Министерство образования Московской области

Государственное бюджетное профессиональное образовательное учреждение Московской области «Физико-технический колледж»

ЗАЩИЩЕНО

ОЦЕНКА

Руководитель специальности

09.02.07 Информационные

системы и программирование

Метод автоматического определения и исправления орфографических и пунктуационных ошибок в тексте

Пояснительная записка к курсовому проекту

# по МДК.05.02 Метод автоматического определения и исправления орфографических и пунктуационных ошибок в тексте

Руководитель КП Разработал

преподаватель студент группы ИСП 4-5 ГБПОУ МО «Физтех-колледж»

Г.В. Базяк

А.А. Козин

Долгопрудный, 2024

**Министерство образования Московской области Государственное бюджетное профессиональное образовательное учреждение**

## Московской области «Физико-технический колледж»

|  |  |
| --- | --- |
| Специальность 09.02.07 Информационные системы и программирование  Квалификация «Программист» | **УТВЕРЖДЕНО**  цикловой комиссией специальности 09.02.07  « » 2024 года Руководитель специальности |

ЗАДАНИЕ

Для курсового проектирования по МДК 05.02 «Разработка и администрирование систем искусственного интеллекта» студенту 4 курса группы ИСП 4-5

|  |  |
| --- | --- |
|  | Козину Александру Алексеевичу |
|  | (фамилия, имя, отчество) |
| Тема задания: | Метод автоматического определения и исправления орфографических и пунктуационных ошибок в тексте |

При выполнении курсового проекта на указанную тему должны быть представлены I Пояснительная записка

|  |
| --- |
| Введение |
| 1 Анализ предметной области и описание технологии проектирования ИС |
| 1.1 Описание предметной области (проблемы проекта) |
| 1.2 Определение целевых групп пользователей |
| 1.3 Определение ограничений проектного решения |
| 1.4 Анализ предметной области  2 Функциональные требования к системе  2.1 Определение требований к эргономике  2.2 Определение требований к надежности системы  2.3 Ввод текста  2.4 Обнаружение и исправление ошибок  2.5 Выделение ошибок  2.6 Интерфейс пользователя  2.7 Ограничение по языку проверки  3 Описание этапов разработки системы искусственного интеллекта  3.1 Описание исходных данных |
| 3.2 Сравнение алгоритмов машинного обучения  3.3 Описание структуры приложения  3.4 Разработка макетов страниц  3.5 Описание дизайна приложения  3.6 Описание приемов верстки  3.7 Анализ и описание технологий программирования  3.8 Выбор домена и хостинга  3.9 Описание этапов тестирования программного решения  4 Руководство пользователя  5 Руководство администратора  3.6 Интерфейс пользователя  3.7 Многоязычная поддержка |
|  |

Заключение

Список используемых источников

Приложение A.

Приложение B

Дата выдачи: « » 2024г. Срок окончания: « » 2024 г

Руководитель курсового проектирования Г. В. Базяк

Содержание

[Введение 4](#_Toc184238303)

[1 Анализ предметной области и описание технологии проектирования ИС 8](#_Toc184238304)

[1.1 Описание предметной области (проблемы проекта) 8](#_Toc184238305)

[1.2 Определение целевых групп пользователей 11](#_Toc184238306)

[1.3 Определение ограничений проектного решения 12](#_Toc184238307)

[1.4 Анализ предметной области 12](#_Toc184238308)

[2 Функциональные требования к системе 15](#_Toc184238309)

[2.1 Определение требований к эргономике 15](#_Toc184238310)

[2.2 Определение требований к надежности системы 16](#_Toc184238311)

[2.3 Ввод текста 17](#_Toc184238312)

[2.4 Обнаружение и исправление ошибок 18](#_Toc184238313)

[2.5 Выделение ошибок 19](#_Toc184238314)

[2.6 Интерфейс пользователя 20](#_Toc184238315)

[2.7 Ограничение по языку проверки 20](#_Toc184238316)

[3 Описание этапов разработки системы искусственного интеллекта 20](#_Toc184238317)

[3.1 Описание исходных данных 20](#_Toc184238318)

[3.2 Сравнение алгоритмов машинного обучения 21](#_Toc184238319)

[3.3 Описание структуры приложения 21](#_Toc184238320)

[3.4 Разработка макетов страниц 23](#_Toc184238321)

[3.5 Описание дизайна приложения 24](#_Toc184238322)

[3.6 Описание приемов верстки 25](#_Toc184238323)

[3.7 Анализ и описание технологий программирования 26](#_Toc184238324)

[3.8 Выбор домена и хостинга 29](#_Toc184238325)

[3.9 Описание этапов тестирования программного решения 29](#_Toc184238326)

[4. Руководство пользователя 29](#_Toc184238327)

[5. Руководство администратора 30](#_Toc184238328)

[Заключение 31](#_Toc184238329)

[Список используемых источников 33](#_Toc184238330)

[Приложение А (Обязательное) 35](#_Toc184238331)

[Приложение Б (Обязательное) 36](#_Toc184238332)

# Введение

В современном мире, с его стремительным развитием технологий и увеличением объёмов текстовой информации, задача обеспечения качества текстов становится одной из важнейших. Текстовый контент окружает нас повсюду: новости, статьи, научные работы, корпоративные документы и пользовательские комментарии — всё это требует качественного подхода к обработке и проверке, особенно в таких областях, как журналистика, наука, образование и бизнес. Ошибки, будь то орфографические, пунктуационные или грамматические, не только снижают читабельность, но и могут серьёзно повлиять на восприятие профессионального уровня документа, подрывая доверие и репутацию его автора или организации. На протяжении многих лет для проверки текстов применялись традиционные методы, основанные на фиксированных правилах и словарях, которые, несмотря на свою популярность, всё чаще оказываются недостаточными. Статические алгоритмы проверки правописания, использующие простые правила, не способны адаптироваться к постоянно меняющемуся языку и учитывать сложные грамматические и контекстные особенности текста. Такие системы, как правило, ограничены стандартными словарями, не включают новейшие термины и не способны выявить ошибки, которые зависят от контекста, из-за чего результат работы может быть далёк от идеала. В ответ на это ограничение всё большую популярность приобретают методы машинного обучения. Они позволяют разрабатывать гибкие системы, способные учитывать контекст, обучаться на больших объемах данных и выявлять ошибки с высокой степенью точности. Машинное обучение в обработке естественного языка (NLP) стало революционным направлением, позволяющим анализировать и корректировать тексты с учётом их семантических и синтаксических особенностей. В отличие от правил, жёстко зафиксированных в традиционных алгоритмах, модели машинного обучения адаптируются к новым данным и могут постоянно улучшаться, что позволяет им выявлять ошибки, недоступные для статических методов.

Целью данного проекта является создание системы автоматического обнаружения и исправления ошибок с использованием модели машинного обучения T5, которая способна учитывать контекст и сложные языковые конструкции, что делает её подходящей для исправления орфографических и пунктуационных ошибок. Основным мотивом разработки является решение проблемы, с которой я неоднократно сталкивался: существующие сайты для проверки текстов часто оказываются малоэффективными и неспособными предложить качественную корректировку. Они не учитывают сложные языковые конструкции, контекст, а также не могут адекватно справляться с типичными и менее очевидными ошибками, что делает их использование недостаточным для реальных задач. Проект направлен на реализацию не только функционального, но и удобного для пользователя инструмента, который поможет значительно повысить качество текстов, улучшив их восприятие и удобочитаемость. Такой подход особенно актуален для профессионалов, работающих с текстами: редакторов, журналистов, студентов, преподавателей и всех, кто ежедневно создаёт и редактирует письменные материалы. Для достижения поставленной цели будет рассмотрен ряд ключевых задач: Анализ существующих подходов к обработке естественного языка и их ограничений. Будет изучен опыт применения традиционных алгоритмов и современных моделей машинного обучения для решения задач автоматического исправления текста. Качество данных играет решающую роль в машинном обучении, поэтому большое внимание уделено созданию качественного набора данных, включающего тексты с разнообразными типами ошибок. Обучение модели на основе подготовленного датасета. Для этого используется модель T5, специально дообученная для решения задач исправления ошибок. Разработка веб-приложения для внедрения модели в удобный интерфейс, который позволяет пользователю быстро вводить текст, запускать проверку и получать исправленный текст с визуальным выделением всех найденных ошибок. Система, разработанная в рамках данного проекта, станет полезным инструментом как для профессионалов, так и для широкой аудитории, поскольку она не только улучшит качество текста, но и сделает процесс редактирования более доступным и удобным. Использование машинного обучения позволяет снизить затраты времени на проверку и исправление текста, обеспечивая высокую точность за счёт адаптации к контексту. Основное преимущество моделей машинного обучения по сравнению с традиционными алгоритмами заключается в их способности адаптироваться к изменениям языка, учитывать грамматический контекст и автоматически обновляться на основе новых данных. Модель T5, применяемая в этом проекте, является одной из передовых моделей, разработанных для работы с задачами текст-в-текст, где на вход подается исходный текст, а на выходе генерируется исправленный текст с учётом всех найденных ошибок. T5 демонстрирует высокую эффективность при работе с различными типами ошибок, что делает её особенно ценной для задач корректировки текста. Современные технологии NLP предлагают широкий спектр инструментов для обработки текстов, начиная от простых коррекций правописания и заканчивая полноценными инструментами анализа текста на основе глубокого обучения. В последние годы активное развитие получили трансформерные модели, такие как GPT, BERT и T5, которые продемонстрировали выдающиеся результаты в понимании языка и контекстуального анализа. Эти модели способны анализировать контекст и учитывать сложные языковые конструкции, а значит, подходить к задаче исправления текста более гибко, чем классические инструменты. Для повышения качества текстов эти алгоритмы могут быть применены как в индивидуальной работе, так и на уровне автоматизированных систем. Интеграция модели T5 в веб-приложение позволяет пользователям легко взаимодействовать с моделью и получать исправленный текст в реальном времени, что существенно упрощает процесс проверки текстов и делает его доступным для широкого круга пользователей. Настоящая работа является шагом к созданию надёжного инструмента для автоматического исправления текста, однако возможности его развития значительно шире. В будущем проект может быть дополнен поддержкой других языков, более глубоким контекстным анализом, что позволит модели лучше разбираться в стиле текста и, при необходимости, учитывать грамматические и стилистические особенности определённых областей (например, научной или деловой лексики). Благодаря постоянному обучению и доработке моделей машинного обучения такой инструмент может стать полноценной платформой для улучшения текстов, которая будет адаптироваться к нуждам пользователей и новым стандартам языка. Таким образом, создание и внедрение системы автоматического исправления текста на основе машинного обучения открывает новые возможности для повышения качества письменных материалов и упрощения процесса их редактирования. Проект представляет собой не только полезное и удобное приложение для коррекции текстов, но и продвинутый пример применения методов машинного обучения для решения актуальных задач обработки естественного языка.

