МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ШКОЛА КОМПЬЮТЕРНЫХ НАУК

ОТЧЕТ ПО ПРОЕКТУ

РАЗРАБОТКА СЕРВИСА ДЛЯ ЛЕММАТИЗАЦИИ РУССКОЯЗЫЧНЫХ ТЕКСТОВ СОЦИАЛЬНЫХ СЕТЕЙ С ПОМОЩЬЮ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ

ФГАОУ ВО «Тюменский государственный университет»

(наименование организации)

Кафедра программного обеспечения

(наименование структурного подразделения)

|  |  |  |
| --- | --- | --- |
| Выполнил  обучающийся 4 курса,  МОиАИС-21.01 группы | (подпись) | Копырин Евгений  Александрович |
| Выполнил  обучающийся 4 курса,  МОиАИС-21.01 группы | (подпись) | Сагадеев Артур  Ринатович |
| Научный руководитель  Проекта  Доцент (к.н.) | (подпись) | Глазкова Анна  Валерьевна |
|  |  |  |

Оглавление

[Введение 4](#_Toc185588370)

[Глава I. Существующие подходы и технологии для лемматизации текстов 8](#_Toc185588371)

[1.1. Современные инструменты для лемматизации текстов 8](#_Toc185588372)

[1.2. Большие языковые модели и их перспектива для лемматизации текстов 11](#_Toc185588373)

[Глава II. Исследование возможностей больших языковых моделей для задачи лемматизации и сравнение с текущими современными решениями 13](#_Toc185588374)

[2.1. Данные 13](#_Toc185588375)

[2.2. Снятие бейслайнов с текущих современных решений для задачи лемматизации 15](#_Toc185588376)

[2.3. Выбор большой языковой модели 16](#_Toc185588377)

[2.4. Prompt engineering: Подбор шаблонов запросов к большой языковой модели для различных вариантов лемматизации 19](#_Toc185588378)

[2.4.1. Лемматизация токенов без указания контекста и части речи 19](#_Toc185588379)

[2.4.2. Лемматизация токенов с учётом контекста 21](#_Toc185588380)

[2.4.3. Лемматизация токенов с учётом части речи 22](#_Toc185588381)

[2.4.4. Результаты замеров точности после подбора шаблонов запросов 25](#_Toc185588382)

[Глава III. Проектирование и реализация API и WEB-сервисов для лемматизации 26](#_Toc185588383)

[3.1. Требования к реализуемому проекту 26](#_Toc185588384)

[3.2. Используемые технологии 27](#_Toc185588385)

[3.2.1. Языки программирования 27](#_Toc185588386)

[3.2.2. СУБД 29](#_Toc185588387)

[3.3. Функциональная схема сервиса 29](#_Toc185588388)

[3.4. Разработка API для лемматизации с использованием LLM 30](#_Toc185588389)

[3.5. Разработка backend-части для WEB-приложения и Telegram Mini App 32](#_Toc185588390)

[3.6. Проектирование и разработка базы данных для WEB-сервисов 33](#_Toc185588391)

[3.7. Проектирование и реализация интерфейсов WEB-приложения и Telegram Mini App 34](#_Toc185588392)

[3.8. Сборка и развертывание сервиса 37](#_Toc185588393)

[Глава IV. Функциональные возможности WEB-сервисов 38](#_Toc185588394)

[4.1. Регистрация и авторизация пользователей 38](#_Toc185588395)

[4.2. Выбор варианта лемматизации 40](#_Toc185588396)

[4.3. Хранение истории запросов 44](#_Toc185588397)

[Глава V. Тестирование. 46](#_Toc185588398)

[СПИСОК ЛИТЕРАТУРЫ 48](#_Toc185588399)

# Введение

Лемматизация (lemmatization) – процесс приведения формы слова к начальной форме (лемме). Этот этап играет важную роль в обработке естественного языка, так как позволяет учитывать различные грамматические формы одного и того же слова как эквивалентные.

Лемматизация применяется в самых различных областях, например:

* В поисковых системах для улучшения релевантности и полноты поиска
* В чат-ботах и голосовых помощниках для понимания запроса пользователя без учёта грамматических форм слова
* При анализе тональности текста для правильного определения эмоциональной окраски независимо от формы слов
* В юридическом анализе текстов для упрощения извлечения ключевой информации

В данной работе лемматизация применяется для обработки русскоязычных текстов из социальных сетей, поскольку в таких текстах постоянно появляются новые слова, сокращения и аббревиатуры, которые порой создают проблемы моделям, основанным на правилах и эвристиках. Именно поэтому, в качестве моделей будут рассмотрены большие языковые модели, основанные на управлении с помощью запросов (промптов), будет оценена их перспективность в данной ситуации, проведено сравнение с текущими решениями, основанными на правилах и будет создан свой сервис для лемматизации русскоязычных текстов социальных сетей с помощью больших языковых моделей.

**Цель проекта:** разработать сервис для лемматизации русскоязычных текстов социальных сетей с помощью больших языковых моделей, который ускорит процесс разметки русскоязычных текстов социальных сетей разметчиками Национального корпуса русского языка, что позволит быстрее размечать корпуса, содержащие русскоязычные тексты социальных сетей

**Для решения поставленной цели определены следующие задачи:**

* Провести анализ требований пользователей (разметчики Национального корпуса русского языка)
* Подобрать размеченный набор данных по домену
* Изучить существующие инструменты для лемматизации, измерить их точность и выявить недостатки
* Изучить существующие большие языковые модели, основанные на инструкциях
* Выбрать наиболее точную большую языковую модель, основанную на инструкциях для задачи лемматизации
* Реализовать программный интерфейс приложения (API), который будет возвращать предсказанную лемму слова
* Разработать архитектуру WEB-приложения, включая клиентскую и серверную части
* Спроектировать базу данных для WEB-приложения • Реализовать WEB-приложение на основе API с удобным для пользователя интерфейсом
* Реализовать Telegram Mini App на основе WEB-приложения

График (план) работы над задачами с уточнением формулировок задач представлен в таблице 1.

*Таблица 1. График (план) работы над задачами с уточнениями формулировок задач*

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Формулировка задачи** | **Сроки начала и завершения задачи** | **Исполнитель** |
| 1 | Первоначальная постановка задач | 16.09-22.09 | Копырин Евгений (Аналитик) |
| 2 | Анализ структуры и содержания наборов данных, выбор набора данных по домену (жанру) | 23.09 – 29.09 | Копырин Евгений (Аналитик) |
| 3 | Изучение существующих инструментов для лемматизации (PyMorphy2, PyMystem, Stanza), измерение точности (accuracy) предсказания лемм и выявление недостатков | 30.09 – 06.10 | Копырин Евгений (Аналитик) |
| 4 | Изучение существующих больших языковых моделей, основанных на инструкциях | 07.10 – 13.10 | Копырин Евгений (Аналитик) |
| 5 | Тестирование LLM T-Lite-Instruct (Задача выбора наиболее точной большой языковой модели, основанной среди T-lite-instruct и Saiga-llama3-8b) | 14.10 – 20.10 | Сагадеев Артур (Разработчик) |
| 6 | Тестирование LLM Saiga-LLaMa-3-8b (Задача выбора наиболее точной большой языковой модели, основанной среди T-lite-instruct и Saiga-llama3-8b) | 14.10 – 20.10 | Копырин Евгений (Аналитик) |
| 7 | Подбор шаблонов запросов (промптов) для выбранной большой языковой модели (Saiga-llama3-8b) для повышения точности | 21.10 – 27.10 | Копырин Евгений (Аналитик) |
| 8 | Проектирование и создание базы данных для WEB-приложения | 28.10 – 03.11 | Сагадеев Артур (Разработчик) |
| 9 | Подбор хостинга для LLM | 29.10 – 03.11 | Копырин Евгений (Аналитик) |
| 10 | Разработка API для взаимодействия с большой языковой моделью и разработка API для взаимодействия с базой данных | 04.11 – 10.11 | Сагадеев Артур (Разработчик) |
| 11 | Разработка frontend-части WEB-приложения | 11.11 – 14.11 | Сагадеев Артур (Разработчик) |
| 12 | Разработка frontend-части Telegram Mini App | 15.11 – 17.11 | Сагадеев Артур (Разработчик) |
| 13 | Объединение frontend-части и backend-части для WEB-приложения и Telegram Mini App | 18.11 – 24.11 | Сагадеев Артур (Разработчик) |
| 14 | Тестирование WEB-приложения и Telegram Mini App | 25.11 – 01.12 | Копырин Евгений (Аналитик) |
| 15 | Устранение ошибок во frontend- и backend-части WEB-приложения и Telegram Mini App | 25.11 - 01.12 | Сагадеев Артур (разработчик) |
| 16 | Деплой WEB-приложения и Telegram Mini App | 02.12 – 08.12 | Сагадеев Артур (Разработчик) |
| 17 | Тестирование развернутых приложений | 09.12 - 15.12 | Копырин Евгений (Аналитик) |
| 18 | Устранение неполадок в задеплоенных приложениях | 09.12 - 15.12 | Сагадеев Артур (Разработчик) |

**Заказчик:** Национальный корпус русского языка

**Проблема:** Недостаточная точность лемматизации русского языка в рамках текстов из социальных сетей, что влияет на качество разных приложений для обработки естественного языка.

# Глава I. Существующие подходы и технологии для лемматизации текстов

## 1.1. Современные инструменты для лемматизации текстов

На данный момент используются следующие основные подходы для лемматизации текстов:

1. **Словарные методы**

Используют предопределенные словари с формами слов и их леммами.

Преимущества: высокая точность для языков с подробными словарями

Недостатки: редкие слова из словарей, либо слова, которых нет в словаре приводятся к лемме с ошибками.

1. **Правила и эвристики**

Используют заранее прописанные правила для преобразования словоформы к лемме.

Преимущества: простота реализации

Недостатки: низкая точность, применимость только к определенным случаям

1. **Комбинированные подходы**

Сочетают в себе словарные методы и правила и эвристики.

Преимущества: баланс между точностью и производительностью

Недостатки: низкая точность при приведении слов, которые не содержатся в словарях или для типовых грамматических конструкций для которых не прописаны правила и эвристики

1. **Методы на основе машинного обучения**

Используют машинное обучение для анализа и нормализации слов.

