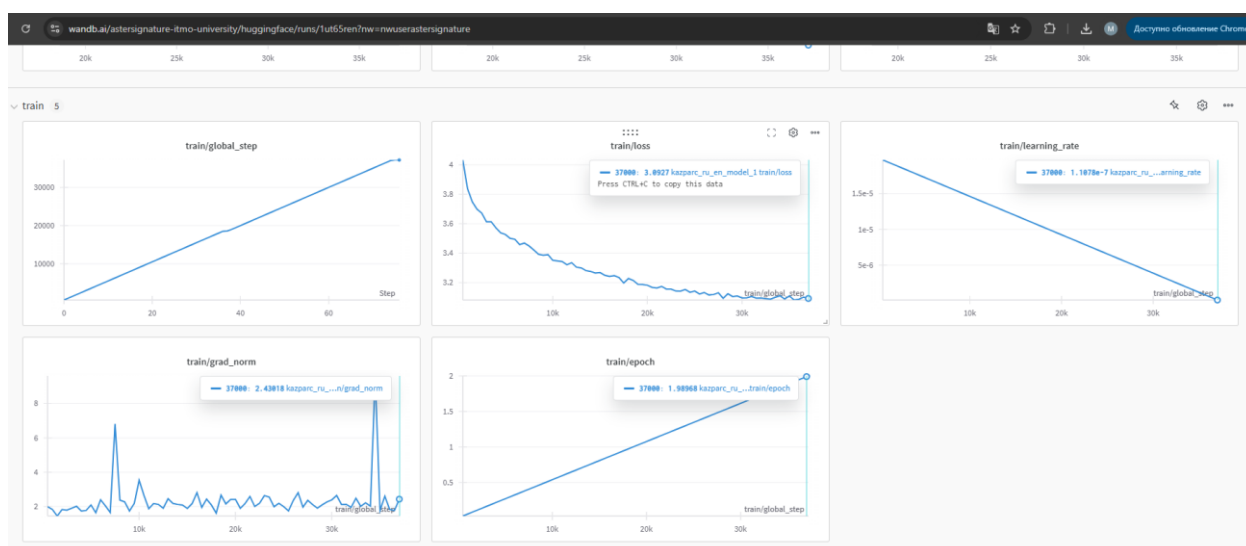


Рисунок 4 – Показатели первых неудачных моделей, натренированных на перевод пар английский-русский

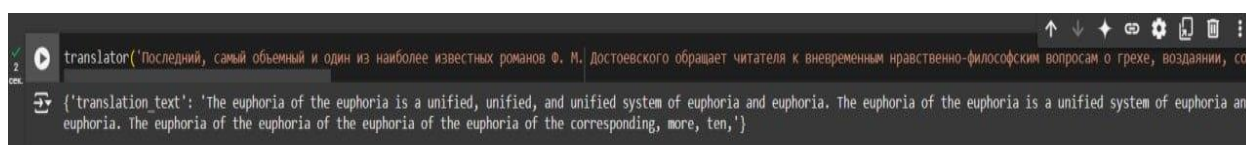
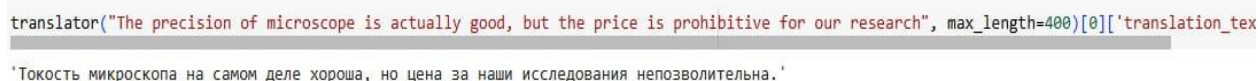


Рисунок 5 – Пример неудачного перевода

Было понятно, что нужно что-то менять, поэтому с того момента мы с Денисом разделились: я работал над тем, чтобы интегрировать нашу работу с переводчиком DeepL (заодно разобрался с тем, как сделать так, чтобы программа принимала и выдавала строки в json-формате, а также запустить её

процессы асинхронно, что потом даже нашло отголоски в финальном проекте, поскольку я отправлял код Тимуру, который в основном и собирал готовый продукт), а Денис продолжал пытаться продолжать решать проблему машинного обучения самостоятельно, используя другие датасеты и модели (иногда я помогал ему с поиском датасетов для обучения). Мы пытались решить проблему того, что датасеты, как правило, были специализированными: некоторые из них представляли собой переводы статей Википедии (что означало бы, что модель всегда переводила бы очень научным языком), некоторые являлись переводами официальных документов (что, опять же, заставило бы модель «разговаривать» официально-деловым стилем речи), поэтому нужно было либо находить параллельные корпуса на более общие тематики, либо комбинировать сразу несколько. Помимо этого, нужно было заниматься и обработкой датасетов: не только нормализовывать данные по тематикам, но и удалять неестественно длинные (мы разобрались, что на изначальных стадиях в основном именно из-за этого модель выдавала слишком длинные ответы) и сокращать сами датасеты. В том числе для диверсификации тематик я впоследствии генерировал датасет из переводов названий книг при помощи DeepL, поскольку для презентации мы выбрали именно перевод книг. Результаты получились достаточно неплохими: например, перевод пар английский-русский получился с показателем Loss, равным 0,0186, да и сами строки переводились близко к истине (см. Рисунок 6).



```
translator("The precision of microscope is actually good, but the price is prohibitive for our research", max_length=400)[0]['translation_text']
```

'Точность микроскопа на самом деле хороша, но цена за наши исследования непопустительна.'