1. Анализ предметной области и описание технологии проектирования ИС
   1. Описание предметной области (проблемы проекта)

Проблема орфографических и пунктуационных ошибок является актуальной для многих сфер деятельности, где требуется высокая точность и грамотность текстов. В условиях быстрого роста объема текстовой информации, а также ужесточения требований к качеству текстов, необходимость автоматической проверки правописания становится все более востребованной. Журналисты, редакторы, студенты, писатели и профессионалы в различных областях регулярно создают большие объемы текстов и часто сталкиваются с необходимостью быстрой проверки на наличие орфографических и пунктуационных ошибок. Однако качественная проверка требует значительных усилий, так как тексты содержат не только стандартные грамматические конструкции, но и сложные речевые обороты, уникальные профессиональные термины и неологизмы, что значительно затрудняет процесс автоматической проверки.

Проблемы традиционных методов проверки

Большинство существующих решений для автоматической проверки текста, таких как встроенные правописные модули в текстовых редакторах, основаны на статических правилах и словарях, что накладывает серьёзные ограничения на их возможности. Традиционные методы, опирающиеся на фиксированные правила, нередко не справляются со сложными грамматическими структурами и контекстно зависимыми ошибками. Эти системы могут игнорировать ошибки или, наоборот, ошибочно выделять корректные слова и конструкции, поскольку их работа ориентирована на использование базовых лингвистических правил и заранее определённых словарей. Такие ограничения приводят к следующим проблемам: Отсутствие учета контекста: Традиционные методы проверяют текст изолированно, не учитывая контекст, в котором используется слово или фраза. В результате ошибки, связанные с контекстом, часто остаются незамеченными, так как правила не адаптированы под реальную языковую среду. Современный язык постоянно меняется и обогащается новыми словами, особенно в профессиональных областях. В узкоспециализированных текстах часто встречаются термины, которые отсутствуют в словарях стандартных проверочных систем. Это приводит к тому, что такие слова помечаются как ошибки, что сбивает пользователя и ограничивает возможности автоматической проверки. Многие грамматические ошибки трудно исправить с помощью фиксированных правил, поскольку язык — динамичная система, где правила зачастую взаимодействуют между собой. Например, в одном и том же предложении может быть ошибка в постановке знаков препинания и согласовании слов, и для таких исправлений требуется гибкий подход, способный учитывать всю структуру предложения.

Ограничения существующих решений

Существующие решения для автоматической проверки текста, такие как стандартные функции проверки в текстовых редакторах или доступные онлайн-сервисы, часто не отвечают потребностям пользователей, так как они предлагают лишь поверхностные исправления. Проблема состоит в том, что такие решения: Могут упускать или неправильно интерпретировать ошибки, оставляя текст с недостатками. Ориентированы на общую аудиторию и не подходят для специализированных текстов, где необходимо учитывать уникальные речевые особенности, характерные для конкретной профессии или области знаний. Не имеют адаптивности к изменениям в языке, что делает их устаревшими в условиях быстрого развития профессиональных и научных терминологий. Личный опыт работы с одним из таких сайтов, которые не оправдал моих ожиданий, показал, что большинство подобных решений оказываются недостаточно точными и гибкими. Сайт, на который я изначально рассчитывал, некорректно обрабатывал текст и допускал ошибки в исправлениях, что сделало его использование неудобным и непродуктивным. Этот опыт стал ключевым стимулом для разработки собственного решения, которое обеспечило бы точность исправлений и отвечало потребностям в проверке текста на уровне, превышающем возможности традиционных методов.

Необходимость создания более точного и гибкого инструмента

Проблемы, возникающие при использовании традиционных методов и имеющихся онлайн-сервисов, демонстрируют необходимость создания более точного и гибкого инструмента для проверки и исправления текста. Такой инструмент должен обладать способностью адаптироваться к контексту, учитывать сложные грамматические конструкции и работать с профессиональным лексиконом. Для этого была выбрана модель машинного обучения T5, которая благодаря своей архитектуре позволяет анализировать текст с учётом его смысловой структуры и контекста. Машинное обучение открывает возможности для решения проблемы, поскольку модели обучаются на больших объемах данных и могут учитывать огромное количество примеров правильного и неправильного использования языка.

Преимущества использования методов машинного обучения

Машинное обучение представляет собой перспективный подход к решению задачи автоматической проверки текста, так как оно позволяет: Адаптироваться к изменяющемуся языку: В отличие от традиционных методов, модели машинного обучения могут обновляться и обучаться на новых данных, что позволяет им учитывать современные тенденции в языке и быть в курсе новых слов и терминов. Системы, построенные на основе методов машинного обучения, способны анализировать не только отдельные слова, но и их окружение, что помогает определять контекстные ошибки, которые могут менять смысл текста. Модели машинного обучения могут обрабатывать значительные объемы данных и работать с длинными текстами, предоставляя результаты проверки в разумные сроки.

Таким образом, современные задачи проверки текста требуют решения, которое может не только исправлять орфографические и пунктуационные ошибки, но и учитывать контекст, сложные грамматические конструкции и специфику профессионального языка. В этом проекте будет разработан инструмент, который решает эти проблемы, создавая более точную и удобную систему для проверки текстов.

* 1. Определение целевых групп пользователей

Профессиональные редакторы часто работают с большими объемами текстов, включая статьи, книги, технические документы, маркетинговые материалы и другие типы контента. Для них важны точность и скорость при проверке и редактировании текстов. Существующие инструменты проверки правописания могут не всегда справляться с контекстными ошибками или специфическими требованиями к форматированию текста. Редакторы заинтересованы в инструменте, который сможет выявлять не только орфографические ошибки, но и сложные пунктуационные ошибки, а также исправлять грамматические неточности. Ручная правка текста требует значительных временных затрат, особенно если требуется проверка множества документов. Автоматизированное решение может помочь ускорить этот процесс, предоставив редакторам качественные предложения по исправлению, что позволит сократить количество ручной работы. Журналисты, блогеры и копирайтеры создают контент на регулярной основе. Для них важно, чтобы текст был не только интересным и информативным, но и свободным от ошибок, поскольку это напрямую влияет на восприятие и доверие аудитории. Авторы часто работают с большими объемами текста в ограниченные сроки, что увеличивает вероятность ошибок и опечаток. Им необходимо решение, которое поможет быстро проверять и исправлять тексты, сохраняя при этом их оригинальность и авторский стиль. Инструмент должен быть гибким, позволяя авторам самостоятельно выбирать, какие исправления применить, а какие отклонить. Также важно, чтобы система была способна выявлять ошибки в контексте, учитывая индивидуальные особенности стиля письма и специфику различных тем. Моё решение может стать помощником для авторов, значительно ускоряя процесс редактуры, помогая исправить ошибки и улучшить читабельность текста. Студенты, особенно пишущие курсовые и дипломные работы, также могут быть заинтересованы в автоматической проверке текстов. Им часто необходимо проверять большие объемы текста на наличие ошибок, особенно когда они работают над важными академическими заданиями. Каждая из этих целевых групп нуждается в инструменте, который будет адаптирован под их специфические задачи и требования.

* 1. Определение ограничений проектного решения

Для обучения и дообучения модели машинного обучения T5 требуется большое количество качественно размеченных данных. Создание таких датасетов может быть трудоемким процессом, требующим значительных усилий для обеспечения точности разметки и покрытия различных типов ошибок. Отсутствие достаточного количества данных может снизить качество и точность модели. Качество обучающих данных напрямую влияет на эффективность модели. Ошибки в разметке, неполные данные или нерелевантные примеры могут негативно повлиять на результаты работы модели. Важно обеспечить высокое качество и разнообразие данных для успешного обучения модели. Модели машинного обучения, хотя и мощные, могут сталкиваться с трудностями при обработке сложных контекстных ошибок или специфических языковых конструкций. Подбор правильных гиперпараметров и настройка модели для учета различных контекстов могут потребовать значительных усилий и времени. Требуются значительных вычислительных ресурсов, включая мощные графические процессоры (GPU) или специализированные аппаратные ускорители (TPU). Эти ресурсы могут быть недоступными для начинающих.