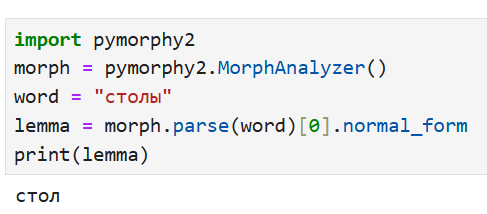
Преимущества: могут учитывать контекст и обрабатывать сложные случаи.

Недостатки: высокая потребность в ресурсах и сложность настройки, требование к наличию датасета с достаточным для обучения размером.

Среди существующих инструментов для лемматизации популярными решениями на данный момент являются PyMorphy2 [1], PyMystem3 [2] и Stanza [3].

**PyMorphy2** – библиотека Python для морфологического анализа и лемматизации русскоязычных текстов. Использует словарную базу OpenCorpora. Работает оффлайн и не требует обучения. Может возвращать сразу несколько вариаций лемм, не учитывает контекст и морфологические характеристики.

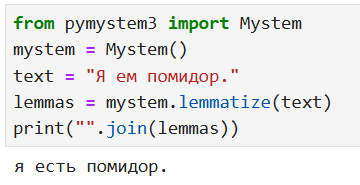
Пример использования библиотеки PyMorphy2 для задачи лемматизации представлен на рисунке 1.



*Рис 1. Пример использования библиотеки PyMorphy2 для задачи лемматизации.*

**PyMystem3** – библиотека-обёртка для Python, использующая приложение MyStem для морфологического анализа и лемматизации русскоязычных текстов. Использует правила и эвристики в качестве подхода для лемматизации. Работает медленнее, чем PyMorphy2, но поддерживает ввод многословных выражений, однако, их контекст аналогично не учитывается.

Пример использования библиотеки PyMystem3 для задачи лемматизации представлен на рисунке 2.



*Рис. 2. Пример использования библиотеки PyMystem3 для задачи лемматизации.*

**Stanza** – библиотека Python для обработки естественного языка. Выполняет задачи лемматизации, токенизации, морфологического и синтаксического анализа. Помимо русского языка поддерживает множество других языков.

Пример использования библиотеки Stanza для задачи лемматизации представлен на рисунке 3.



*Рис. 3. Пример использования библиотеки Stanza для задачи лемматизации.*

## 1.2. Большие языковые модели и их перспектива для лемматизации текстов

Как было сказано выше, модели, основанные на правилах или словарном подходе, не всегда точно лемматизируют новые слова, которых нет в корпусе или которые не подчиняются прописанным правилам и эвристикам.

В этом случае возникает идея использовать модель, основанную на архитектуре нейронных сетей для таких задач, ведь нейронные сети хорошо подходят для того, чтобы улавливать сложные закономерности и вполне могут быть приспособлены для поиска лемм для новых слов, постоянно появляющихся в текстах социальных сетей.

Существуют и более продвинутые модели, которые используют нейронные слои в комбинации с другими решениями. Одним из таких решений являются **трансформеры** [4] **-** модели, которые обрабатывают входные данные с помощью механизма внимания (attention) и нейронных слоев.

На основе архитектуры трансформеров были основаны **большие языковые модели**.

**Большая языковая модель (Large Language Model, LLM)** — это модель машинного обучения, основанная на архитектуре трансформеров и обученная на огромных объемах текстовых данных, способна выполнять широкий спектр задач обработки естественного языка. Может содержать в себе сотни миллионов и миллиарды параметров.

Наиболее популярными большими языковыми моделями сейчас являются:

1. GPT (Generative Pre-trained Transformer), актуальная версия на данный момент – GPT-4o – подходит для задач генерации текста [5].
2. BERT (Bidirectional Encoder Representations from Transformers) - подходит для задач классификации, извлечения информации, анализа тональности [6].
3. T5 (Text-to-Text Transfer Transformer) - универсальная модель, переводящая любую задачу по обработке естественного языка в текстовый формат [7].
4. PaLM, LLAMA, BLOOM – новейшие модели, разработанные для повышения производительности и эффективности в NLP задачах

Некоторые из этих моделей, находящихся в открытом доступе для разработчиков и доступные для бесплатной разработки, заточены только под англоязычные тексты. В связи с этим многие инженеры машинного обучения занимаются **дообучением (fine-tuning)** на корпусах русскоязычных текстов, чтобы использовать модель в контексте русскоязычных текстов.

Примерами больших языковых моделей, дообученных на корпусах русскоязычных текстов являются:

* Saiga-LLaMa-3
* T-lite-instuct-0.1.

Существуют различные виды данных моделей, отличающиеся кол-вом параметров: от восьми до восьмидесяти миллиардов.

В качестве примеров больших языковых моделей данные модели взяты непросто так – сейчас они являются наиболее актуальными в сообществе инженеров машинного обучения, работающих с обработкой русскоязычных текстов, а также продолжают постоянно обновляться.

Подробнее о моделях saiga-llama-3 и t-lite-instruct-0.1 и о конкретном выборе большой языковой модели для дальнейших исследований будет описано в главе II.

# Глава II. Исследование возможностей больших языковых моделей для задачи лемматизации и сравнение с текущими современными решениями

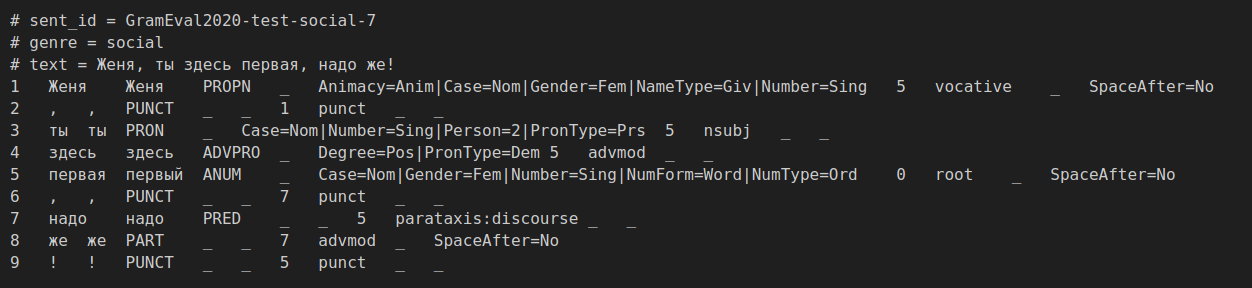
## 2.1. Данные

Поскольку целью проекта является разработка сервиса для лемматизации русскоязычных текстов социальных сетей с помощью больших языковых моделей, то, соответственно, в качестве данных нам нужен корпус с русскоязычными текстами из социальных сетей.

Такой корпус имеется - это данные с соревнований по анализу морфологии русского языка GramEval-2020 [8] (Таблица 2).

Датасет оформлен в формате CoNLL-U – это набор текстов, каждый токен которых разобран морфологически.

Пример структуры одного из предложений датасета представлен на рисунке 4.



*Рис. 4. Пример структуры предложения из корпуса формата CoNLL-U.*

Предложение разбивается на отдельные токены, прописаны морфологические характеристики каждого токена.

В первом столбце указана текущая словоформа токена, во втором - лемма токена. Именно со значениями во втором столбце (с леммами) и будут сравниваться значения, предсказанные большой языковой моделью.

В третьем столбце указана часть речи. Для наименования морфологических характеристик в корпусах CoNLL-U используется стандартизированный формат морфологической и синтаксической разметки UD 2.0: например, PROPN - это имя собственное, а ANUM – порядковое числительное. Лемматизация с учетом части речи - один из видов лемматизации в проекте.

Также, имеются и другие морфологические характеристики, например: число, лицо, род. Иногда имеются специальные пометки, например: foreign - это значит, что слово не является русскоязычным и, чаще всего, для ситуации с анализом русскоязычных текстов, его словоформа сохраняется.

Предобработка значений входящих токенов не производится – поскольку нужно учитывать максимально разнообразные варианты словоформ в естественном виде для обучения модели.

*Таблица 2. Информация о данных в корпусе GramEval-2020*

|  |  |
| --- | --- |
| **Статистический показатель** | **Значение** |
| Кол-во токенов в корпусе | 1122 |
| Кол-во текстов в корпусе | 114 |
| Распределение частей речи в корпусе | Существительных – 64;  глаголов – 108;  прилагательных – 66;  вспомогательных глаголов – 5;  местоимений – 64;  числительных – 4;  порядковых числительных – 8;  имён собственных – 37;  детерминативов – 38;  частиц – 57;  наречий – 5;  местоимённых наречий – 22;  предлогов – 95;  символов – 30;  сочинительных союзов – 39;  подчинительных союзов – 13;  междометий – 3;  предикативов – 8;  вводных слов – 6;  комбинированных частей речи (COM) – 1;  неопределенных – 33. |

## 2.2. Снятие бейслайнов с текущих современных решений для задачи лемматизации

Для эксперимента, в качестве современных инструментов для выполнения задачи лемматизации были взяты следующие инструменты:

* PyMorphy2
* PyMystem3
* Stanza

Замер точности предсказания лемм на каждом из этих инструментов произведен с использованием данных из корпуса с русскоязычными текстами социальных сетей GramEval-2020. В качестве метода замера точности было произведено сравнение лемм, значения которых уже размечены в датасете с леммами, сгенерированными моделями (Таблица 3).

*Таблица 3. Результаты замера точности предсказания лемм при использовании библиотек Pymorphy2, Pymystem3, Stanza на данных корпуса GramEval-2020*

|  |  |  |
| --- | --- | --- |
| **Библиотека** | **Кол-во ошибок предсказания** | **Точность (Accuracy)** |
| Pymorphy2 | 119 | 89,39% |
| Pymystem3 | 274 | 75,58% |
| Stanza | 122 | 89,13% |

Библиотеки PyMorphy2 и Stanza показали примерно одинаковые результаты по точности, точность предсказаний PyMystem3 примерно на 14 процентов ниже, чем у PyMorphy2 и Stanza. Таким образом, максимально достигнутая точность среди трёх инструментов составила 89,39%.