Рисунок 6 – Перевод пар английский-русский готовой натренированной моделью

После обучения моделей мы с Денисом прописывали локалы и эндпоинты для обращения к моделям, а также консультировали Тимура по поводу того, как заставить backend-часть обращаться к ним (модели были выгружены на сервер

Hugging Face и расположены на репозиториях: "Goshective/kazparc_en_ru_marian_1", "Goshective/wlhb_en_zh_marian_1", "Goshective/kazparc_en_tr_marian_1", "Goshective/wmt14-en-de_marian_1", "Goshective/opus_books_model_french").

2.4 Анализ моей работы

Проанализировав свою работу, я думаю, что я сделал меньше, чем хотелось бы, однако это не повлияло на результат, поскольку Денис очень хорошо справился со своей работой, которая была смежна с моей. Большую часть времени работы с моделями провёл как раз Денис, а мне не хватило из-за академической загруженности. Кроме того, в распоряжении Дениса был компьютер, а у меня только ноутбук, который я ежедневно использовал и брал на учёбу в университете или для выполнения домашних заданий, что замедляло обучение, поскольку это и так долгий процесс, который несколько раз у меня обрывался (более того, Игорь впоследствии докупил Денису вычислительные мощности от Google, и процесс пошёл ещё быстрее). Помимо того, часть времени я потратил на работу с переводчиком DeepL, которая не попала в финальный проект и была нам нужна «на всякий случай».

2.5 Взаимодействие с командой

Я считаю, что нам действительно удалось работать в команде. На протяжении всего проекта мы устраивали групповые звонки, на которых обсуждали результаты нашей работы, периодически помогали и подсказывали друг другу, несколько раз устраивали звонки даже без Игоря, просто обсудить работу. Из-за специфики своей задачи я в основном общался с Денисом, с которым мы распределяли вычислительные часы и делились результатами своей работы, а на финальных стадиях проекта я общался и с Тимуром, которому помогал с синхронизацией backend-части проекта с моделями для перевода, которыми занимались мы с Денисом.

2.6 Взаимодействие с руководителем

Наш руководитель Игорь Манаков всегда был на связи, отвечал на вопросы, помогал справляться с трудностями, интересовался нашими успехами, прислушивался к нашим мнениям и направлял работу в нужное русло. Как правило, он инициировал наши встречи и групповые звонки, всегда следил за ходом выполнения работы. Кроме того, Игорь действительно заботился о том, чтобы мы получили новые знания в результате нашего участия в проекте: привлекал своих знакомых, которые помогали в некоторых аспектах, отправлял обучающие видео и другие материалы. Единственный недостаток заключается в том, что иногда его указания были немного противоречивы и не всегда понятны, что немного путало меня и других членов команды, но в конце концов результат получился очень неплохим, поэтому я оцениваю результат работы Игоря как отличный, а также считаю, что у него есть задатки лидера, под управлением которого команда может не только справляться с рабочими задачами, но и саморазвиваться.

ЗАКЛЮЧЕНИЕ

Поскольку целью проекта было разработать MVP сервиса для локализации, а не готовый сервис, то я считаю, что с ней мы справились весьма успешно. Из этого может получиться достаточно успешный стартап, для которого уже подготовлена определённая база, однако для выхода на глобальный рынок нужно работать над точностью перевода, для чего нужно больше вычислительных ресурсов, более мощные компьютеры, более ёмкие, вместительные и тяжёлые датасеты, на которых можно более качественно обучить модель (желательно также, чтобы они были заточены под специфику e-commerce), а также более квалифицированные специалисты. Таким образом, проекту нужны финансовые вложения, после которого он может стать полноценным, сотрудничать с бизнесом и приносить прибыль, если начнёт работать, например, по подписке.

Несмотря на то, что, по моему мнению, я приложил меньше усилий, чем мои партнёры по команде, я считаю, что я научился некоторым вещам: например, работе с Hugging Face, хостингу удалённых моделей, попробовал поработать с NLP (Natural Language Processing), поскольку Денис всё равно поделился со мной готовым кодом и результатами своей работы. Таким образом, лично для себя я считаю участие в проекте полезным, а также думаю, что и я был полезен для команды.