* 1. Анализ предметной области

Обработка текстовой информации, особенно когда речь идёт о проверке орфографии и пунктуации, представляет собой сложную и многоаспектную задачу, которая требует тщательного анализа и разработки эффективных методов. Сегодня текстовая информация является неотъемлемой частью жизни большинства людей. Ежедневно создаются миллиарды текстов, от коротких сообщений до многостраничных документов, и каждый из них нуждается в корректной грамматике, пунктуации и правописании. Ошибки в таких текстах не только снижают их читабельность, но и влияют на восприятие автора, а также на общее качество и профессиональный уровень материала. Тексты, написанные с ошибками, могут нанести урон репутации как автора, так и компании или бренда. Например, в бизнесе некачественно написанные электронные письма могут создать у партнёров впечатление непрофессионализма, в то время как грамотно составленные и правильно оформленные документы, напротив, усиливают доверие и подчеркивают ответственность компании. То же самое касается и других областей, таких как журналистика, наука и образование, где текст играет ключевую роль. Тем не менее, обеспечить безупречное качество текста без специальных инструментов для его проверки и исправления зачастую оказывается сложно. Процесс обработки текста имеет множество нюансов, которые накладывают ограничения на традиционные методы проверки правописания. Они, как правило, построены на фиксированных правилах и словарях, которые не всегда учитывают контекст и сложные грамматические конструкции. Например, простая проверка орфографии может исправить слово с ошибкой, но в более сложных ситуациях, когда ошибки зависят от синтаксической структуры предложения или требуют учета соседних слов, традиционные системы часто оказываются недостаточно эффективными. В частности, такие системы могут неверно распознать ошибки, игнорировать сложные случаи или не учитывать современные изменения в языке. Одной из основных проблем автоматической обработки текстов является необходимость учитывать разнообразие языковых конструкций и их сложные связи. В естественном языке слова и предложения имеют множество значений, которые могут изменяться в зависимости от контекста, и корректное понимание этих значений требует гибкого подхода. Например, омонимы — слова, которые пишутся одинаково, но имеют разные значения в зависимости от контекста, — представляют особую сложность для стандартных систем проверки. Без использования контекста такие системы не могут точно распознать, какое значение слова используется в конкретном предложении, что приводит к неправильному исправлению ошибок или их пропуску. Кроме того, особую сложность представляют грамматические структуры, в которых порядок слов влияет на значение фразы. Такие структуры часто встречаются в журналистике, научных текстах и профессиональной литературе, где важна точность формулировок. Традиционные методы не способны анализировать структуру сложных предложений, особенно если речь идет о неполных или необычных конструкциях. Они могут допускать ошибки в случаях, когда предложение построено нестандартно, что снижает общую точность системы. Язык постоянно развивается, появляются новые слова, термины и выражения, особенно в таких областях, как информационные технологии, маркетинг, наука и культура. Это делает традиционные системы, работающие на основе словарей, неэффективными, так как они не успевают адаптироваться к изменениям. Например, новый термин может отсутствовать в словаре, и система отметит его как ошибку, хотя на самом деле слово корректно. Подобные ситуации встречаются особенно часто в текстах, написанных в определённых профессиональных областях, где использование специализированных терминов является нормой. В таких случаях инструмент для исправления текста должен обладать возможностью динамического обновления, чтобы учитывать новые лексические единицы и правила их употребления. Существующие решения для автоматической проверки правописания и пунктуации зачастую имеют недостаточную точность и гибкость. Многие популярные сервисы предлагают стандартные функции проверки орфографии, но их возможности по исправлению пунктуации и учету контекста ограничены. Эти инструменты могут быть полезными для базовой проверки правописания, но они не всегда подходят для профессиональной работы с текстом. Например, корректура статей, научных работ, юридических и технических документов требует высокого уровня точности и учёта грамматического контекста, а существующие решения не всегда могут обеспечить такой уровень. Это вызывает потребность в более комплексных инструментах, способных справляться со сложными случаями, анализируя текст на глубоком уровне. Методы машинного обучения, особенно на основе глубоких нейронных сетей, представляют собой перспективный подход к решению проблем, связанных с автоматической корректировкой текста. Эти методы позволяют моделям обучаться на больших объёмах данных и адаптироваться к языковым изменениям и различным контекстам. Модели машинного обучения способны учитывать сложные грамматические конструкции и обнаруживать ошибки, которые были бы пропущены традиционными системами. Например, модель T5, которая используется в рамках данного проекта, разработана специально для задач преобразования текста, что позволяет ей исправлять ошибки на уровне фраз и предложений, не теряя смысла исходного текста. Особое значение в машинном обучении имеет контекстуальный анализ. Модель обучается на больших текстовых корпусах, где слова и выражения используются в различных контекстах, что позволяет ей «учиться» распознавать значения слов в зависимости от их окружения. Это делает модели машинного обучения особенно эффективными для обработки сложных текстов, где ошибки зависят от контекста, и позволяет разрабатывать гибкие инструменты для проверки и исправления текста, адаптируемые под разнообразные языковые задачи. Таким образом, различия между традиционными и современными подходами к обработке текста заключаются в следующих аспектах: Гибкость и адаптивность: Традиционные методы ограничены фиксированными правилами и не учитывают контекст. Модели машинного обучения, напротив, могут учитывать семантические и синтаксические особенности текста, что делает их более гибкими и точными. Традиционные методы часто не справляются с новыми словами и терминами, в то время как модели машинного обучения могут обучаться на данных с неологизмами и терминологией, специфичной для определённых областей. Традиционные системы требуют ручного обновления правил и словарей, что может занимать время. Модели машинного обучения автоматически обновляются и обучаются на новых данных, что позволяет им быстрее адаптироваться к изменениям языка. Анализ предметной области показал, что существующие инструменты для автоматической проверки текста недостаточно точны и универсальны для работы с профессиональными текстами.

1. Функциональные требования к системе
   1. Определение требований к эргономике

Интерфейс построен так, чтобы исключить все ненужные элементы и визуальный шум. Основные функции представлены максимально просто и логично, что позволяет пользователю сосредоточиться на выполнении задачи без отвлекающих элементов. Все ключевые элементы, такие как поле ввода текста, кнопка проверки и поле для отображения исправленного текста, размещены так, чтобы пользователи могли сразу понять их назначение и не тратили время на поиск нужных функций. Навигация построена таким образом, чтобы пользователь быстро находил нужные действия и мог легко ориентироваться. Чтобы повысить удобство исправления, ошибки выделяются красным цветом, что помогает пользователям сразу видеть и понимать, где в тексте есть проблемы. Это выделение сделано достаточно ярким, но при этом ненавязчивым, чтобы не мешать восприятию всего текста. Интерфейс реагирует мгновенно на действия пользователя, обеспечивая плавное и предсказуемое взаимодействие. Например, исправленный текст появляется без задержек, что делает работу с приложением быстрой и удобной.

* 1. Определение требований к надежности системы

Приложение должно корректно обрабатывать длинные тексты, включая крупные документы, статьи и другие материалы, которые содержат большое количество символов. Это особенно важно для пользователей, работающих с профессиональными или академическими текстами, где объем документа может быть значительным. Приложение должно оставаться стабильным и функциональным даже при высоких нагрузках на сервер, когда одновременно множество пользователей отправляют свои тексты на проверку. Для этого рекомендуется внедрить мониторинг производительности и использовать кэширование или распределение нагрузки на сервере.

Приложение должно обрабатывать ситуации, когда пользователь вводит некорректные данные, например, пустой текст или текст, содержащий только специальные символы. В таких случаях система должна обнаруживать ошибку и уведомлять пользователя о необходимости ввести валидный текст. В случае некорректного ввода или возникновения технической ошибки, система должна предоставлять понятные сообщения с объяснением причин проблемы. Это позволяет пользователю быстро понять суть ошибки и предпринять действия для её устранения. Исправление текста должно происходить в разумные сроки, чтобы пользователи не испытывали неудобств при ожидании результата. Для этого нужно оптимизировать работу модели машинного обучения и настроить серверные процессы так, чтобы обеспечивать быстрое время отклика даже при больших объёмах текста. Чтобы избежать долгих задержек, система должна быть оптимизирована для обработки каждого запроса в кратчайшие сроки. В случае задержек можно добавить индикатор загрузки, чтобы пользователь знал, что процесс идёт и результат будет предоставлен скоро. Важные данные, такие как обученная модель и конфигурации приложения, должны регулярно резервироваться для предотвращения потери данных в случае сбоя или обновления. Резервное копирование позволяет быстро восстановить систему и снизить риски, связанные с непредвиденными техническими проблемами. Пользовательские данные должны обрабатываться безопасно, в соответствии с требованиями конфиденциальности. Это особенно важно для обеспечения доверия пользователей к приложению и соблюдения стандартов безопасности.

* 1. Ввод текста

Текстовое поле позволяет вводить и редактировать большие фрагменты текста, включая статьи, длинные документы или отчёты. Это обеспечивает гибкость в использовании приложения и удобство работы с текстами любого объёма. Текстовое поле автоматически подстраивается под введённый текст, чтобы при увеличении объёма поле расширялось и обеспечивало пользователю комфортное восприятие. Это позволяет легко редактировать текст без необходимости постоянно прокручивать его вверх и вниз. Поле для ввода имеет чёткие границы и нейтральный фон, что позволяет легко фокусироваться на содержимом и не отвлекает пользователя от задачи. Чёткие границы также помогают пользователю видеть, где начинается и заканчивается поле для ввода текста. Текстовое поле расположено рядом с полем вывода исправленного текста, что делает интерфейс удобным для работы и позволяет сразу видеть исправления, не переключаясь между экранами. Независимо от того, используют ли пользователи приложение для коротких заметок или для редактирования больших документов, текстовое поле адаптируется под нужды каждого пользователя, делая процесс ввода и редактирования удобным и непрерывным.