## 2.3. Выбор большой языковой модели

Для достижения поставленной цели были выбраны две следующие большие языковые модели для исследования:

* Saiga-LLaMa-3 (8b)
* T-Lite-instruct-0.1 (8b)

**Saiga-LLaMa-3** – модель, представляющая собой вариацию модели **LLaMa** (Large Language Model Meta AI), которая была разработана Meta (признана экстремистской организацией на территории РФ). Дообучена на корпусах русскоязычных текстов.

**T-Lite-Instruct-0.1** — это облегченная версия модели **T-Instruct.** Обучена на корпусах, содержащих тексты различных языков, в том числе и на русскоязычных корпусах.

Выбор обусловлен тем, что предложенные модели на данный момент являются актуальными в сообществе инженеров машинного обучения, имеющих дело с обработкой русскоязычных текстов и продолжают постоянно обновляться. Модели имеют одинаковое количество параметров – 8 миллиардов. Средний объем использования видеопамяти в процессе предсказания лемм – 10 гигабайт.

Так как данные LLM работают под управлением промптов (шаблонов запросов), первым делом были составлены разные базовые промпты для каждого подхода к лемматизации:

* Промпт для лемматизации без указания контекста и части речи

“Приведи слово '{word}' к начальной форме (лемме).

В ответе напиши только одно слово - лемму, без комментариев и предложений.

Все символы по типу ')', '-', '/', а также знаки препинания сохраняй в изначальном виде.”

* Промпт для лемматизации с учетом контекста

Приведи слово '{word}' из предложения '{full\_sentence}' к начальной форме (лемме).

В ответе напиши только одно слово - лемму, без комментариев и предложений.

Все символы по типу ')', '-', '/', а также знаки препинания сохраняй в изначальном виде.

* Промпт для лемматизации с учетом части речи

Приведи слово '{word}' к начальной форме (лемме) с учетом части речи ({pos}).

В ответе напиши только одно слово - лемму, без комментариев и предложений.

Все символы по типу ')', '-', '/' а также знаки препинания сохраняй в изначальном виде.

Результаты замеров точности при прогонке всего датасета с использованием базовых шаблонов запросов **без шаблона чата** представлены в таблице 4. Лучшее значение метрики выделено полужирным шрифтом.

*Таблица 4. Показатели точности предсказания лемм большими языковыми моделями при разных подходах к лемматизации без использования шаблона чата при запросе*

|  |  |  |
| --- | --- | --- |
| **Подход к лемматизации** | **Точность (Accuracy) Saiga-LLaMa-3-8b** | **Точность (Accuracy) T-lite-instruct-0.5** |
| Лемматизация без учета контекста и части речи | 79,48% | 71,31% |
| Лемматизация с учетом контекста | 74,26% | 56,89% |
| Лемматизация с учетом части речи | **79,71%** | 74,33% |

Затем были построены и использованы шаблоны чата с двумя ролями: пользователь и система (роль для выдачи ответов от модели) для каждого подхода к лемматизации **(подробнее о построении шаблонов чата будет рассказано в следующем пункте 2.4)**

Результаты замеров точности при прогонке всего датасета с использованием базовых шаблонов запросов **с шаблоном чата** представлены в таблице 5. Лучшее значение метрики выделено полужирным шрифтом.

*Таблица 5. Показатели точности предсказания лемм большими языковыми моделями при разных подходах к лемматизации с использованием шаблона чата при запросе*

|  |  |  |
| --- | --- | --- |
| **Подход к лемматизации** | **Точность (Accuracy) Saiga-LLaMa-3-8b** | **Точность (Accuracy) T-lite-instruct-0.1** |
| Лемматизация без учета контекста и части речи | 83,08% | 79,05% |
| Лемматизация с учетом контекста | 77,16% | 60,07% |
| Лемматизация с учетом части речи | **84,22%** | 81,1% |

Таким образом, по показателям точности более высокое значение метрики Accuracy показывает Saiga-LLaMa-3-8b. Именно эта большая языковая модель и будет использована для дальнейших исследований.

## 2.4. Prompt engineering: Подбор шаблонов запросов к большой языковой модели для различных вариантов лемматизации

Как было установлено ранее, взаимодействие с выбранной моделью Saiga-LLaMa-3-8b происходит через подачу текстовых запросов (промптов). Соответственно, настройка промптов - это один из способов повышения точности. В этом пункте будет рассказано о том, как подбирались промпты для различных подходов к лемматизации для достижения конечных результатов по точности.

### 2.4.1. Лемматизация токенов без указания контекста и части речи

Для базового промпта, использующегося в предыдущем примере при сравнении точностей LLM в подходе к лемматизации без указания контекста и части речи Saiga и T-Lite был использован следующий шаблон:

“Приведи слово '{word}' к начальной форме (лемме).

В ответе напиши только одно слово - лемму, без комментариев и предложений.

Все символы по типу ')', '-', '/', а также знаки препинания сохраняй в изначальном виде.”

Помимо того, чтобы описать базовый промпт требуется реализовать шаблон чата с запросами для того, чтобы повысить точность. Шаблон чата – это шаблон общения пользователя, подающего запрос на вход к модели и шаблон общения модели, выдающей результат обработки запроса.

Первоначально было определено две роли:

* **User** – пользователь, вводящий запросы к LLM. В этой роли также были прописаны некоторые инструкции для поведения модели (например, если в слове попадается буква ‘ё’, то всегда заменяй ее на ‘е’)
* **System** – роль для выдачи ответов от LLM

Изначально, чат формируется в формате JSON, но для наглядности будет демонстрироваться примерный вид сформированного шаблона чата.

Фрагмент сформированного шаблона чата:

"user": "Лемматизируй вводимые токены. Знаки пунктуации при лемматизации не изменяются. В ответ выводи только одно слово или токен."

"user": "Все слова, начинающиеся на символ '#' не изменяй. Точку в конце токена не ставь."

"user": """Приведи слово '.' к начальной форме (лемме). В ответе напиши только одно слово - лемму, без комментариев и предложений. Все символы по типу ')', '-', '/', а также знаки препинания сохраняй в изначальном виде."""

"system": "."

"user": """Приведи слово 'мною' к начальной форме (лемме). В ответе напиши только одно слово - лемму, без комментариев и предложений. Все символы по типу ')', '-', '/', а также знаки препинания сохраняй в изначальном виде."""

“system”: “я”

…

"user": “{базовый запрос, определенный выше, с подставленным словом из корпуса}”

И к последнему запросу ответ для роли system не прописывается, т.к. это именно то, предсказание, которое ожидается от модели. Однако, вышеописанный шаблон чата уже описал поведение модели при получении типовых запросов.

В результате исследований были внесены следующие дополнительные изменения:

* Добавлены эвристики для знаков препинания и символов
* Изменены роли, добавлена третья роль – assistant. В роли system теперь хранятся инструкции перед отправкой запросов, в роли user – отправляемые запросы, а роль assistant вместо system выдает ответы от LLM

### 2.4.2. Лемматизация токенов с учётом контекста

Для базового промпта, использующегося в предыдущем примере при сравнении точностей LLM в подходе к лемматизации без указания контекста и части речи Saiga и T-Lite был использован следующий шаблон:

Приведи слово '{word}' из предложения '{full\_sentence}' к начальной форме (лемме).

В ответе напиши только одно слово - лемму, без комментариев и предложений.

Все символы по типу ')', '-', '/', а также знаки препинания сохраняй в изначальном виде.

Аналогично подходу к лемматизации выше, был составлен шаблон чата.

Фрагмент сформированного шаблона чата в наглядном виде:

"user": "Лемматизируй вводимые токены с учетом контекста. Знаки пунктуации при лемматизации не изменяются. В ответ выводи только одно слово или токен."

"user": "Все слова, начинающиеся на символ '#' не изменяй. Точку в конце токена не ставь."

"user": """Приведи слово 'мыла' из предложения 'Мама мыла раму.' к начальной форме (лемме). В ответе напиши только одно слово - лемму, без комментариев и предложений. Все символы по типу ')', '-', '/', а также знаки препинания сохраняй в изначальном виде."""

"system": "мыть"

"user": """Приведи слово 'стали' из предложения 'Они сделаны из стали.' к начальной форме (лемме).В ответе напиши только одно слово - лемму, без комментариев и предложений. Все символы по типу ')', '-', '/', а также знаки препинания сохраняй в изначальном виде."""

“system”: “я”

…

"user": “{базовый запрос, определенный выше, с подставленным словом из корпуса}”

Аналогично подходу выше, в результате исследований были внесены следующие дополнительные изменения:

* Добавлены эвристики для знаков препинания и символов
* Изменены роли, добавлена третья роль – assistant. В роли system теперь хранятся инструкции перед отправкой запросов, в роли user – отправляемые запросы, а роль assistant вместо system выдает ответы от LLM

### 2.4.3. Лемматизация токенов с учётом части речи

Для этого подхода требовалось получить наибольшее значение точности, так как у заказчика уже имеется модель, предсказывающая части речи для различных слов с точностью около 98%.

Изначально, шаблон чата для подхода с использованием частей речи был похож на шаблоны чата, описанные выше: совершенно таким же образом были внедрены три роли: user, assistant и system, также были прописаны эвристики.

Однако, для повышения точности, было решено изменить подход к формированию шаблона чата и ввести эвристики, исходя из части речи.

Некоторые части речи не являются изменяемыми: их словоформы остаются такими же, какими и были на входе

К таким частям речи относятся:

* частицы
* наречия
* местоименные наречия
* предлоги
* символы
* сочинительные и подчинительные союзы
* междометия
* предикативы
* вводные слова
* знаки препинания

Также, к эвристикам были добавлены следующие условия:

* Если вводимый токен является числительным или порядковым числительным и при этом содержит в себе только цифры
* Если длина слова не превышает одного символа (полезно учитывать, например, с частицами (“б”), с предлогами (“y”), с союзами (“а”)

Для всех частей речи, не входящих в эвристики были составлены индивидуальные шаблоны чата.

Примеры для некоторых частей речи:

Существительное:

"system": "Лемматизируй вводимые токены с учетом контекста. Знаки пунктуации при лемматизации не изменяются. В ответ выводи только одно слово или токен."

