Реализация модели для обнаружения сгенерированных текстов

1 Постановка задачи

В последние годы широкое распространение получили крупные языковые модели, способные генерировать тексты, трудноотличимые от написанных человеком. Такие модели могут быть использованы не только для полезных задач, но и для создания фейковых новостей, спама, пропаганды и других видов недобросовестного контента. В связи с этим возникает практическая задача: обнаружение сгенерированных текстов.

В рамках данной работы задача формулируется как задача бинарной классификации: по входному тексту необходимо определить, был ли он написан человеком (Human) или сгенерирован одной из языковых моделей (Non-Human). В качестве эмбеддингов используется предобученная модель DeepPavlov/rubert-base-cased, после чего классификатор обучается на признаковом пространстве.

2 Используемый набор данных и их подготовка

Для обучения и оценки модели используется датасет RuATD 2022, предназначенный для задачи детектирования искусственно сгенерированных текстов на русском языке. Он содержит тексты, сгенерированные различными языковыми моделями (например, ruGPT3, mT5, ruT5, M-BART и др.), а также тексты, написанные человеком (Human). Подготовка данных включает следующие этапы:

* Загрузка и чтение CSV-файлов: используются два файла — train.csv (обучающая выборка) и val.csv (валидационная выборка). Каждая строка содержит идентификатор (Id), текст (Text) и исходный класс (Class).
* Преобразование меток для бинарной классификации. Все тексты с меткой Human получают значение метки 0, отражающее тексты, написанные человеком. Все остальные классы (генерированные моделями) объединяются в метку 1, что позволяет свести задачу к бинарной классификации и сосредоточиться на выявлении «нечеловеческих» текстов.
* Токенизация и векторизация. Для каждого текста выполняется токенизация с использованием модели rubert-base-cased. Далее тексты подаются в модель, где для каждого токена вычисляется скрытое представление. Затем эмбеддинг всего текста формируется методом mean pooling — усреднением векторов по всем токенам с учётом маски внимания (attention\_mask).
* Сохранение признаков. Полученные эмбеддинги (векторы размерности 768) и соответствующие метки сохраняются в формате .pt (бинарные файлы библиотеки torch) для последующего использования на этапе обучения модели классификации.

Такой подход позволяет получить компактное, фиксированной размерности представление для каждого текста, подходящее для подачи в любую классическую модель машинного обучения.

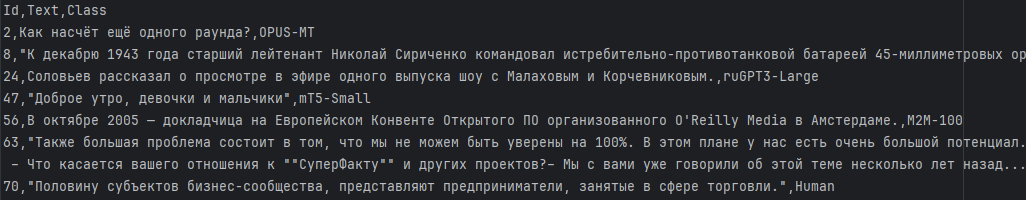


Рисунок 1 – демонстрация разметки набора данных

3 Использование предобученной модели DeepPavlov/rubert-base-cased

Для извлечения признаков из текстов используется предобученная трансформерная модель DeepPavlov/rubert-base-cased, основанная на архитектуре BERT и адаптированная под русский язык. Данная модель обучена на больших русскоязычных корпусах, что делает её хорошо подходящей для понимания и представления текстов на русском языке.

Модель rubert-base-cased реализует механизм само-внимания (self-attention), позволяющий учитывать контекст слов в пределах всего предложения. На выходе модель формирует векторные представления для каждого токена. Для получения одного вектора на весь текст используется агрегация по токенам с помощью mean pooling — усреднение скрытых состояний по токенам с учётом их значимости, заданной маской внимания. Достоинствами использования данной модели можно назвать:

* Поддержку кириллического алфавита и морфологии русского языка;
* Глубокое представление семантики текста, подходящее для задач классификации;
* Отсутствие необходимости дополнительного обучения модели: она используется как функция преобразования текста в вектор признаков.