* 1. Обнаружение и исправление ошибок

Одной из ключевых функций приложения является анализ текста на наличие орфографических и пунктуационных ошибок, их автоматическое исправление и наглядное отображение исправлений для пользователя. Система использует дообученную модель T5 для определения и исправления ошибок, что делает процесс проверки точным и гибким, позволяя исправлять различные виды ошибок в тексте, включая орфографические и пунктуационные. Приложение анализирует текст на наличие различных пунктуационных ошибок, включая неправильное использование запятых, точек, дефисов, тире, кавычек и других знаков препинания. Это помогает улучшить структуру текста и его читаемость, делая его более профессиональным и понятным для читателя. Благодаря дообученной модели T5 приложение способно учитывать контекст предложения и определять ошибки, которые могут быть неочевидны при использовании традиционных методов. Модель обучена распознавать закономерности и определять ошибки в зависимости от контекста, что улучшает качество исправлений.

Типы ошибок, которые обнаруживает приложение

Орфографические ошибки: Приложение автоматически исправляет ошибки в написании слов, обнаруживая опечатки и неточности. Это включает замену неверно написанных слов на правильные и исправление опечаток.

Приложение исправляет распространённые ошибки, такие как:

Неправильная расстановка запятых: Например, отсутствие запятых в сложных предложениях или лишние запятые в простых конструкциях. Система распознаёт случаи, когда необходимо использовать тире вместо дефиса и наоборот. Приложение исправляет ошибки в расстановке точек и заменяет множественные точки на одно многоточие при необходимости. Ошибки в использовании кавычек: Система автоматически исправляет случаи неправильного использования кавычек, обеспечивая соответствие знаков препинания и структуры предложения. После анализа текста система автоматически применяет исправления, обновляя текст с устранением обнаруженных ошибок. Приложение также сохраняет исходную структуру текста, чтобы пользователь мог легко найти и проанализировать исправленные места. Благодаря возможности исправления как орфографических, так и пунктуационных ошибок, приложение помогает пользователям улучшить качество своих текстов, делая их более читаемыми и профессиональными. Автоматическое исправление ошибок позволяет пользователю сократить время, затрачиваемое на ручную проверку текста. Пользователю достаточно ввести текст и нажать кнопку проверки, чтобы увидеть исправления сразу же, что особенно полезно при работе с большими объемами текста. Использование модели T5 позволяет приложению точно определять и исправлять ошибки даже в сложных текстах, что делает его более эффективным по сравнению с традиционными средствами проверки правописания, которые могут пропускать контекстные ошибки.

* 1. Выделение ошибок

Независимо от типа ошибки (орфографическая, пунктуационная или грамматическая), все исправления выделяются красным цветом. Это позволяет пользователю легко заметить исправленные места в тексте, так как красный цвет привлекает внимание и выделяется на фоне основного текста. Оттенок красного цвета подобран так, чтобы выделяться на фоне текста, не отвлекая от основного содержания. Это создаёт баланс, при котором ошибки видны, но общая читаемость текста остаётся комфортной. Поскольку красный цвет применяется ко всем типам ошибок, пользователь не тратит время на попытки понять, какой именно тип ошибки исправлен. Все выделенные части сразу показывают места, где текст был изменён, что упрощает анализ. В приложении используются только цветовые акценты без дополнительных визуальных элементов, таких как подчёркивание или курсив. Это делает процесс выделения предельно простым и понятным для пользователей. Красный цвет эффективно направляет внимание пользователя к исправленным фрагментам текста, помогая сразу увидеть внесённые изменения. Это позволяет пользователю не отвлекаться на анализ исправлений и сосредоточиться на чтении исправленного текста. Благодаря единому цвету выделение ошибок остаётся понятным и ненавязчивым. Пользователь может легко просматривать текст в целом, при этом не теряя из виду внесённые изменения.

* 1. Интерфейс пользователя

Интерфейс сайта должен быть интуитивно понятным и легким в использовании. Пользователь должен легко находить основные функции, такие как ввод текста, проверка и исправление ошибок. Интерфейс должен быть выполнен в стиле минимализма, избегая избыточных элементов и визуального шума. Основное внимание должно быть сосредоточено на ключевых функциях, чтобы пользователь мог легко находить и использовать их. Простой и чистый дизайн помогает пользователю сосредоточиться на задачах, а не на изучении интерфейса.

* 1. Ограничение по языку проверки

Веб-приложение разработано для работы с текстами на русском языке, что позволяет максимально адаптировать функционал для языковых особенностей и повысить качество исправлений. Это ограничение было введено для упрощения интерфейса и повышения точности исправлений, так как сложные многоязычные правила могли бы затруднить работу системы и снизить эффективность работы с текстами.

1. Описание этапов разработки системы искусственного интеллекта
   1. Описание исходных данных

Для обучения модели использовался специально собранный и проверенный датасет, включающий тексты с различными типами ошибок: орфографическими, пунктуационными и грамматическими. Датасет был собран из разных источников, включая новостные сайты, форумы и научные тексты. Важным этапом было ручное исправление разметки, чтобы улучшить качество данных.

* 1. Сравнение алгоритмов машинного обучения

В процессе работы я рассмотрел несколько альтернативных моделей для исправления текста, включая GPT-2 и BERT. Однако модель T5 была выбрана благодаря ее способности решать задачи преобразования текста, что делает ее идеальной для исправления ошибок в тексте. T5 позволяет работать с задачей преобразования текста "с ошибками" в "исправленный текст", что идеально соответствует цели проекта.

* 1. Описание структуры приложения

Веб-приложение разработано с использованием фреймворка Django и основано на архитектуре клиент-сервер. Такая структура позволяет эффективно обрабатывать запросы пользователей, обеспечивая быстрый доступ к функции исправления текста с помощью модели T5. Основные компоненты и взаимодействия приложения описаны ниже.

Архитектура клиент-сервер

Клиентская часть: На клиентской стороне пользователь взаимодействует с приложением через веб-интерфейс. Он вводит текст в текстовое поле на веб-странице и отправляет его на проверку, после чего получает исправленный текст с визуальным выделением ошибок. Клиентская часть построена с использованием HTML5 и CSS3 и обеспечивает доступный интерфейс для работы с текстом.

Серверная часть: Django выступает в роли серверной части приложения, отвечая за обработку запросов от клиента. Сервер обрабатывает отправленный текст, передает его в модель для исправления и возвращает результат клиенту. Эта структура позволяет делегировать все вычисления и работу с моделью серверу, обеспечивая плавную работу приложения для пользователя. Views (контроллеры): Views играют роль контроллеров, обрабатывающих запросы пользователя. Когда пользователь отправляет текст на проверку, Django View получает запрос, инициализирует обработку и передает текст модели для исправления. После получения исправленного текста он отправляет его обратно на клиентскую часть, где он отображается пользователю. Серверная часть приложения взаимодействует с дообученной моделью T5, которая была интегрирована в Django для выполнения задач исправления текста. Текст, отправленный пользователем, поступает на вход модели T5, которая анализирует его, обнаруживает ошибки и возвращает исправленный текст. Django выполняет роль промежуточного слоя, принимая текст от клиента, передавая его модели и возвращая результат. Взаимодействие с моделью организовано через API, которые позволяют отправлять текст для обработки и получать исправленный текст, не привязываясь к конкретным запросам пользователя. В приложении определены функции, которые выполняют различные этапы обработки текста. Одна из таких функций инициирует передачу текста модели T5, другая отвечает за получение исправленного текста и его возврат клиенту. Чтобы минимизировать время ожидания, используется механизм асинхронной обработки, который позволяет серверу обрабатывать несколько запросов одновременно. Это особенно важно для задач, требующих вычислительных ресурсов, таких как исправление текста с использованием модели T5. Для оптимизации работы приложения можно предусмотреть кэширование результатов для повторяющихся запросов. Например, если один и тот же текст отправляется несколько раз, сервер может сохранить исправленный вариант, избегая повторной обработки моделью.

Схема взаимодействия клиент-сервер

Этап 1: Ввод и отправка текста: Пользователь вводит текст в текстовое поле на веб-странице и нажимает кнопку для запуска проверки. Клиентская часть отправляет текст на сервер через HTTP-запрос.

Этап 2: Передача текста модели T5: Django получает запрос, извлекает текст из него и передает его на обработку модели T5, запущенной на сервере.

Этап 3: Обработка и исправление текста: Модель T5 анализирует текст, определяет ошибки и возвращает исправленную версию текста серверу.

Этап 4: Возврат исправленного текста клиенту: После обработки Django получает исправленный текст от модели и возвращает его клиенту. На клиентской стороне текст выводится в соседнем текстовом поле с выделением ошибок.

Преимущества клиент-серверной структуры

Централизованное управление: Все задачи, связанные с вычислениями и обработкой текста, выполняются на сервере, что упрощает управление и обновление модели. При необходимости сервер можно обновить или дообучить модель, не затрагивая клиентскую часть. Пользователю не нужно иметь мощное оборудование для работы с моделью — все ресурсоёмкие процессы выполняются на сервере, что делает приложение доступным для любого устройства с веб-браузером.

Такая структура позволяет легко добавлять новые функции. Например, можно добавить другие модели для исправления ошибок или разнообразные алгоритмы обработки текста.

* 1. Разработка макетов страниц

Макеты страниц были разработаны с учетом принципов минимализма и удобства. Пользовательский интерфейс включает текстовое поле для ввода текста, кнопку для запуска проверки и отображение исправленного текста с выделением ошибок.

* 1. Описание дизайна приложения

Дизайн веб-приложения выполнен в минималистичном стиле, который позволяет пользователям сосредоточиться на исправлении текста, не отвлекаясь на излишние визуальные элементы. Основная цель дизайна — предоставить пользователю простой и интуитивно понятный интерфейс для ввода текста, его проверки и просмотра исправлений.