"system": "Точку в конце токена не ставь. Букву 'ё' во всех словах заменяй на 'е'

"user": "Приведи существительное 'собаку' к именительному падежу, единственному числу.”

"assistant": "собака"

"user": "Приведи существительное 'мышью' к именительному падежу, единственному числу. “

“assistant”: “мышь”

…

"user": “{базовый запрос, определенный выше, с подставленным словом из корпуса}”

Глагол:

"system": "Лемматизируй вводимые токены с учетом контекста. Знаки пунктуации при лемматизации не изменяются. В ответ выводи только одно слово или токен."

"system": "Точку в конце токена не ставь. Букву 'ё' во всех словах заменяй на 'е'

"user": "Приведи глагол 'играли' к инфинитиву, сохранив его вид (совершенный или несовершенный), так чтобы он отвечал на вопрос 'что делать?' или 'что сделать?’

"assistant": "играть"

"user": "Приведи глагол 'ешь' к инфинитиву, сохранив его вид (совершенный или несовершенный), так чтобы он отвечал на вопрос 'что делать?' или 'что сделать?'“

“assistant”: “есть”

…

"user": “{базовый запрос, определенный выше, с подставленным словом из корпуса}”

При лемматизации токенов при подаче CoNLL-U файла используется данный подход к лемматизации с учетом части речи.

### 2.4.4. Результаты замеров точности после подбора шаблонов запросов

Итоговые результаты показателей Accuracy при различных подходах к лемматизации после подбора шаблонов запросов представлены в таблице 6. Лучшее значение метрики выделено полужирным шрифтом.

*Таблица 6. Показатели точности предсказания лемм большой языковой моделью Saiga-LLaMa-3-8b при разных подходах к лемматизации после завершения подбора промптов*

|  |  |
| --- | --- |
| **Подход к лемматизации** | **Точность (Accuracy) Saiga-LLaMa-3-8b** |
| Лемматизация без учета контекста и части речи | 85,88% |
| Лемматизация с учетом контекста | 80,39% |
| Лемматизация с учетом части речи | **91,1%** |

В дальнейшем планируется провести fine-tuning модели для повышения точности предсказаний.

# Глава III. Проектирование и реализация API и WEB-сервисов для лемматизации

## 3.1. Требования к реализуемому проекту

**Требования заказчика:**

* Исследовать текущие решения для задачи лемматизации; исследовать возможность использования больших языковых моделей, основанных на управлении инструкциями для задачи лемматизации русскоязычных текстов социальных сетей;
* Сравнить текущие решения для задачи лемматизации с большими языковыми моделями в рамках задачи лемматизации русскоязычных текстов социальных сетей;
* Реализовать различные подходы к лемматизации текстов: лемматизация токенов без указания контекста и части речи; лемматизация токенов с указанием контекста; лемматизация токенов с указанием части речи; лемматизация токенов в CoNLL-U файлах с использованием подхода лемматизации с указанием части речи;
* Путем подбора шаблонов запроса к большой языковой модели получить максимально возможную точность предсказания лемм;
* Создать сервис для лемматизации русскоязычных текстов социальных сетей (API, WEB-приложение и Telegram Mini App) с использованием выбранной большой языковой модели;

**Требования к функционалу системы:**

1) Регистрация и авторизация:

Пользователи (разметчики Национального корпуса русского языка) могут зарегистрироваться и войти в систему.

Результат: пользователи могут создавать учетные записи и безопасно входить в случае работы с WEB-приложением, а также использовать уже существующую учетную запись в Telegram. Это обеспечивает доступ к истории генерации лемм, а также возможность просматривать историю с разных платформ (WEB-приложение либо Telegram) путем синхронизации данных с аккаунтов на разных платформах.

2) Генерация лемм:

Пользователи могут генерировать леммы с помощью различных подходов.

Результат: возможность генерировать леммы четырьмя способами: путем передачи одного слова без контекста и части речи; передавая предложение и генерируя лемму для каждого слова в этом предложении; генерируя лемму для конкретного слова в предложении с использованием предложения в качестве контекста; передавая часть речи вместе со словом, начальную форму которого нужно предсказать.

## 3.2. Используемые технологии

### 3.2.1. Языки программирования

Основной язык программирования, на котором в проекте исследуются современные решения по лемматизации, большие языковые модели, реализуется API и backend-часть WEB-приложения и Telegram Mini App – это язык программирования **Python**.

**Python** — мультипарадигмальный высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

Одной из фишек этого языка программирования является огромное кол-во библиотек - наборов часто используемых кодов, которые разработчики могут включать в свои программы Python, чтобы не писать код с нуля, а также фреймворков - наборов модулей или пакетов, которые предоставляют готовые компоненты и решения для создания приложений любого типа.

В текущем проекте, помимо описанных выше **PyMorphy2, PyMystem3** и **Stanza** используются следующие основные фреймворки и библиотеки Python:

* **PyTorch** - фреймворк машинного обучения для языка Python с открытым исходным кодом
* **Transformers** – библиотека, представляющая инструменты и интерфейсы для загрузки и использования предобученных Deep Learning моделей с платформы Hugging Face
* **Conllu** – библиотека для работы с данными формата CoNLL-U
* **FastAPI** - веб-фреймворк для создания API, в основе которого лежит стандартная аннотация типов Python
* **Uvicorn** - библиотека, позволяющая запустить свой собственный веб-сервер
* **Pydantic** – библиотека, предназначенная для валидации и трансформации данных

Также, помимо Python, для разработки frontend-частей WEB-приложения и Telegram Mini App в проекте используется язык программирования **Javascript** с надстройкой в виде библиотеки **React.**

**JavaScript** (JS) — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам. Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

**React** (React.js или ReactJS) — библиотека для языка программирования JavsScript с открытым исходным кодом для разработки пользовательских интерфейсов.

### 3.2.2. СУБД

**Система управления базами данных (СУБД) —** совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных (БД).

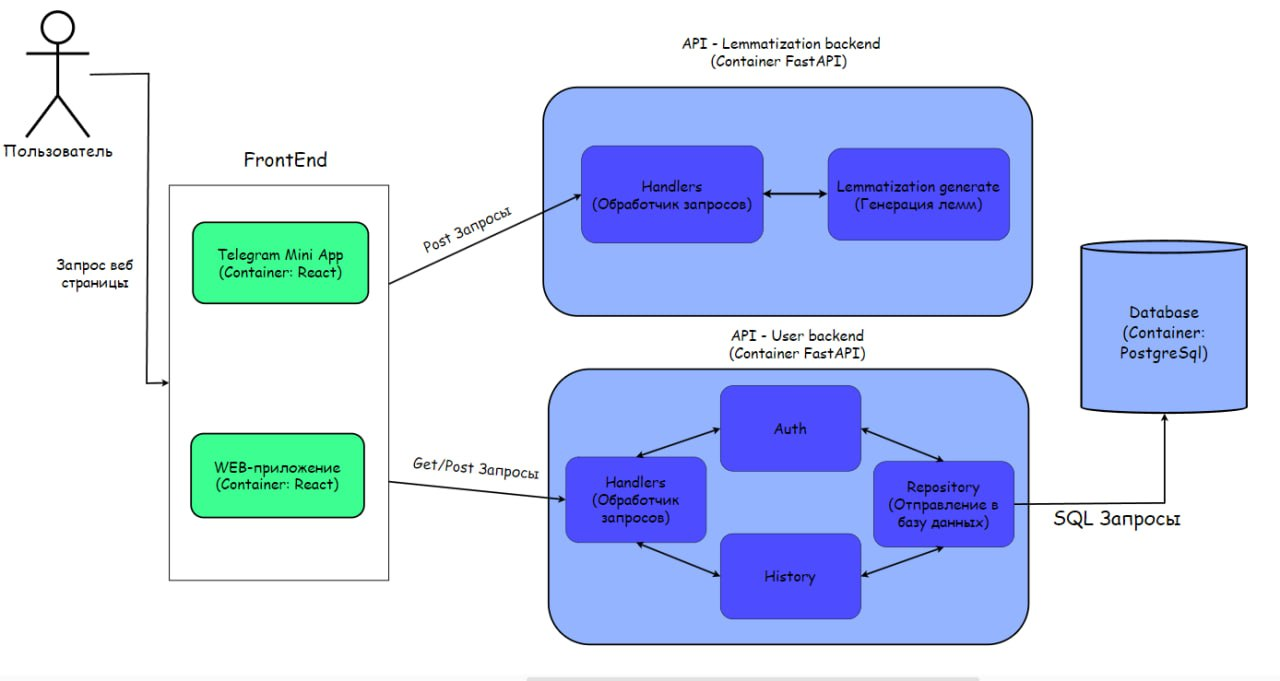
В нашем проекте в качестве СУБД используется PostgreSQL.

**PostgreSQL -** свободная объектно-реляционная система управления базами данных, базируется на языке SQL.

База данных в проекте нужна для хранения истории запросов, а также данных аккаунта пользователя.

## 3.3. Функциональная схема сервиса

Функциональная схема сервиса представлена на рисунке 5.



*Рис. 5. Функциональная схема сервиса*

## 3.4. Разработка API для лемматизации с использованием LLM

Разработка API для лемматизации является основным этапом в разработке веб приложения для лемматизации. В качестве языка программирования был выбран Python, что обусловлено наличием библиотек для работы с нейронными сетями, таких как Hugging Face Transformers и Pytorch. Эти библиотеки предоставляют доступ к предобученным моделям, которые могут выполнять задачи лемматизации.

API состоит из двух методов POST и имеет следующую структуру:

1) Лемматизация текста

POST <https://lemmaapp.ru/lemmatize>

Тело запроса принимает JSON-формат:

json

{

"mode": int,

"selected\_option": str,

"input\_value": str

}

Параметр mode определяет вариант лемматизации и может принимать два значения:

* Лемматизация одного слова (токена): в этом режиме API принимает одно слово и возвращает его лемму в зависимости от части речи.
* Лемматизация предложения: в этом режиме пользователь может выбрать одно слово из предложения или указать все предложение для лемматизации.
* Лемматизация переданного CoNLL-U файла: в этом режиме API принимает корпус формата .conllu и лемматизирует все токены, находящиеся в корпусе.