Таким образом, rubert-base-cased выступает в качестве эмбеддингового блока, обеспечивая качественное и устойчивое представление текстов для последующего обучения модели классификации.

3.4 Обучение модели

Для классификации текстов на сгенерированные искусственным интеллектом и написанные человеком была реализована глубокая нейросетевая модель на основе полносвязного многослойного перцептрона (MLP). Архитектура модели реализована с использованием библиотеки PyTorch и представлена в виде класса DeepMLClassifier:

Листинг 1 – класс DeepMLClassifier

class DeepMLPClassifier(nn.Module):

def \_\_init\_\_(self, input\_dim, dropout=0.4):

super().\_\_init\_\_()

self.net = nn.Sequential(

nn.Linear(input\_dim, 768),

nn.ReLU(),

nn.Dropout(dropout),

nn.Linear(768, 256),

nn.ReLU(),

nn.Dropout(dropout),

nn.Linear(256, 64),

nn.ReLU(),

nn.Dropout(dropout),

nn.Linear(64, 2)

)

def forward(self, x):

return self.net(x)

Модель принимает на вход эмбеддинги размерностью 768, полученные предварительно (например, с помощью BERT-подобной модели). Далее данные проходят через серию линейных слоёв с функцией активации ReLU и регуляризацией в виде Dropout (с вероятностью отключения нейронов 0.4). Последний слой возвращает логиты для двух классов: "человеческий текст" и "сгенерированный ИИ".

Обучение проводилось на 30 эпохах, с использованием кросс-энтропийной функции потерь (CrossEntropyLoss) и оптимизатора Adam. Процесс обучения сопровождался валидацией модели на отложенной выборке. Ниже представлены примеры логов из процесса обучения:

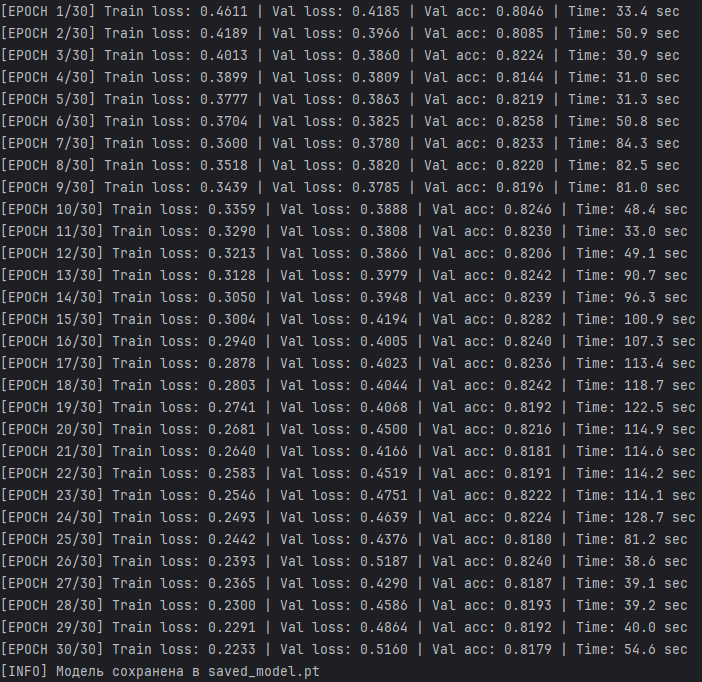


Рисунок 2 – демонстрация логов при обучении

Из логов видно, что в течение первых 10 эпох наблюдается стабильное уменьшение функции потерь и рост точности. Начиная примерно с 15–20 эпохи модель начинает переобучаться: несмотря на снижение тренировочной ошибки, валидационная ошибка увеличивается. Тем не менее, точность на валидации сохраняется на высоком уровне (~82%).

Финальная модель была сохранена на диск в файл saved\_model.pt и использовалась для итоговой оценки качества.

5 Оценка качества модели

Для финальной оценки модели использовалась тестовая выборка размером 21 511 объектов. Классы представлены неравномерно:

* Человеческие тексты: 8 524 объекта;
* Сгенерированные ИИ: 12 987 объектов.