На главной странице пользователи видят только необходимые элементы: большое текстовое поле для ввода текста и кнопку для запуска проверки. Это обеспечивает удобство навигации и интуитивность, позволяя пользователю сразу приступить к проверке текста.

В интерфейсе нет дополнительных кнопок или функций, таких как выбор языка или переключение между типами проверок. Такое упрощение дизайна помогает пользователю полностью сосредоточиться на основной задаче — проверке текста на ошибки.

Ошибки, найденные системой, выделяются с помощью цветовых акцентов. Например, орфографические ошибки могут подсвечиваться красным, а пунктуационные — другим оттенком, чтобы ошибки сразу бросались в глаза, при этом не отвлекая внимание от основного текста.

При выборе цветовой схемы использованы спокойные и приглушенные оттенки, которые делают ошибки заметными, но не вызывают излишнего визуального напряжения. Такое решение позволяет пользователю видеть и понимать места исправлений без перенасыщения экрана яркими цветами.

Просторное текстовое поле позволяет вводить большие объемы текста, адаптируясь под его размер. Поле разработано с учетом удобства пользователя, оно сохраняет читабельность и комфорт даже при увеличении объема введенного текста.

Ошибки в тексте выделяются визуально, но система не предоставляет пояснений о природе ошибок. Это обеспечивает простоту восприятия — пользователю нужно только просмотреть текст и принять или отклонить исправления на основе визуальных выделений.

Кнопка для запуска проверки текста расположена непосредственно под текстовым полем, что позволяет пользователю легко ее найти и запустить процесс проверки. После нажатия на кнопку текст обновляется, и система отображает ошибки с помощью цветовых акцентов.

Пользовательский опыт не осложнен дополнительными переключателями или настройками. Это помогает сохранить интуитивность и простоту интерфейса, так как от пользователя требуется только ввести текст и нажать на кнопку проверки.

Основные цвета интерфейса — светлые и спокойные, что создает чувство легкости и не отвлекает от текста. Это помогает пользователю фокусироваться на проверке текста и восприятии исправлений.

В интерфейсе используется стандартный, легко читаемый шрифт, который не отвлекает и удобен для восприятия. Это обеспечивает удобочитаемость текста в текстовом поле и облегчает восприятие исправлений.

Таким образом, минималистичный и интуитивный дизайн приложения "Исправиста" помогает пользователям сосредоточиться на задаче проверки текста. Визуальное выделение ошибок с помощью цветовых акцентов позволяет быстро находить и исправлять ошибки без отвлечения на дополнительные пояснения, выбор языка или переключение между режимами. Это создает комфортный пользовательский опыт, ориентированный на простоту и удобство.

* 1. Описание приемов верстки

Для разработки веб-приложения "Исправиста" использовались современные стандарты HTML5 и CSS3, что позволило создать простой, удобный и адаптивный интерфейс, ориентированный на комфортное использование. Вёрстка была реализована с учетом минималистичного дизайна, направленного на удобство пользователей, а также на быструю и эффективную проверку текста.

Структура страниц построена с использованием семантических тегов HTML5, таких как <header>, <main>, <section>, <footer>. Это улучшает доступность контента и помогает пользователям и поисковым системам легко ориентироваться на сайте.

Для ввода текста было использовано поле <textarea>, которое позволяет пользователям вводить большие объемы текста и адаптируется к его размеру. Поле обладает функцией автоматического расширения при увеличении объема текста, чтобы поддерживать удобство и читабельность.

Кнопка для запуска проверки текста и формы для его ввода структурированы в логические блоки, что упрощает навигацию по странице и делает её более интуитивной.

Текстовое поле автоматически подстраивается под введённый текст и остаётся комфортным для чтения, не теряя своей функциональности и удобства на экранах различного размера.

* 1. Анализ и описание технологий программирования

Для реализации проекта и обеспечения функциональности автоматического исправления текста были выбраны передовые инструменты и технологии программирования, обеспечивающие высокую производительность, гибкость и масштабируемость системы. Основные используемые технологии включают библиотеку transformers от Hugging Face для работы с моделью машинного обучения. Библиотека transformers от Hugging Face является одним из самых популярных инструментов для работы с моделями машинного обучения, основанными на архитектуре трансформеров. Эта библиотека предоставляет доступ к множеству предобученных моделей для различных задач обработки естественного языка (NLP), включая исправление текста. В рамках проекта была выбрана модель T5, которая поддерживает задачи преобразования текста и эффективна для задач исправления орфографических и пунктуационных ошибок. Библиотека transformers включает множество готовых к использованию моделей, таких как BERT, GPT и T5. Для проекта была выбрана модель T5, так как она способна решать задачи преобразования текста, принимая на вход исходный текст и генерируя исправленную версию: Одним из главных преимуществ transformers является возможность легко дообучить предобученные модели на собственных данных. Это позволяет адаптировать модель к специфическим задачам и улучшить её точность, что особенно важно для исправления текстов на разных языках и в различных областях.

Интеграция с Python и другими библиотеками: transformers интегрируется с такими библиотеками, как datasets для работы с данными и torch для использования GPU, что упрощает процесс обучения и тестирования модели. Это позволило ускорить обработку данных и снизить вычислительные затраты.

Для создания веб-интерфейса приложения была выбрана платформа Django — один из наиболее популярных веб-фреймворков на Python. Django предлагает удобные средства для разработки и управления веб-приложениями, что позволяет быстро создавать, тестировать и развертывать системы. Использование Django дало возможность создать масштабируемое и гибкое приложение, которое легко адаптируется под требования пользователей. Django разделяет приложение на отдельные модули, такие как модели, представления и шаблоны, что позволяет легко управлять кодом и добавлять новые функции. В рамках "Исправиста" были разработаны модули, отвечающие за работу с текстом, отправку запросов на проверку и отображение результатов. Django позволяет создать API для взаимодействия с моделью машинного обучения, что упрощает обмен данными между клиентской и серверной частью. API принимает текст от пользователя, отправляет его на обработку модели и возвращает исправленный текст для отображения на клиенте. Django предоставляет встроенные инструменты для защиты от SQL-инъекций, межсайтовых запросов и других угроз безопасности, что позволяет защитить данные пользователей и обеспечить безопасное взаимодействие с системой. Django разработан для работы с высокими нагрузками, что позволяет масштабировать приложение при увеличении числа пользователей и объёма текстов. Фреймворк поддерживает интеграцию с базами данных и кэшированием, что помогает оптимизировать производительность.

Python был выбран как основной язык программирования проекта благодаря своей простоте, богатой экосистеме библиотек и возможностям для работы с машинным обучением. Его популярность и широкое применение в сфере Data Science и NLP делают Python идеальным выбором для разработки подобных приложений. Python позволяет писать лаконичный и понятный код, что делает разработку более продуктивной и снижает риск ошибок. Python предоставляет богатый набор библиотек для работы с данными и моделями машинного обучения, таких как transformers, numpy, pandas и torch. Это упрощает обработку данных, подготовку и обучение моделей, а также интеграцию машинного обучения в веб-приложение. Python имеет обширное сообщество и обширную документацию, что облегчает поиск решений для сложных задач и поддержку проекта в будущем.

Для обработки запросов и хранения данных был настроен сервер на базе Django, который обрабатывает запросы пользователей, передаёт текст в модель T5 для исправления и возвращает результат. Важно, чтобы серверная часть могла эффективно работать с большими объёмами данных и обрабатывать многочисленные запросы пользователей. Серверная часть управляет запросами, поступающими от клиента, передаёт их модели и возвращает обработанный текст. Это позволяет разгрузить клиентскую часть и оставить основную обработку на сервере. Для оптимизации производительности и сокращения времени отклика используется кэширование повторяющихся запросов. Это позволяет снизить нагрузку на сервер при обработке одинаковых текстов. В серверной части предусмотрены механизмы для обработки ошибок и ведения логов, что помогает администратору отслеживать работу приложения и устранять возможные проблемы. В проекте также были задействованы другие библиотеки, обеспечивающие высокую эффективность и удобство работы с данными и интерфейсом.

* 1. Выбор домена и хостинга

Для размещения приложения был выбран хостинг, поддерживающий Python и Django. Были проведены исследования доступных вариантов, и выбран оптимальный провайдер, обеспечивающий надежную работу и поддержку веб-приложения.

* 1. Описание этапов тестирования программного решения

Тестирование включало несколько этапов:

Функциональное тестирование: проверка корректной работы всех функций приложения.

Тестирование производительности: оценка времени отклика системы при различных объемах текста.

Пользовательское тестирование: сбор обратной связи от потенциальных пользователей для улучшения интерфейса и функциональности.

4. Руководство пользователя

После загрузки страницы веб-приложения пользователь видит два текстовых поля: одно для ввода текста, а другое — для отображения исправленного текста. Ниже текстовых полей расположена кнопка для запуска проверки текста. Интерфейс имеет минималистичный дизайн, позволяя пользователю сосредоточиться на вводе текста и просмотре исправлений. В первом текстовом поле пользователь вводит текст, который необходимо проверить на ошибки. Это поле поддерживает ввод больших объемов текста, что позволяет обрабатывать как короткие заметки, так и длинные тексты. Поле ввода настроено так, чтобы отображать весь введенный текст, делая его удобным для чтения и редактирования. После ввода текста пользователь нажимает кнопку «Исправить текст», расположенную под полем ввода. При нажатии на кнопку текст передается на сервер для анализа и обработки. Приложение начинает проверку, и пользователю следует подождать, пока текст будет обработан и исправлен. После завершения проверки исправленный текст отображается во втором текстовом поле, расположенном возле первого поля. Все обнаруженные ошибки выделены цветом, что помогает пользователю сразу увидеть исправленные места в тексте. Цветовые акценты подчеркивают ошибки, не отвлекая внимание от основного текста. Пользователь может просмотреть исправленный текст, сравнить его с исходным, а затем, при необходимости, скопировать его для использования в других приложениях. Если требуется повторная проверка, пользователь может обновить текст в первом поле и снова нажать кнопку «Исправить текст».