Параметр selected\_option определяет дополнительные параметры лемматизации, которые зависят от выбранного режима:

* При реализации первого режима (лемматизация одного слова) API обрабатывает слово с использованием промта, в зависимости от указанной частью речи в selected\_option.
* При реализации второго режима (лемматизация предложения) API обрабатывает все слова из предложения по порядку, используя предложение как контекст или обрабатывает одно слово, в зависимости от указанного параметра в selected\_option

Параметр input\_value передает текст, вводимый пользователем, который необходимо лемматизировать.

2) Лемматизация conllu файла

POST <https://lemmaapp.ru/process_conllu>

Тело запроса принимает JSON-формат:

json

{

"file": file

}

Данный запрос принимает на вход conllu файл и возвраващает обработанный conllu файл с леммами

## 3.5. Разработка backend-части для WEB-приложения и Telegram Mini App

Backend-часть приложения была разработана с использованием библиотеки FastAPI. Для создания RESTful API с целью поддержки асинхронного программирования, что позволяет эффективно обрабатывать множество одновременных запросов без задержек.

Структура проекта включает следующие основные компоненты:

* Модели данных: описывают структуру данных, используемых в приложении, включая модели для хранения истории запросов и информации о пользователях. Модели реализованы с использованием SQLAlchemy, что обеспечивает удобный интерфейс для работы с реляционными базами данных.
* Маршруты: определяют конечные точки API и связывают их с соответствующими обработчиками запросов. Каждый маршрут реализует определенную функциональность, обеспечивая четкую структуру кода и легкость в поддержке.
* Системы аутентификации и авторизации: обеспечивают безопасность пользовательских данных и управление доступом к API. Для этого использованы механизмы JWT (JSON Web Tokens), что позволяет реализовать безопасную аутентификацию пользователей.

Для интеграции с Telegram Mini App был разработан специальный обработчик, который принимает сообщения от пользователей через Telegram API и отправляет запросы к нашему API для получения результатов лемматизации. Это обеспечивает удобное взаимодействие пользователей с приложением через популярную платформу мессенджеров.

## 3.6. Проектирование и разработка базы данных для WEB-сервисов

Для реализации регистрации и авторизации пользователей и хранения истории запросов и ответов в СУБД PostgreSQL была спроектирована следующая база данных, схема которой представлена рисунке 6:

Таблица “User” (“Пользователь”) – хранит информацию о пользователе сервиса (Таблица 7).

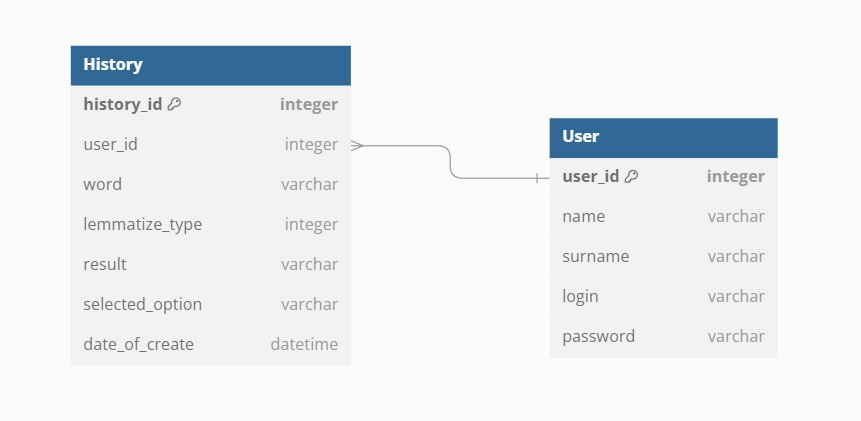
*Таблица 7. Таблица “User” (“Пользователь”), хранящая информацию о пользователе сервиса*

|  |  |
| --- | --- |
| **Название поля** | **Значение, содержащееся в поле** |
| user\_id | Первичный ключ, идентификатор таблицы |
| name | Имя пользователя |
| surname | Фамилия пользователя |
| login | Логин |
| password | Пароль |

Таблица “History” (“История”) - хранит информацию об истории запросов по лемматизации (Таблица 8).

*Таблица 8. Таблица “History” (“История”), хранящая информацию об истории запросов*

|  |  |
| --- | --- |
| **Название поля** | **Значение, содержащееся в поле** |
| history\_id | Первичный ключ, идентификатор таблицы |
| user\_id | Идентификатор таблицы пользователя, внешний ключ |
| word | Текст, отправленный пользователем для лемматизации |
| lemmatize\_type | Тип лемматизации выбранный пользователем |
| result | Лемматизированный текст |
| selected\_option | Выбранный параметр для лемматизации |
| date\_of\_create | Дата создания запроса |

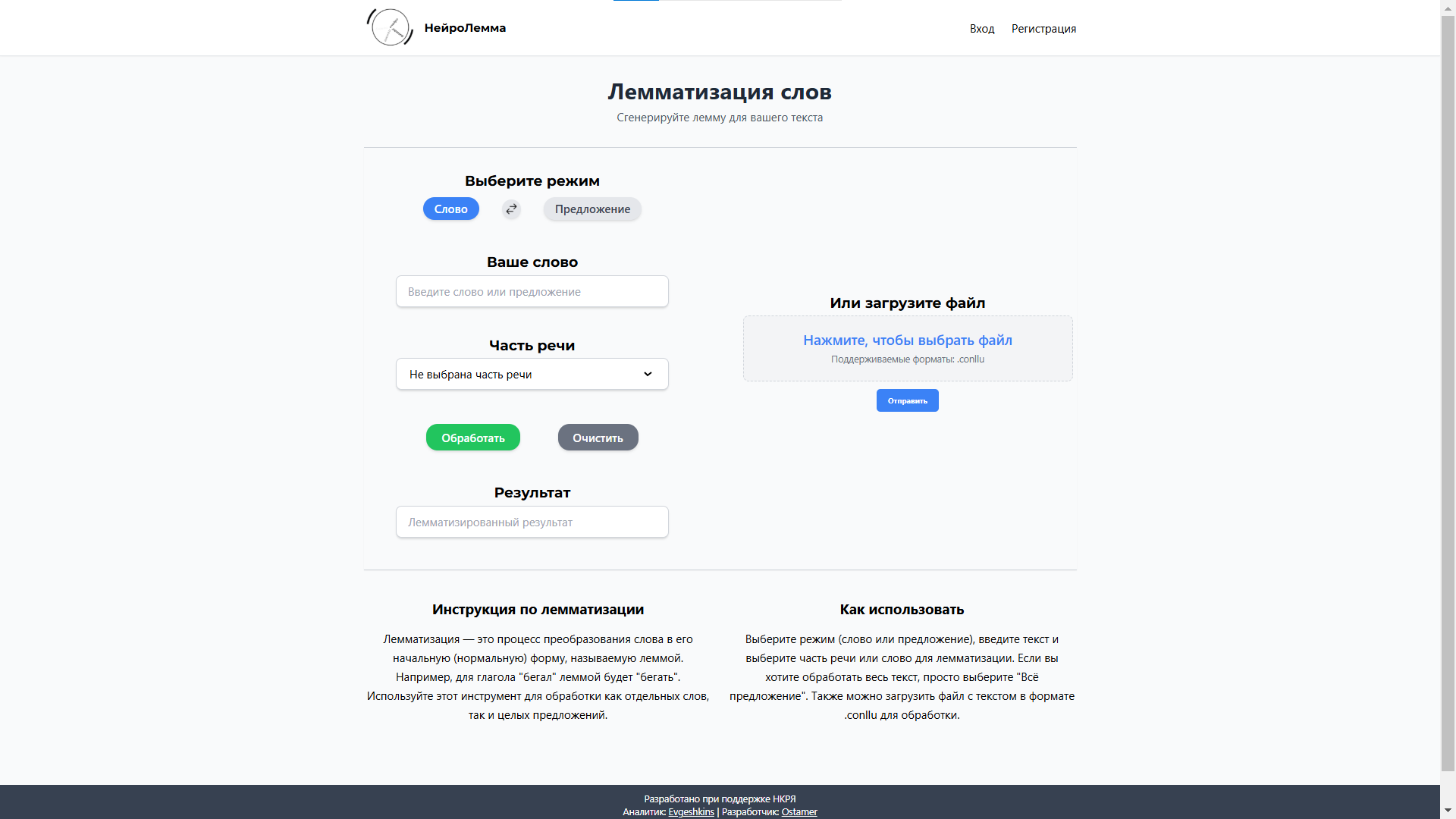


*Рис. 6. Схема базы данных*

## 3.7. Проектирование и реализация интерфейсов WEB-приложения и Telegram Mini App

Интерфейс WEB-приложения был разработан с использованием библиотеки React, что обеспечило создание современного и отзывчивого дизайна. React позволяет эффективно управлять состоянием приложения и обновлять интерфейс без перезагрузки страницы.

Вид интерфейса WEB-приложения представлен на рисунке 7.