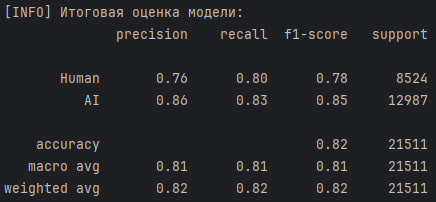


Рисунок 3 – демонстрация итоговой оценки

Общая точность модели составила 0.82 (82%). Дополнительно были рассчитаны усреднённые показатели:

* Macro average: precision = 0.81, recall = 0.81, F1 = 0.81
* Weighted average: precision = 0.82, recall = 0.82, F1 = 0.82

Эти метрики показывают, что модель хорошо различает оба класса, несмотря на дисбаланс данных. Особенно высокие значения precision и recall для класса "AI" (0.86 и 0.83 соответственно) указывают на способность модели эффективно определять сгенерированные тексты.

6 Сохранение и использование модели

После завершения обучения классификационной нейросети модель сохраняется в файл saved\_model.pt. Это позволяет исключить необходимость повторного обучения при каждом запуске, обеспечить воспроизводимость результатов и упростить переносимость модели между средами. Сохранение выполняется с использованием средств библиотеки PyTorch, которая предоставляет компактное и удобное средство сохранения весов нейронной сети в формате .pt. Для демонстрации работы модели и возможности её практического применения было разработано простое веб-приложение на основе микрофреймворка Flask. Его задачей является организация пользовательского интерфейса для анализа текстов, поданных на вход, и визуализация результата — вероятности того, что текст сгенерирован искусственным интеллектом.

После запуска сервера Flask пользователь может открыть страницу по адресу http://127.0.0.1:5000. Интерфейс позволяет ввести произвольный текст и отправить его на обработку. Далее выполняются следующие шаги:

* Токенизация текста с использованием того же токенизатора, что применялся на этапе обучения.
* Преобразование текста в эмбеддинги с помощью модели RuBERT.
* Классификация с помощью заранее обученного MLP, загруженного из saved\_model.pt.
* Отображение результата — пользователю показывается вероятность того, что текст был сгенерирован ИИ.

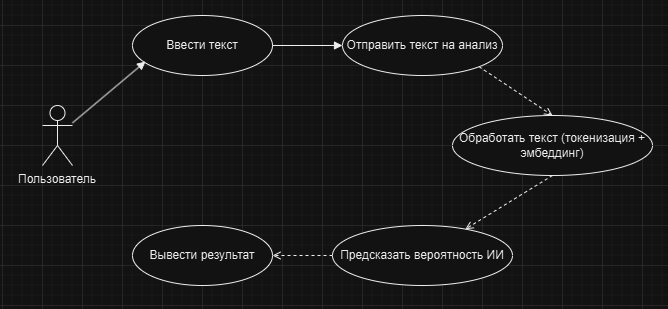


Рисунок 4 – диаграмма активности

Этот подход обеспечивает полноценную интеграцию модели в интерактивную среду и демонстрирует её применение в условиях, приближенных к реальным. Благодаря отделению этапов обучения и инференса достигается высокая гибкость, масштабируемость и простота использования.

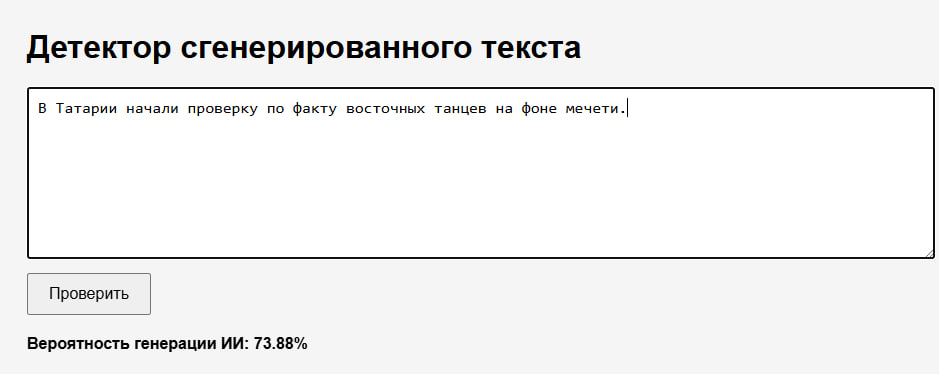


Рисунок 5 – демонстрация работы приложения