5. Руководство администратора

Администратор должен следить за состоянием серверов, на которых работает приложение, контролируя загрузку CPU и GPU, потребление памяти и сетевой трафик. Высокая производительность сервера важна для обеспечения быстрого отклика при обработке текстов пользователями. Настройка автоматических оповещений для выявления потенциальных сбоев или чрезмерной нагрузки на сервер поможет администратору вовремя выявить проблемы и оперативно на них отреагировать. Администратор должен периодически обновлять и дообучать модель T5 на основе новых данных или улучшений алгоритмов, чтобы поддерживать высокую точность исправлений. Обновление модели может также включать внедрение новых функций, если это требуется. После дообучения администратор загружает новую версию модели на сервер и тестирует её работу. Это позволяет избежать сбоев и проверить, что исправления работают корректно, а производительность сервера остается на должном уровне. Если приложение требует авторизации или ограниченного доступа для определённых пользователей, администратор должен настраивать и контролировать систему прав доступа. Это включает создание учётных записей, назначение ролей и управление привилегиями пользователей. При необходимости администратор должен вносить изменения в права доступа, например, добавлять новых пользователей, изменять их роли или блокировать доступ если такое существует. Администратор должен периодически проверять, как функционирует интерфейс сайта, включая корректность расположения текстовых полей. В интерфейсе приложения поле ввода текста и поле вывода исправленного текста расположены рядом друг с другом, что обеспечивает удобство использования и лёгкость для сравнения исходного текста и исправлений. Администратор должен проверять работу модели на корректность исправлений, контролируя визуальное выделение ошибок и их правильность. Регулярное тестирование поможет обеспечить качество сервиса и улучшить восприятие пользователями. Регулярное резервное копирование модели, данных и конфигураций поможет предотвратить потерю информации в случае непредвиденных сбоев или обновлений. Администратор обязан следить за безопасностью данных пользователей и модели, включая защиту от несанкционированного доступа и утечек информации.

Заключение

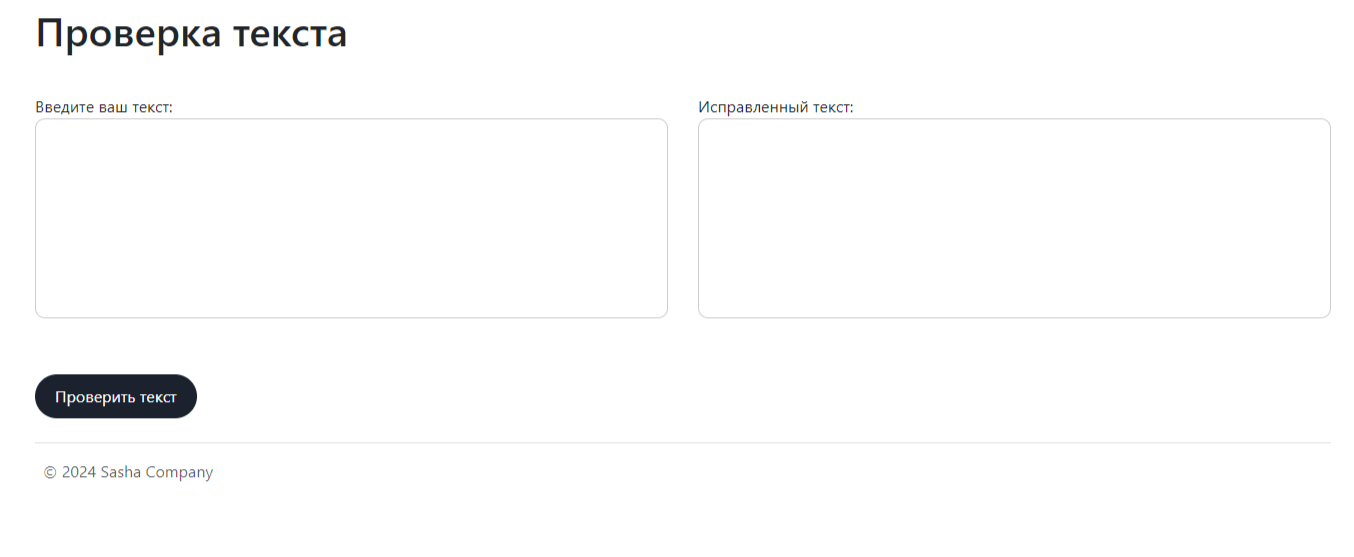
В рамках данной работы была разработана система автоматического исправления орфографических и пунктуационных ошибок на основе модели T5. Мы провели анализ существующих методов автоматической корректировки текста, изучили архитектуру модели T5 и особенности её применения для задач обработки естественного языка. Для реализации проекта был подготовлен и обработан обучающий набор данных, содержащий пары текстов: исходные тексты с ошибками и их корректные версии. Процесс обучения включал в себя настройку гиперпараметров модели, выбор стратегии оптимизации, а также проверку результатов на валидационных данных. Важной частью работы стало создание веб-интерфейса, позволяющего пользователям вводить текст для проверки. Разработанный интерфейс прост и интуитивно понятен, что обеспечивает удобство использования. Также были учтены такие аспекты, как минималистичный дизайн и акцент на подсветку ошибок, что помогает пользователю наглядно видеть результаты корректировки. Результаты демонстрируют высокую эффективность модели в исправлении орфографических и пунктуационных ошибок. Это подтверждает, что модель T5, дообученная на специально подготовленном наборе данных, может успешно решать задачи исправления текста. В будущем система может быть дополнена функциональностью, включающей расширенную обработку текстов на разных языках, предоставление объяснений для обнаруженных ошибок и интеграцию с другими платформами. Это позволит улучшить пользовательский опыт и расширить сферу применения разработанного инструмента. Работа показала значимость использования современных моделей глубокого обучения для автоматической обработки текста, а также продемонстрировала перспективы их применения в реальных задачах.

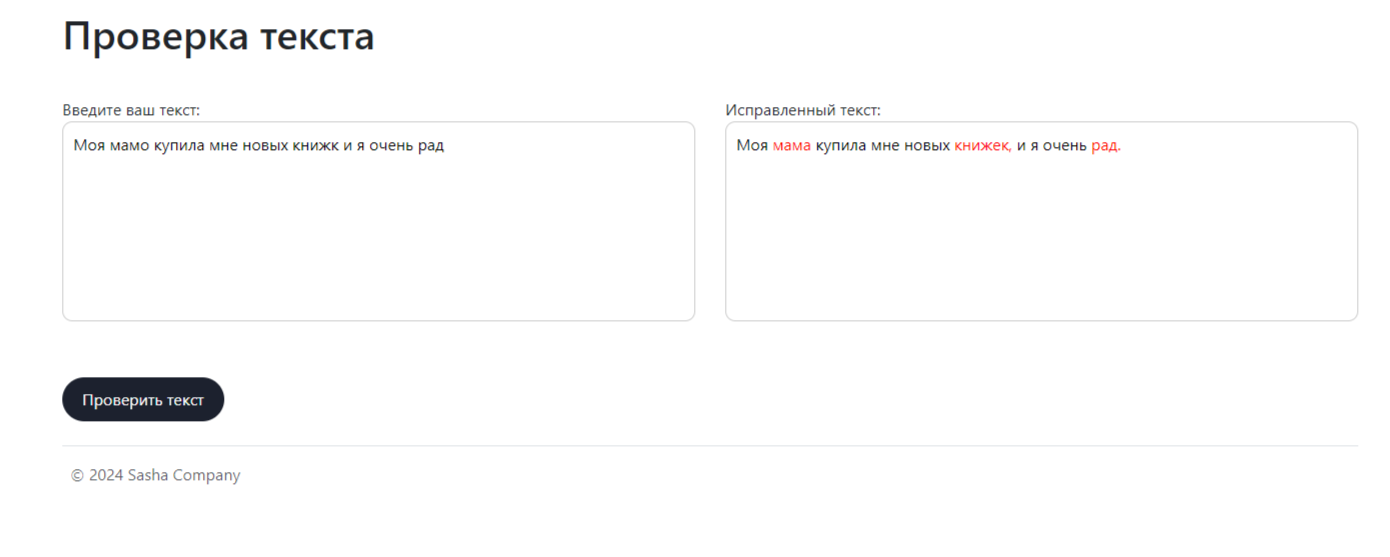
Список используемых источников

1. Hugging Face. Библиотека Transformers: [Электронный ресурс].  
   URL: <https://huggingface.co/>  
   (Дата обращения: 15.10.2024).
2. Hugging Face. Работа с датасетами: [Электронный ресурс].  
   URL: https://huggingface.co/docs/datasets/  
   (Дата обращения: 15.10.2024).
3. Django. Официальная документация: [Электронный ресурс].  
   URL: <https://docs.djangoproject.com/>  
   (Дата обращения: 15.10.2024).
4. PyTorch. Официальная документация: [Электронный ресурс].  
   URL: https://pytorch.org/docs/  
   (Дата обращения: 15.10.2024).
5. Google Colab. Официальная платформа для обучения моделей: [Электронный ресурс].  
   URL: https://colab.research.google.com/  
   (Дата обращения: 16.10.2024).
6. NLTK (Natural Language Toolkit). Инструменты для обработки текста: [Электронный ресурс].  
   URL: <https://www.nltk.org/>  
   (Дата обращения: 16.10.2024).
7. Medium. Работы с трансформерами и T5: [Электронный ресурс].  
   URL: <https://medium.com/>  
   (Дата обращения: 17.10.2024).
8. Kaggle. Датасеты для обучения моделей: [Электронный ресурс].  
   URL: <https://www.kaggle.com/>  
   (Дата обращения: 17.10.2024).
9. Stack Overflow. Решение проблем при разработке: [Электронный ресурс].  
   URL: <https://stackoverflow.com/>  
   (Дата обращения: 17.10.2024).
10. Модель T5-small. Hugging Face Hub: [Электронный ресурс].  
    URL: https://huggingface.co/t5-small  
    (Дата обращения: 17.10.2024).