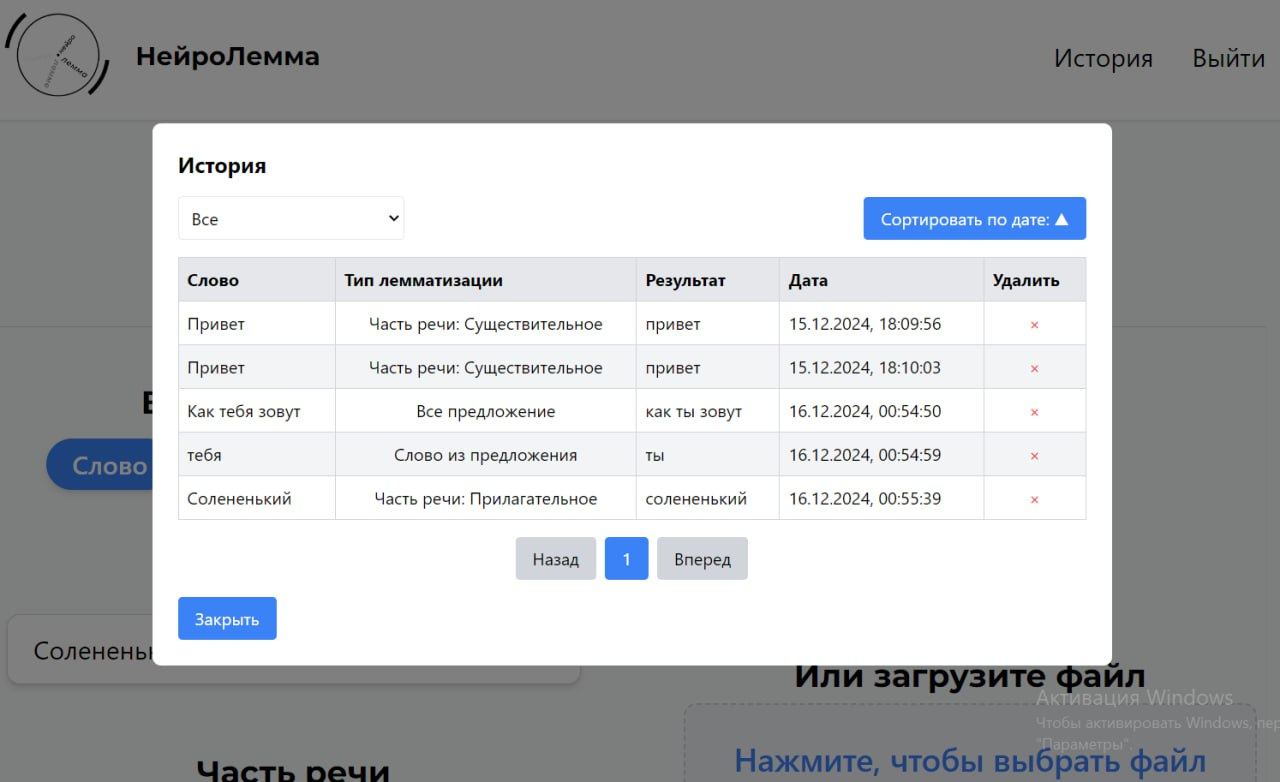


*Рис. 7. Интерфейс WEB-приложения*

Основные компоненты WEB-интерфейса включают:

* Шапка: предоставляет пользователю возможность авторизироваться/зарегистрироваться чтобы сохранять результаты лемматизации в базу данных.
* Основная форма: предоставляет пользователям возможность вводить текст для лемматизации, выбирать режим обработки и отображает результат лемматизации.
* История: предоставляет пользователям доступ к предыдущим запросам и их результатам, фильтрациям по способам лемматизации а также сортировке по дате запроса.
* Подсказки: описание того, как пользоваться приложением по лемматизации

Вид интерфейса истории WEB-приложения представлен на рисунке 8.



*Рис. 8. Интерфейс истории WEB-приложения*

Telegram Mini App был реализован аналогично веб-приложению с учетом особенностей платформы Telegram.

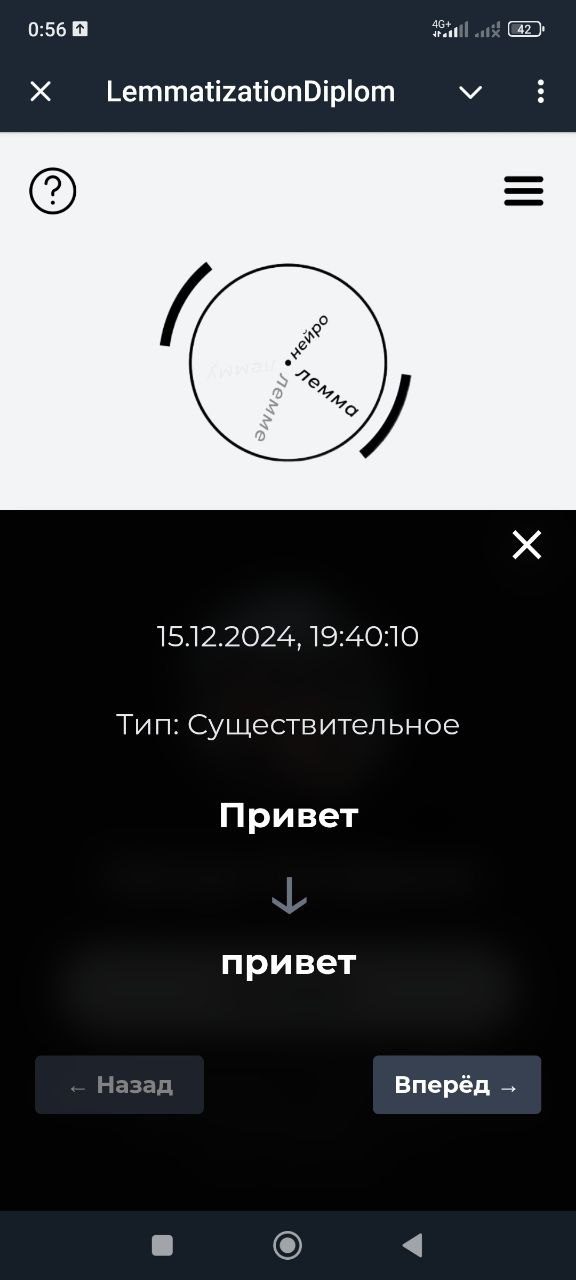
Вид интерфейса Telegram Mini App представлен на рисунке 9.



*Рис. 9. Интерфейс Telegram Mini App*

У интерфейса истории Telegram Mini App упрощенный функционал без фильтрации, сортировки и возможности удалить запись.

Вид интерфейса истории Telegram Mini App представлен на рисунке 10.



*Рис. 10. Интерфейс истории Telegram Mini App*

## 3.8. Сборка и развертывание сервиса

После завершения разработки frontend и backend частей WEB-приложения была выполнена сборка проекта.

Развертывание (Deploy) backend-частей был осуществлен на удаленном арендованном сервере хостинга Selectel с использованием Nginx для проксирования запросов. Nginx был настроен для обработки HTTPS-запросов с установленными SSL-сертификатами и собственным доменом, что обеспечивает безопасность передачи данных между клиентами и сервером. База данных была развернута на том же сервере с использованием Docker-контейнеров для упрощения управления окружением и обеспечения изоляции сервисов.

Frontend-часть WEB-приложения была развернута на удаленном арендованном сервере хостинга Timeweb.

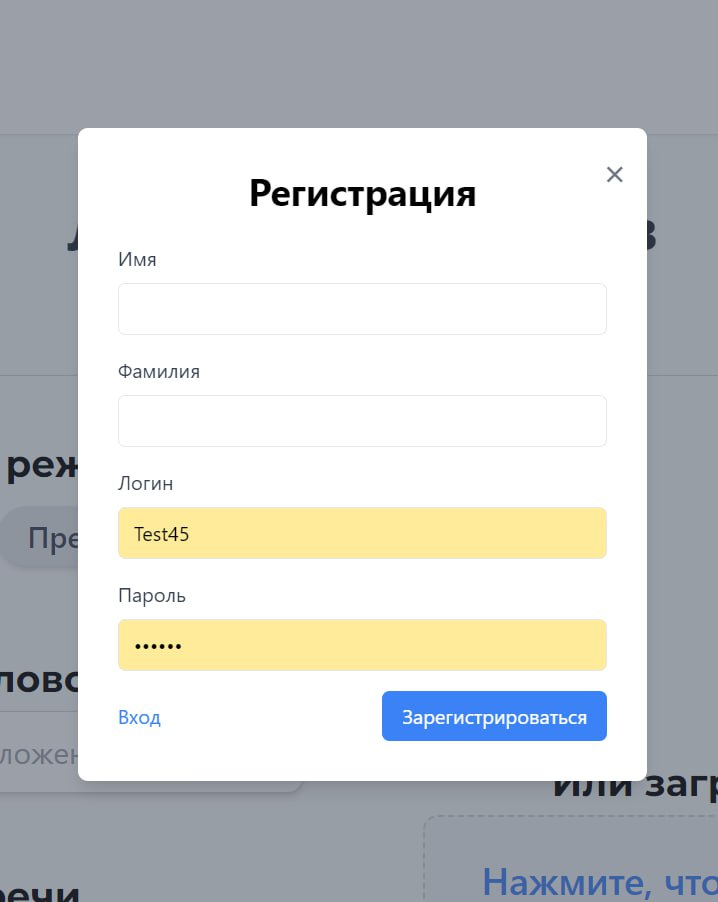
Для запуска Telegram Mini App был реализован телеграм-бот.

# Глава IV. Функциональные возможности WEB-сервисов

## 4.1. Регистрация и авторизация пользователей

В WEB-приложении для авторизации и регистрации пользователей в верхней части сайта предусмотрены соответствующие кнопки с возможностью раскрытия модальных окон.  
 При переходе на окно регистрации пользователю открывается модальное окно, в котором необходимо ввести логин, пароль, имя, фамилию для создания аккаунта. После успешного создания аккаунта пользователю предлагается пройти авторизацию для входа в аккаунт.

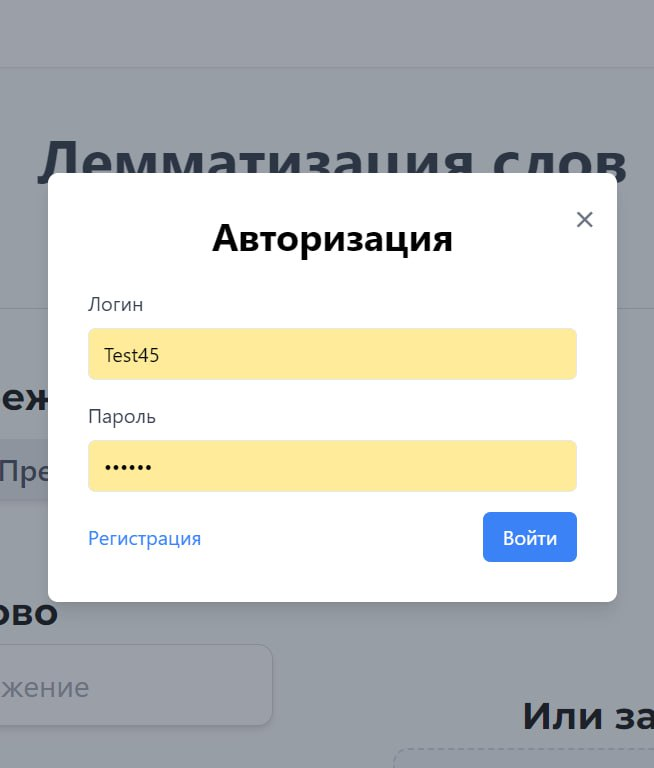
Вид модального окна регистрации WEB-приложения представлен на рисунке 11.