Мне было очень приятно поработать в такой дружной команде, я полагаю, что помимо прикладных навыков я улучшил и навыки работы в команде: я учился работать с малознакомыми мне людьми (например, с Денисом я нигде не пересекался до этого проекта), впервые встретился с тем, что нужно было отчитываться по проделанной работе на групповых звонках, которые тоже нужно было учитывать в моём расписании. Раньше я если и участвовал в групповых проектах, то как правило это были небольшие программы, которые писали мы с друзьями, поэтому общение в рамках проекта всегда было неформальным, а здесь я впервые встретился с

соблюдением формальных/полужформальных норм во время выполнения достаточно большого проекта. Я получил ценный опыт в командном создании продукта, как и мои партнёры и руководитель.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Habr. Блог компании AliExpress Россия. Экваторист со светоотражающим звуком на солнечной батарее: что не так с переводами на AliExpress. URL: https://habr.com/ru/companies/aliexpress_russia/articles/565800/ Дата обращения: 02.01.2025.
2. Hugging Face. issai/kazparc · Datasets at Hugging Face. URL: <https://huggingface.co/datasets/issai/kazparc>. Дата обращения: 04.01.2025
3. Hugging Face. Helsinki-NLP/opus_books · Datasets at Hugging Face. URL: https://huggingface.co/datasets/Helsinki-NLP/opus_books. Дата обращения: 04.01.2025
4. Hugging Face. wmt/wmt14 · Datasets at Hugging Face. URL: <https://huggingface.co/datasets/wmt/wmt14/tree/main>. Дата обращения: 04.01.2025
5. Hugging Face. wlhb/Translation-Chinese-2-English. URL: <https://huggingface.co/datasets/wlhb/Transaltion-Chinese-2-English>. Дата обращения: 04.01.2025
6. Hugging Face. Translation. URL: <https://huggingface.co/docs/transformers/en/tasks/translation>. Дата обращения: 03.01.2025.

ПРИЛОЖЕНИЕ. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1. **Название:** Разработка сервиса локализации мобильного приложения
2. **Цель (назначение):** Разработать MVP сервиса для локализации, который может принимать нужные строки приложения, а затем выводить их переведёнными на нужные языки.
3. **Сроки выполнения:** Начало – 31.10.2024 Окончание – 18.12.2024
4. **Исполнитель проекта (руководитель проекта):** Игорь Манаков
5. **Термины и сокращения:**

MVP (minimal viable product) – продукт, обладающий минимальными, но достаточными для удовлетворения первых потребителей функциями

Параллельный корпус – большие собрания текстов с выравниванием по предложениям с сопоставлением одного языка с другим.

Датасет – коллекция табличных данных

Loss (Функция потерь) – функция, характеризующая потери при неправильном принятии решений на основе наблюдаемых данных

6. Технические требования:

- сервис должен уметь переводить тексты асинхронно на выбранных языках
- сервис должен базироваться на удалённом сервере
- сервис должен поддерживать хостинг картинок
- сервис должен иметь API для доступа к модели

7. Содержание работы

Таблица 1 – Содержание работы

№	Этап плана	Сроки выполнения этапов	Ответственный за этап
---	------------	-------------------------------	--------------------------

1	Собрать данные	31.10.2024 – 06.11.2024	Калинин Марк Алексеевич
2	Изучить FastAPI	31.10.2024 – 10.11.2024	Ахмедов Бахадыр Бахтиёрович
3	Изучить Docker	31.10.2024 – 10.11.2024	Толкачёв Тимур Сергеевич
4	Познакомиться с Hugging Face	31.10.2024 – 10.11.2024	Скворцов Денис Александрович
5	Обработать данные	10.11.2024 – 17.11.2024	Скворцов Денис Александрович
6	Создать интерфейс для интеграции с backend	10.11.2024 – 17.11.2024	Калинин Марк Алексеевич
7	Добавить перевод навигационных компонентов	17.11.2024 – 20.11.2024	Калинин Марк Алексеевич
8	Разработать Endpoint'ы с конфигурацией мобильного приложения	10.11.2024 – 24.11.2024	Ахмедов Бахадыр Бахтиёрович
9	Развернуть сервер	10.11.2024 – 24.11.2024	Толкачёв Тимур Сергеевич
10	Оптимизировать модель	17.11.2024 – 01.12.2024	Скворцов Денис Александрович
11	Обучить модель	20.11.2024 – 01.12.2024	Калинин Марк Алексеевич
12	Интегрироваться в сервис	24.11.2024 – 01.12.2024	Толкачёв Тимур Сергеевич
13	Разработать контракт API	24.11.2024 – 01.12.2024	Ахмедов Бахадыр Бахтиёрович
14	Сверстать UI мобильного приложения	31.10.2024 – 08.12.2024	Данилевский Алексей Александрович
15	Сгенерировать документацию API	01.12.2024 – 08.12.2024	Ахмедов Бахадыр Бахтиёрович
16	Доработать модель	01.12.2024 – 20.12.2024	Скворцов Денис Александрович

17	Добавить кэширование данных	01.12.2024 – 20.12.2024	Толкачёв Тимур Сергеевич
18	Интегрировать S3	08.12.2024 – 20.12.2024	Ахмедов Бахадыр Бахтиёрович
19	Создать сервисный слой приложения	08.12.2024 – 20.12.2024	Данилевский Алексей Александрович

Примечание: этапы работы были скопированы из приложения Odoо и не полностью совпадают с реальностью, иногда не совпадают даже ответственные за этап.