Приложение

Приложение А  
(Обязательное)

Интерфейс



Приложение Б  
(Обязательное)

Листинг программы

Models.py

from django.db import models

class CorrectionRequest(models.Model):

    original\_text = models.TextField()  # Введённый пользователем текст

    corrected\_text = models.TextField()  # Исправленный текст

    timestamp = models.DateTimeField(auto\_now\_add=True)  # Время запроса

    def \_\_str\_\_(self):

        return f"{self.original\_text} -> {self.corrected\_text}"

Urls.py

from django.urls import path

from . import views

urlpatterns = [

    path('', views.index, name = 'home'),

    path('about-us', views.about, name = 'about'),

    path('api/word-count/', views.word\_count, name='word\_count'),

]

Views.py

from django.shortcuts import render

from django.http import JsonResponse

from collections import Counter

from .models import CorrectionRequest

from transformers import T5Tokenizer, T5ForConditionalGeneration

# Загрузка модели и токенизатора

tokenizer = T5Tokenizer.from\_pretrained('E:\\python\\site\\wordsite\\modd')  # Укажите путь к tokenizer.json и spiece.model

model = T5ForConditionalGeneration.from\_pretrained('E:\\python\\site\\wordsite\\modd')  # Укажите путь к model.safetensors или pytorch\_model.bin

# Обработка главной страницы сайта

def index(request):

    output\_text = ''

    input\_text = ''  # Переменная для хранения введенного текста

    if request.method == 'POST':

        input\_text = request.POST.get('input\_text')

        if input\_text:

            # Токенизация текста

            inputs = tokenizer(input\_text, return\_tensors='pt', max\_length=512, truncation=True)

            # Генерация исправленного текста

            outputs = model.generate(inputs['input\_ids'], max\_length=512, num\_beams=4, early\_stopping=True)

            corrected\_text = tokenizer.decode(outputs[0], skip\_special\_tokens=True)

            # Подсвечиваем ошибки

            output\_text = highlight\_errors(input\_text, corrected\_text)

            # Сохраняем в базу данных

            CorrectionRequest.objects.create(

                original\_text=input\_text,

                corrected\_text=corrected\_text

            )

    return render(request, 'main/index.html', {'input\_text': input\_text, 'output\_text': output\_text})

# Функция для подсветки ошибок в исправленном тексте

def highlight\_errors(original\_text, corrected\_text):

    original\_words = original\_text.split()

    corrected\_words = corrected\_text.split()

    highlighted\_text = []

    for orig\_word, corr\_word in zip(original\_words, corrected\_words):

        if orig\_word != corr\_word:

            # Подсвечиваем ошибки красным цветом

            highlighted\_text.append(f'<span style="color: red;">{corr\_word}</span>')

        else:

            highlighted\_text.append(corr\_word)

    # Объединяем слова обратно в строку с подсветкой

    return ' '.join(highlighted\_text)

# API для подсчета количества слов

def word\_count(request):

    # Получаем все исправленные тексты из базы

    all\_corrections = CorrectionRequest.objects.values\_list('corrected\_text', flat=True)

    # Считаем частоту всех слов

    word\_counter = Counter()

    for text in all\_corrections:

        words = text.split()  # Разбиваем текст на слова

        word\_counter.update(words)

    # Формируем словарь для ответа

    word\_count\_data = dict(word\_counter)

    return JsonResponse({'word\_counts': word\_count\_data})

# Обработка страницы "О проекте"

def about(request):

    return render(request, 'main/about.html')

About.html

{% extends 'main/base.html' %}

{% block title %}

Страница про сайт

{% endblock %}

{% block content %}

<div class="wrapper  bg-white ">

                <div class="box">

                    <div class="padding-big">

                        <div class="block-constructor">

                 <div class="content-wrapper">

    <div class="block-constructor">

<h3 class="h2">О САЙТЕ</h3>

<div class="columns large-12 medium-12 small-12">

<div class="col">

<div class="content-wrapper">

           <div> <br> </div>

           <p class="MsoNormal">"Исправиста" — это инновационный онлайн-сервис, который преобразит ваше восприятие текстовой коррекции.

            Этот уникальный инструмент не только улучшает качество ваших текстов, но и делает процесс редактирования быстрым, эффективным и интеллектуальным. Это самый быстрый и точный сайт с исправлением слов и текста<ul></ul>

            Независимо от того, кем вы являетесь:</p>

<ul>

<li>Студентом</li>

<li>Писателем</li>

<li>Учеником</li>

<li>Редактором</li>

<li>Журналистом</li>

</ul>

</div>

<p class="content-wrapper">"Исправиста" поможет вам довести текст до совершенства, выделяя ошибки и предоставляя исправления в одном удобном интерфейсе.

    </p>

</div>

</div>

</div>

</div>

<div style="clear: both"></div>

                                </div>

                    </div>

                </div>

            </div>

{% endblock %}

Base.html

<!doctype html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <meta name="viewport"

          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>{% block title %}{% endblock %}</title>

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">

</head>

<body>

    <div>

    <header class="d-flex flex-wrap justify-content-center py-3 mb-4 border-bottom">

      <a href="{% url 'home' %}" class="d-flex align-items-center mb-3 mb-md-0 me-md-auto link-body-emphasis text-decoration-none">

        <svg class="bi me-2" width="10" height="32"><use xlink:href="#bootstrap"></use></svg>

        <span class="fs-4">Исправиста</span>

      </a>

      <ul class="nav nav-pills">

        <li class="nav-item"><a href="{% url 'about' %}" class="nav-link">О сайте</a></li>

      </ul>

    </header>

</div>

    <div class="container">

    {% block content %}{% endblock %}

    </div>

    <svg xmlns="http://www.w3.org/2000/svg" style="display: none;">

  <symbol id="facebook" viewBox="0 0 16 16">

    <path d="M16 8.049c0-4.446-3.582-8.05-8-8.05C3.58 0-.002 3.603-.002 8.05c0 4.017 2.926 7.347 6.75 7.951v-5.625h-2.03V8.05H6.75V6.275c0-2.017 1.195-3.131 3.022-3.131.876 0 1.791.157 1.791.157v1.98h-1.009c-.993 0-1.303.621-1.303 1.258v1.51h2.218l-.354 2.326H9.25V16c3.824-.604 6.75-3.934 6.75-7.951z"></path>

  </symbol>

  <symbol id="instagram" viewBox="0 0 16 16">

      <path d="M8 0C5.829 0 5.556.01 4.703.048 3.85.088 3.269.222 2.76.42a3.917 3.917 0 0 0-1.417.923A3.927 3.927 0 0 0 .42 2.76C.222 3.268.087 3.85.048 4.7.01 5.555 0 5.827 0 8.001c0 2.172.01 2.444.048 3.297.04.852.174 1.433.372 1.942.205.526.478.972.923 1.417.444.445.89.719 1.416.923.51.198 1.09.333 1.942.372C5.555 15.99 5.827 16 8 16s2.444-.01 3.298-.048c.851-.04 1.434-.174 1.943-.372a3.916 3.916 0 0 0 1.416-.923c.445-.445.718-.891.923-1.417.197-.509.332-1.09.372-1.942C15.99 10.445 16 10.173 16 8s-.01-2.445-.048-3.299c-.04-.851-.175-1.433-.372-1.941a3.926 3.926 0 0 0-.923-1.417A3.911 3.911 0 0 0 13.24.42c-.51-.198-1.092-.333-1.943-.372C10.443.01 10.172 0 7.998 0h.003zm-.717 1.442h.718c2.136 0 2.389.007 3.232.046.78.035 1.204.166 1.486.275.373.145.64.319.92.599.28.28.453.546.598.92.11.281.24.705.275 1.485.039.843.047 1.096.047 3.231s-.008 2.389-.047 3.232c-.035.78-.166 1.203-.275 1.485a2.47 2.47 0 0 1-.599.919c-.28.28-.546.453-.92.598-.28.11-.704.24-1.485.276-.843.038-1.096.047-3.232.047s-2.39-.009-3.233-.047c-.78-.036-1.203-.166-1.485-.276a2.478 2.478 0 0 1-.92-.598 2.48 2.48 0 0 1-.6-.92c-.109-.281-.24-.705-.275-1.485-.038-.843-.046-1.096-.046-3.233 0-2.136.008-2.388.046-3.231.036-.78.166-1.204.276-1.486.145-.373.319-.64.599-.92.28-.28.546-.453.92-.598.282-.11.705-.24 1.485-.276.738-.034 1.024-.044 2.515-.045v.002zm4.988 1.328a.96.96 0 1 0 0 1.92.96.96 0 0 0 0-1.92zm-4.27 1.122a4.109 4.109 0 1 0 0 8.217 4.109 4.109 0 0 0 0-8.217zm0 1.441a2.667 2.667 0 1 1 0 5.334 2.667 2.667 0 0 1 0-5.334z"></path>