*Рис. 11. Интерфейс модального окна регистрации WEB-приложения*

При переходе на окно авторизации пользователю открывается модальное окно, в котором необходимо ввести логин и пароль ранее созданного аккаунта. После успешного входа в аккаунт у пользователя появляется кнопка истории на месте кнопок входа и регистрации и возможность сохранения результатов проведенной лемматизации.

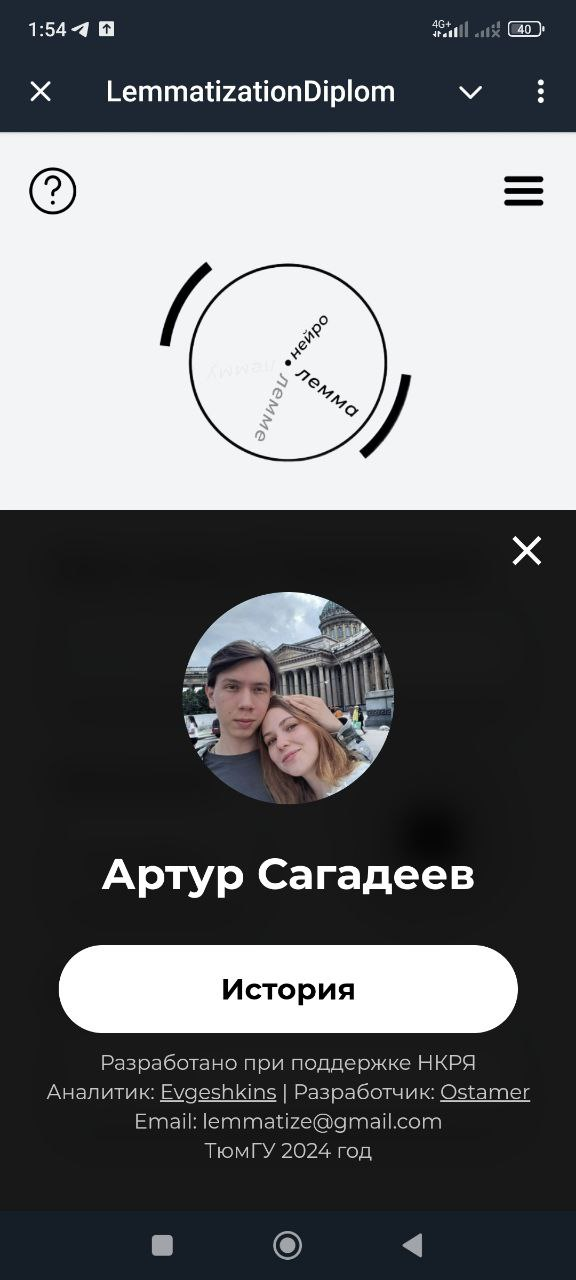
Вид модального окна регистрации WEB-приложения представлен на рисунке 12.



*Рис. 12. Интерфейс модального окна авторизации WEB-приложения*

В Telegram Mini App авторизация происходит при запуске приложения по telegram\_id:

Вид главного меню Telegram Mini App представлен на рисунке 13.



*Рис. 13. Интерфейс главного меню Telegram mini app*

## 4.2. Выбор варианта лемматизации

В WEB-приложении для лемматизации существует 5 возможных вариантов:

1) Лемматизация слова без указания части речи

2) Лемматизация слова с указанием части речи

3) Лемматизация каждого слова предложения

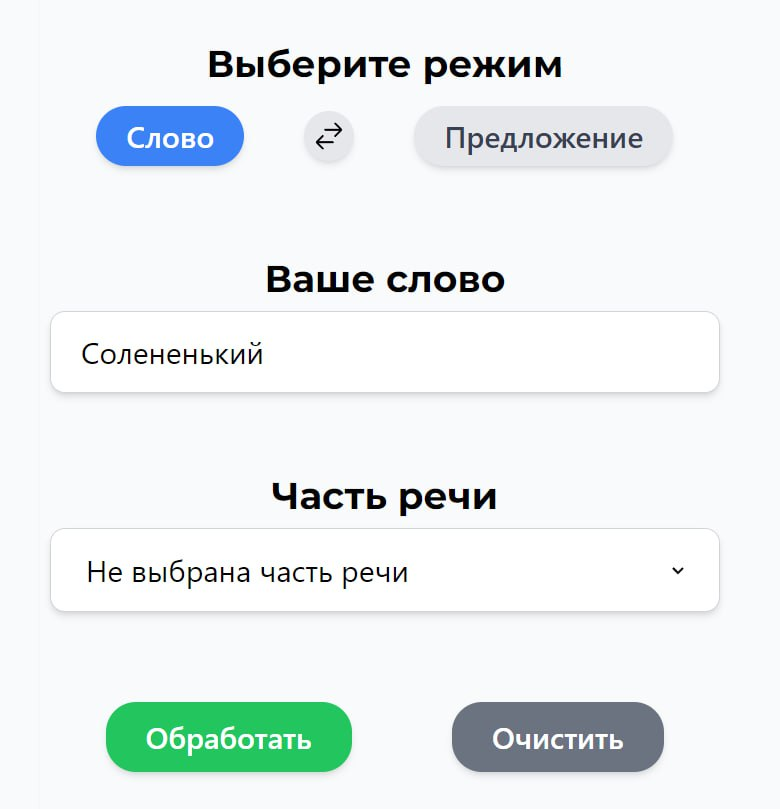
4) Лемматизация конкретного слова из предложения

5) Лемматизация CoNLL-U файла

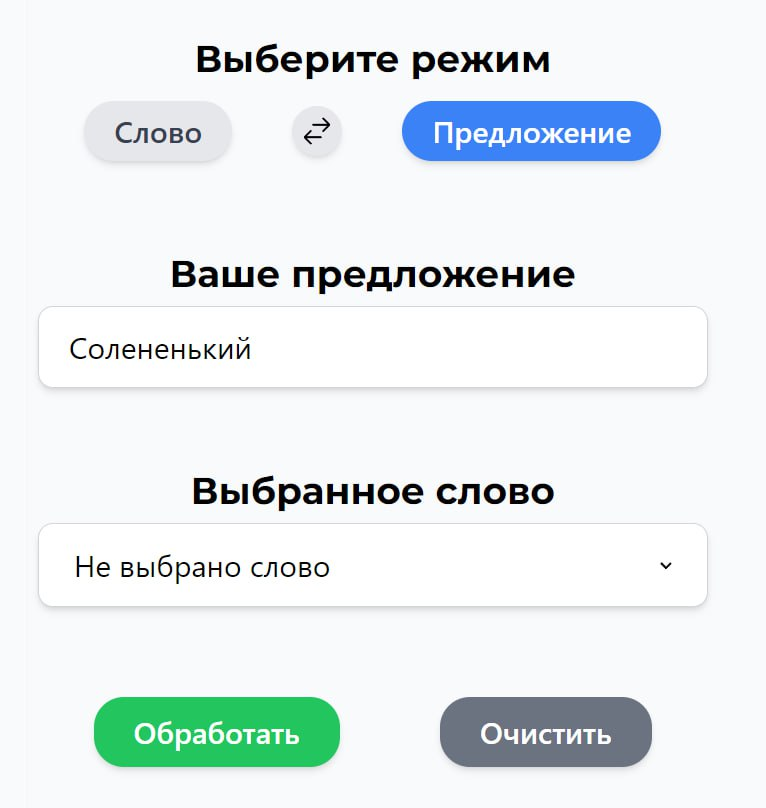
Способ лемматизации с указанием части речи и без указания части речи в Web-приложении представлен на рисунке 14.

Способ лемматизации каждого слова предложения и конкретного слова предложения в Web-приложении представлен на рисунке 15.

Способ лемматизации conllu файла в Web-приложении представлен на рисунке 16.

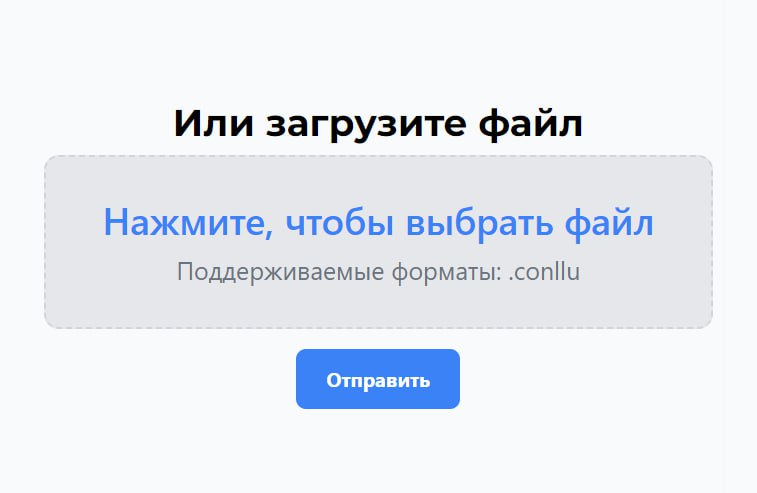


*Рис. 14. Способ лемматизации с указанием части речи и без указания части речи в Web-приложении*



*Рис. 15. Способ лемматизации каждого слова предложения*

*и конкретного слова предложения в Web-приложении*



*Рис. 16. Способ лемматизации conllu файла в Web-приложении*

В WEB-приложении для лемматизации существует 4 возможных варианта:

1) Лемматизация слова без указания части речи

2) Лемматизация слова с указанием части речи

3) Лемматизация каждого слова предложения

4) Лемматизация конкретного слова из предложения

Способ лемматизации с указанием части речи и без указания части речи в Telegram Mini App представлен на рисунке 17.

Способ лемматизации каждого слова предложения и конкретного слова предложения в Telegram Mini App представлен на рисунке 18.