  </symbol>

  <symbol id="twitter" viewBox="0 0 16 16">

    <path d="M5.026 15c6.038 0 9.341-5.003 9.341-9.334 0-.14 0-.282-.006-.422A6.685 6.685 0 0 0 16 3.542a6.658 6.658 0 0 1-1.889.518 3.301 3.301 0 0 0 1.447-1.817 6.533 6.533 0 0 1-2.087.793A3.286 3.286 0 0 0 7.875 6.03a9.325 9.325 0 0 1-6.767-3.429 3.289 3.289 0 0 0 1.018 4.382A3.323 3.323 0 0 1 .64 6.575v.045a3.288 3.288 0 0 0 2.632 3.218 3.203 3.203 0 0 1-.865.115 3.23 3.23 0 0 1-.614-.057 3.283 3.283 0 0 0 3.067 2.277A6.588 6.588 0 0 1 .78 13.58a6.32 6.32 0 0 1-.78-.045A9.344 9.344 0 0 0 5.026 15z"></path>

  </symbol>

</svg>

<div class="container">

  <footer class="d-flex flex-wrap justify-content-between align-items-center py-3 my-4 border-top">

    <div class="col-md-4 d-flex align-items-center">

      <a href="/" class="mb-3 me-2 mb-md-0 text-body-secondary text-decoration-none lh-1">

      </a>

      <span class="mb-3 mb-md-0 text-body-secondary">© 2024 Sasha Company</span>

    </div>

  </footer>

</div>

</body>

</html>

Index.html

{% extends 'main/base.html' %}

{% load static %}

{% block title %}

Главная страница

{% endblock %}

{% block content %}

    <style>

        h1 {

            margin-bottom: 40px; /\* Увеличенный отступ под заголовком \*/

        }

        .form-container {

            display: flex;

            justify-content: space-between;

            align-items: flex-start;

            gap: 30px; /\* Увеличенное расстояние между полями \*/

            margin-bottom: 20px;

        }

        .form-container textarea {

            width: 100%; /\* Одинаковая ширина для обоих полей \*/

            min-height: 200px;

            resize: none;

            padding: 10px;

            border: 1px solid #ccc;

            border-radius: 10px;

            font-size: 16px;

            line-height: 1.5;

            background-color: white; /\* Добавляем белый фон, как у обычного textarea \*/

            color: black; /\* Цвет текста для нормального отображения \*/

            transition: border-color 0.3s ease; /\* Плавный переход цвета рамки \*/

        }

        .form-container textarea:focus {

            border-color: #3498db; /\* Голубая рамка при фокусе \*/

            outline: none; /\* Убираем стандартный обвод \*/

        }

        .form-container div.corrected-text {

            width: 100%;

            min-height: 200px;

            padding: 10px;

            border: 1px solid #ccc;

            border-radius: 10px;

            font-size: 16px;

            line-height: 1.5;

            background-color: white; /\* Белый фон для исправленного текста \*/

            color: black; /\* Цвет текста \*/

            white-space: pre-wrap; /\* Сохранение переносов строк \*/

            transition: border-color 0.3s ease; /\* Плавный переход цвета рамки \*/

        }

        .form-container div.corrected-text:focus {

            border-color: #3498db; /\* Голубая рамка при фокусе \*/

            outline: none; /\* Убираем стандартный обвод \*/

        }

        .form-group button {

            display: inline-block;

            padding: 10px 20px;

            border: none;

            border-radius: 25px;

            background-color: #1C212E;

            color: #fff;

            font-size: 16px;

            cursor: pointer;

            margin-top: 20px;

            transition: background-color 0.3s ease;

        }

        .form-group button:hover {

            background-color: #3498db;

        }

        /\* Обёртка формы \*/

        form {

            display: flex;

            flex-direction: column;

            gap: 10px;

            max-width: 100%;

        }

        .input-container, .output-container {

            flex: 1; /\* Одинаковая ширина для обоих контейнеров \*/

        }

    </style>

    <h1>Проверка текста</h1>

    <form method="POST">

        {% csrf\_token %}

        <div class="form-container">

            <div class="input-container">

                <label for="input\_text">Введите ваш текст:</label>

                <textarea id="input\_text" name="input\_text" required oninput="autoExpand(this)">{{ input\_text }}</textarea>

            </div>

            <div class="output-container">

                <label for="output\_text">Исправленный текст:</label>

                <!-- Используем div вместо textarea для отображения исправленного текста с подсветкой ошибок -->

                <div id="output\_text" class="corrected-text" tabindex="0">{{ output\_text|safe }}</div>

            </div>

        </div>

        <div class="form-group">

            <button type="submit">Проверить текст</button>

        </div>

    </form>

    <script>

        // Функция для автоматического расширения текстового поля

        function autoExpand(field) {

            field.style.height = 'auto';

            field.style.height = (field.scrollHeight) + 'px';

        }

    </script>

{% endblock %}

Resu.py

print('Запуск кода')

import torch

from transformers import AutoModelForSeq2SeqLM, T5TokenizerFast, Trainer, TrainingArguments

from datasets import load\_dataset, Dataset

import pandas as pd

import random

import re

# Параметры

MODEL\_NAME = 'C://Users//user//Downloads//fsdfs//ezpz'  # Исходная модель

DATASET\_NAME = 'ai-forever/spellcheck\_punctuation\_benchmark'

MAX\_INPUT\_LENGTH = 128

MAX\_TARGET\_LENGTH = 128

BATCH\_SIZE = 16

EPOCHS = 3

LEARNING\_RATE = 3e-5

device = "cuda" if torch.cuda.is\_available() else "cpu"

print("Используемое устройство:", device)

torch.cuda.empty\_cache()

print('Кэш очищен')

# Загрузка токенизатора и модели

tokenizer = T5TokenizerFast.from\_pretrained(MODEL\_NAME)

model = AutoModelForSeq2SeqLM.from\_pretrained(MODEL\_NAME)

# Загрузка исходного датасета

dataset = load\_dataset(DATASET\_NAME, 'MultidomainGold', trust\_remote\_code=True)

def remove\_punctuation(text):

    return re.sub(r'[.,]', '', text)

# Увеличение датасета

def generate\_mistakes(word, n=5):

*"""Создаёт n вариантов с ошибками для одного слова."""*

    def drop\_random\_char(w):  # Удаляет случайный символ

        if len(w) > 1:

            idx = random.randint(0, len(w) - 1)

            return w[:idx] + w[idx+1:]

        return w

    def add\_random\_char(w):  # Добавляет случайный символ

        idx = random.randint(0, len(w))

        char = random.choice("абвгдеёжзийклмнопрстуфхцчшщъыьэюя")

        return w[:idx] + char + w[idx:]

    def swap\_adjacent\_chars(w):  # Меняет соседние символы

        if len(w) > 1:

            idx = random.randint(0, len(w) - 2)

            return w[:idx] + w[idx+1] + w[idx] + w[idx+2:]

        return w

    def duplicate\_random\_char(w):  # Дублирует случайный символ

        if len(w) > 0:

            idx = random.randint(0, len(w) - 1)

            return w[:idx] + w[idx] + w[idx] + w[idx+1:]

        return w

    mistakes = []

    for \_ in range(n):

        error\_type = random.choice([drop\_random\_char, add\_random\_char, swap\_adjacent\_chars, duplicate\_random\_char])

        mistakes.append(error\_type(word))

    return mistakes

# Преобразование исходного датасета в формат DataFrame

original\_data = [{"source": item['source'], "correction": item['correction']} for item in dataset['train']]

df = pd.DataFrame(original\_data)

# Генерация новых данных с ошибками

augmented\_data = []

for \_, row in df.iterrows():

    correct\_text = row['correction']

    mistakes = generate\_mistakes(correct\_text)

    for mistake in mistakes:

        augmented\_data.append({"source": mistake, "correction": correct\_text})

# Объединение исходных и увеличенных данных

df\_augmented = pd.DataFrame(augmented\_data)

df\_combined = pd.concat([df, df\_augmented])

# Преобразование в Dataset

augmented\_dataset = Dataset.from\_pandas(df\_combined)

# Токенизация данных

def preprocess\_data(batch):

    inputs = [f"Spell correct: {text}" for text in batch['source']]

    targets = batch['correction']

    model\_inputs = tokenizer(inputs, max\_length=MAX\_INPUT\_LENGTH, truncation=True, padding="max\_length")

    labels = tokenizer(targets, max\_length=MAX\_TARGET\_LENGTH, truncation=True, padding="max\_length")

    model\_inputs['labels'] = labels['input\_ids']

    return model\_inputs

tokenized\_dataset = augmented\_dataset.map(preprocess\_data, batched=True)

train\_test\_split = tokenized\_dataset.train\_test\_split(test\_size=0.1)

train\_dataset = train\_test\_split['train']

eval\_dataset = train\_test\_split['test']

# Параметры обучения

training\_args = TrainingArguments(

    output\_dir="C://Users//user//Downloads//fsdfs//saved",

    evaluation\_strategy="epoch",

    learning\_rate=LEARNING\_RATE,

    per\_device\_train\_batch\_size=BATCH\_SIZE,

    per\_device\_eval\_batch\_size=BATCH\_SIZE,

    num\_train\_epochs=EPOCHS,

    weight\_decay=0.01,

    save\_total\_limit=1,

    save\_strategy="epoch",

    logging\_dir='C://Users//user//Downloads//fsdfs//logs',

    logging\_steps=1000,

    load\_best\_model\_at\_end=True,

    metric\_for\_best\_model="loss"

)

# Создание и запуск тренера

trainer = Trainer(

    model=model,

    args=training\_args,

    train\_dataset=train\_dataset,

    eval\_dataset=eval\_dataset,

    tokenizer=tokenizer

)

trainer.train()

# Сохранение модели

trainer.save\_model("C://Users//user//Downloads//fsdfs//modelfinal")

tokenizer.save\_pretrained("C://Users//user//Downloads//fsdfs//modelfinal")

print("Обучение завершено и модель сохранена.")