*Рис. 17. Способ лемматизации с указанием части речи и без указания части речи в Telegram Mini App*



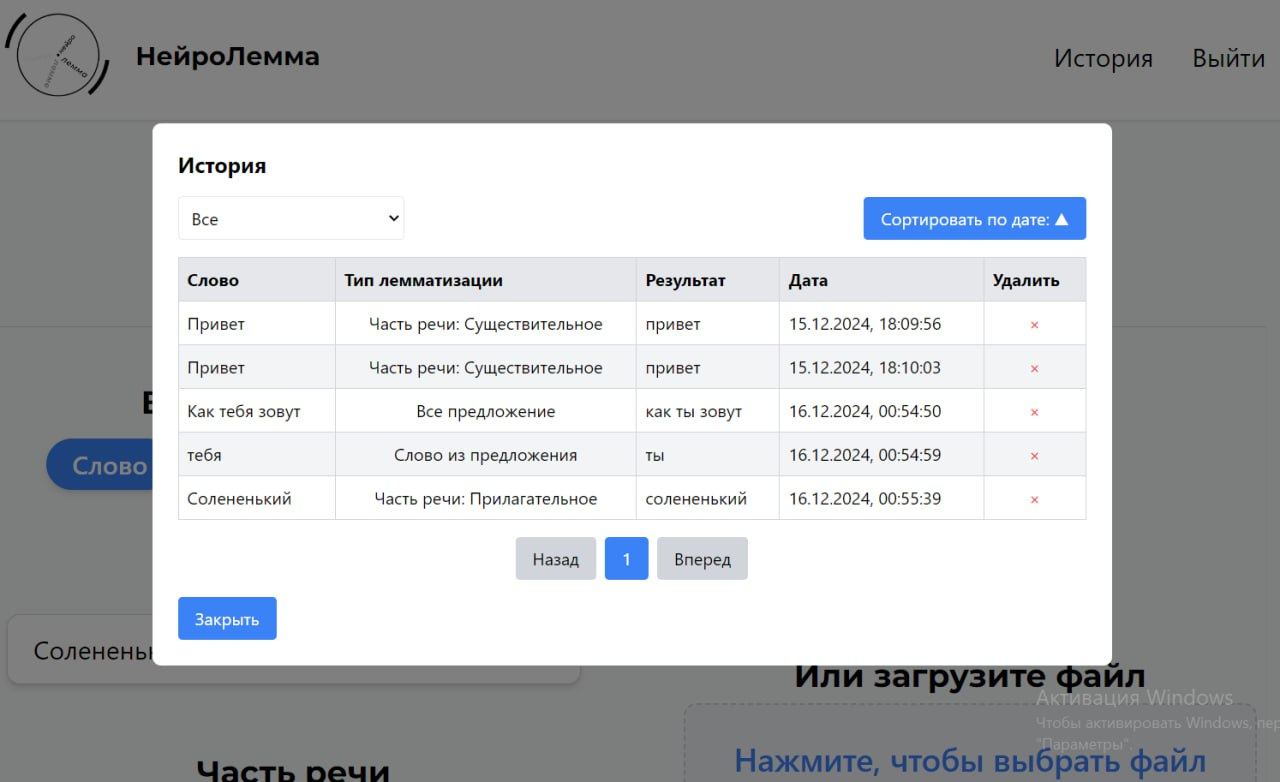
*Рис. 18. Способ лемматизации каждого слова предложения и конкретного слова предложения в Telegram Mini App*

## 4.3. Хранение истории запросов

В Telegram Mini App и Web-приложении реализовано хранение истории запросов при условии что пользователь авторизирован.

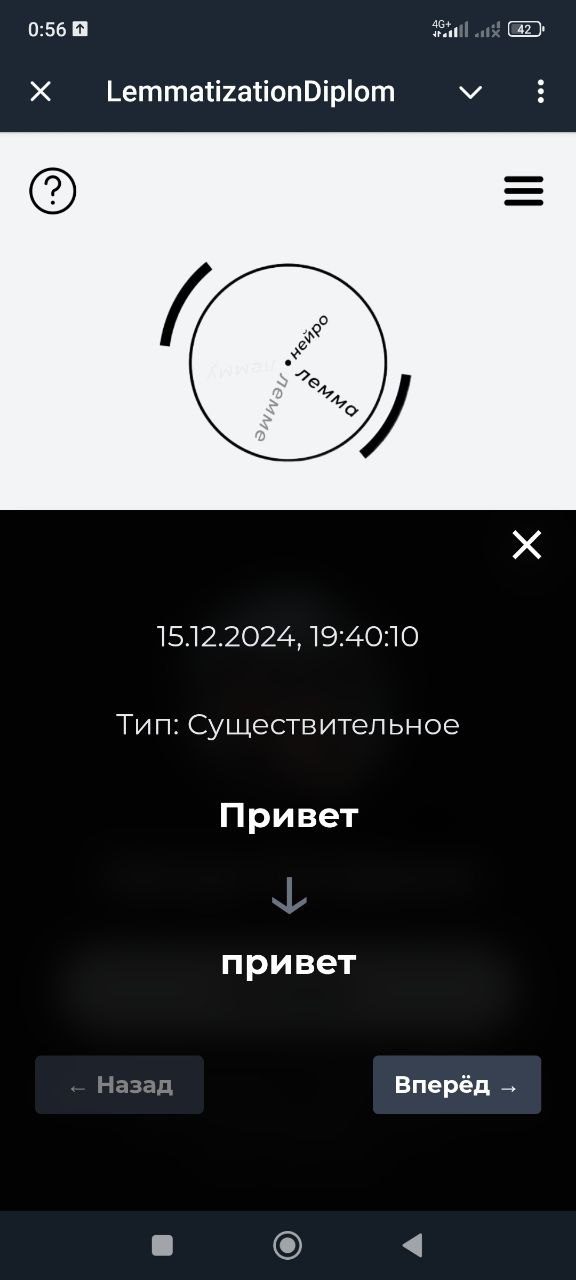
В Web-приложении, функционал истории выглядит следующим образом: просмотр истории лемматизаций, пагинация по 5 записей на странице, фильтрация по типу лемматизации, сортировка по дате запроса и возможность удаления записи из истории.

Вид интерфейса истории в Web-приложении представлен на рисунке 19.

 *Рис. 19. Интерфейс истории в Web-приложении*

В Telegram Mini App, функционал истории выглядит следующим образом: просмотр истории лемматизаций по 1 запросу на странице.

Вид интерфейса истории в Telegram Mini App представлен на рисунке 20:



*Рис. 20. Интерфейс истории в Telegram Mini App*

# Глава V. Тестирование.

Для процедуры проведения тестирования сервиса составлен график проведения тестирования (Таблица 9).

*Таблица 9. График проведения тестирования сервиса*

|  |  |  |
| --- | --- | --- |
| **Тестируемый сервис** | **Кто тестирует** | **Диапазон тестирования (даты)** |
| Telegram Mini App | Аналитик (Копырин Евгений) и разработчик (Сагадеев Артур) | 05.12-07.12 |
| WEB-приложение (версия для ПК) | Аналитик (Копырин Евгений) и разработчик (Сагадеев Артур) | 08.12-10.12 |
| WEB-приложение (мобильная версия) | Аналитик (Копырин Евгений) и разработчик (Сагадеев Артур) | 10.12-11.12 |
| Telegram Mini App, WEB-приложение (версия для ПК и мобильная версия) | Сотрудники Национального корпуса русского языка | 12.12-15.12 |

Проблемы, выявленные аналитиком и разработчиком проекта при тестировании сервиса представлены в таблице 10.

*Таблица 10. Проблемы, обнаруженные аналитиком и разработчиком сервиса при тестировании сервиса*

|  |  |
| --- | --- |
| **Обнаруженные проблемы** | **Рекомендации** |
| Галлюцинации большой языковой модели | Уменьшить значение параметра "temperature" для модели; попробовать guidance; перегенерировать предсказание |
| В истории запросов к сервису лемматизации не отображается выбранный подход к лемматизации | Добавить отображение выбранного пользователем подхода к лемматизации (в процессе решения) |
| Не реализована адаптивность интерфейса WEB-приложения под экраны с большим разрешением (например, 4К - 3840x2160) | Адаптировать верстку интерфейса под экраны с большим разрешением |

Проблемы, выявленные сотрудниками Национального корпуса русского языка при тестировании сервиса представлены в таблице 11.

*Таблица 11. Проблемы, обнаруженные сотрудниками Национального корпуса русского языка сервиса при тестировании сервиса*

|  |  |
| --- | --- |
| **Обнаруженные проблемы** | **Рекомендации** |
| В разделе “история” не сохраняется история запросов | Починить функционал сохранения истории запросов |
| В поле ввода можно вводить только пробелы, писать английские буквы, хотя работа ведется только с русскоязычными текстами | Добавить валидацию поля ввода |

# СПИСОК ЛИТЕРАТУРЫ

* 1. Korobov M. Morphological analyzer and generator for Russian and Ukrainian languages //Analysis of Images, Social Networks and Texts: 4th International Conference, AIST 2015, Yekaterinburg, Russia, April 9–11, 2015, Revised Selected Papers 4. – Springer International Publishing, 2015. – С. 320-332.
  2. Segalovich I. A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine //Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications. – 2003. – С. 273-280.
  3. Qi P. et al. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages //Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations. – 2020. – С. 101-108.
  4. Wolf T. et al. Transformers: State-of-the-art natural language processing //Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations. – 2020. – С. 38-45.
  5. Radford A. et al. Language models are unsupervised multitask learners //OpenAI blog. – 2019. – Т. 1. – №. 8. – С. 9.
  6. Devlin J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding //Proceedings of naacL-HLT. – 2019. – Т. 1. – С. 2.
  7. Raffel C. et al. Exploring the limits of transfer learning with a unified text-to-text transformer //Journal of machine learning research. – 2020. – Т. 21. – №. 140. – С. 1-67.
  8. Lyashevskaya O. N. et al. GramEval 2020 shared task: Russian full morphology and universal dependencies parsing //Proc. of the International Conference Dialogue. – 2020. – Т. 2020. – С. 553-569.