Проект AI-Stream

Студент 1 курса магистратуры ИТМО «Искусственный интеллект»
Бокарев Степан

Показать сообщения, отфильтрованные с помощью категорий AutoMod

Contents of this template

You can delete this slide when you're done editing the presentation

Fonts	To view this template correctly in PowerPoint, download and install the fonts we used
Used and **alternative resources**	An assortment of graphic resources that are suitable for use in this presentation
Thanks slide	You must keep it so that proper credits for our design are given
Colors	All the colors used in this presentation
Icons and **infographic resources**	These can be used in the template, and their size and color can be edited
Editable presentation theme	You can edit the master slides easily. For more info, click **here**

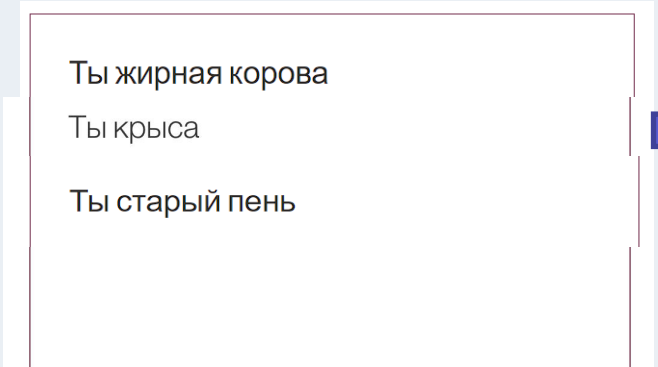
For more info:
SLIDESGO | **BLOG** | **FAQs**

You can visit our sister projects:
FREEPIK | **FLATICON** | **STORYSET** | **WEPIK** | **VIDEVO**

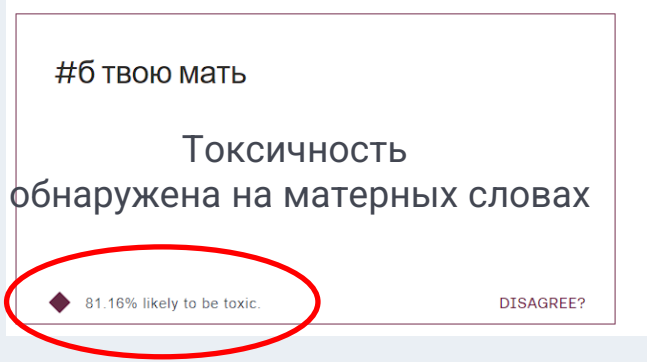
Проблема

Как улучшить взаимодействие стримеров с аудиторией в реальном времени, минимизируя негативное влияние токсичных комментариев и повышая общее качество общения на трансляциях?
1.Токсичные сообщения ухудшают атмосферу на стриме
2.Спам и реклама мешают конструктивному общению
3.Понимание настроений и реакции зрителей помогает стримерам адаптировать контент и коммуникацию в режиме реального времени.

Текущие решения для русского языка плохо справляются с распознаванием, токсичных комментариев https://www.perspectiveapi.com/



Токсичность не распознается



Цель проекта

Создание сервиса, который поможет улучшить бизнес-процесс в коммуникациях за счет сбора, очистки, предобработки, разметки и анализа данных. Проект позволяет фильтровать негативные и токсичные комментарии на трансляциях Twitch, групповых чатах тг или по API.

Рабочий домен - онлайн трансляции Twitch.

Что такое токсичный комментарий?

Токсичные сообщения содержат оскорбительный, негативный контент, который нарушает нормы общения, вызывает дискомфорт у участников. Виды:

1. Разжигание ненависти:
    - Расовая, этническая, религиозная, гомофобная ненависть.
2. Харассмент и угрозы:
    - Запугивание, прямые угрозы физического, эмоционального или социального характера.
    - Унижение, троллинг
3. Несанкционированный контент:
    - Реклама, спам, ссылки на сторонние ресурсы без разрешения.
    - Попытки обойти фильтры или инструменты модерации (AutoMod).

Выходной продукт: пайплан предобработки, сбора, хранения и дообучения модели toxic BERT для мультиязычного использования

Содержание

01

□ĠĢĸĀℓ ĝ ĝ Ĵ Ĩ

Источники данных

1

115 собственных токсичных примеров

Формат: json

2

jigsaw-toxic-comment-train-google-ru-cleaned.csv

Формат: Dataset kaggle

Ссылка

3

jigsaw-toxic-comment-train.csv

Формат: Dataset kaggle

Ссылка

4

jigsaw-unintended-bias-train_ru_clean.csv

Формат: Dataset kaggle

Ссылка

5

russian-language-toxic-comments.csv

Формат: Dataset kaggle

Ссылка

Инструменты сбора данных

1) Собственные данные можно считать синтетически сгенерированными, потому что использовался DeepSeek для генерации списка 57 обидных прилагательных и списка 115 примеров комментариев. Разметка класса токсичности также была сделана автоматически. Результат генерации был проверен вручную

2) Kaggle CLI

Итоги этапа "Сбор данных"

1) Скачано 4 дата сета в формате csv на русском и англ языках
2) Собран собственный набор данных для примеров токсичных сообщений на русском
3) Суммарный объем данных
    1) 2,2 Гб
    2) ~3 млн. записей

Следующая цель работы:

1) Предобработать данные для дообучения toxic BERT

02

Предобратока данных

Схема предобработки

Необходимо привести все данные к виду 8 столбцов : id, comment_text, toxic, severe_toxic, obscence, threat, insult, identity_hate

1

115 собственных токсичных примеров

Формат:
2 столбца (id, коммент)
Действие:
Присвоить комменту одну из 6 меток с помощью detoxify

2

jigsaw-toxic-comment-train-google-ru-cleaned.csv

Формат:
10 столбца (индекс1, индекс, id, коммент, 6 нужных классов)
Действие:
Убрать 2 индекса в датасете, оставить только id

По 3, 4, 5 см. следующий слайд

Атрибут	Смысл
id	Уникальный идентификатор комментария
comment_text	Текст комментария
toxic	Метка, указывающая, является ли комментарий токсичным
severe_toxic	Метка, указывающая, является ли комментарий крайне токсичным
obscence	Метка, указывающая, содержит ли комментарий непристойные выражения
threat	Метка, указывающая, содержит ли комментарий угрозы
insult	Метка, указывающая, содержит ли комментарий оскорбления
identity_hate	Метка, указывающая, содержит ли комментарий ненависть к определённой группе людей

Предобработка данных

3

jigsaw-toxic-comment-train.csv

Формат:
45 столбцов (id, комментарий и подробная разметка данных)
Действие:
Свести данные комментария к 6 меткам

4

jigsaw-unintended-bias-train_ru_clean.csv

Формат:
45 столбцов (id, комментарий и подробная разметка данных)
Действие:
Свести данные комментария к 6 меткам

5

russian-language-toxic-comments.csv

Формат:
2 столбца (id, токсичный комментарий)
Действие:
Присвоить каждому комментарию одну из 6 меток токсичности с помощью detofixy

Анализ данных на наличие пропусков

```
print("\nКоличество пропущенных значений в каждом столбце:")
print(df.isnull().sum())
```

[2] ✓ 0.5s

```
Количество пропущенных значений в каждом столбце:
id               0
comment_text     0
toxic            0
severe_toxic     0
obscene          0
threat           0
insult           0
identity_hate    0
dtype: int64
```

Итог:
Все входные данные не содержали пропусков

Очистка данных

```python
# 1. Проверка текстовых данных на выбросы
def analyze_text_outliers(column):
    # Длина каждого комментария
    df['text_length'] = df[column].apply(len)

    # Статистика по длине текста
    print("\nСтатистика по длине текста:")
    print(df['text_length'].describe())

    # Визуализация распределения длины текста
    import matplotlib.pyplot as plt
    plt.figure(figsize=(10, 6))
    plt.hist(df['text_length'], bins=50, color='blue', edgecolor='black')
    plt.title('Распределение длины текста')
    plt.xlabel('Длина текста')
    plt.ylabel('Количество комментариев')
    plt.show()
```

Анализ данных на наличие выбросов

```
Статистика по длине текста:
count    3.942562e+06
mean     3.107745e+02
std      3.079714e+02
min      1.000000e+00
25%      9.600000e+01
50%      2.070000e+02
75%      4.260000e+02
max      8.095000e+03
Name: text_length, dtype: float64
```

Итог:
Все входные данные были проверены по длине комментария. Стало понятно, что есть комментарии минимум из одного символа и максимум из 8 тыс символов

Очистка данных

Удалить пунктуацию, спец. символы и комментарии короче 2 символов

```
2025-01-14 13:32:18,461 - Removed 263 rows with special characters or numeric-only comments.
2025-01-14 13:32:18,461 - Data shape after removing special/numeric comments: (3886314, 8)
2025-01-14 13:32:20,354 - Removed 1508 rows with short comments (length < 2).
2025-01-14 13:32:20,354 - Data shape after removing short comments: (3884806, 8)
2025-01-14 13:32:22,414 - Data preprocessing completed.
```

```python
# Удаление комментариев, состоящих только из спецсимволов или цифр
initial_count = len(df)
df = df[~df['comment_text'].apply(is_special_or_numeric)]
removed_count = initial_count - len(df)
logging.info(f"Removed {removed_count} rows with special characters or numeric-only comments.")
logging.info(f"Data shape after removing special/numeric comments: {df.shape}")

# Удаление коротких комментариев
initial_count = len(df)
df = df[df['comment_text'].str.len() >= 2]
removed_count = initial_count - len(df)
logging.info(f"Removed {removed_count} rows with short comments (length < 2).")
logging.info(f"Data shape after removing short comments: {df.shape}")
```

Итог:
Данные были очищены от спец. символов, длины меньше 2, пунктуации

╫ĩ Ēġ Ĥ ℓ ќĀ ℓ ĝ ĝ Ĵ Ĩ

Удаление стоп слов из комментариев

Использую библиотеку nltk

```
# Удаление стоп-слов
text = ' '.join([word for word in text.split() if word not in stop_words])
```

Итог:
Данные были очищены от часто встречающихся слов, но не несущей смысловой нагрузки. Например, "и", "в", "на", "с", "по".

Скрипт предобработки данных

После предобработки данных все датасеты были приведены к одинаковой структуре столбцов с помощью скрипта https://github.com/Stepan5024/itmo-ai-stream/blob/main/explore_data_production_jigsaw-toxic-comment-train-google-ru-cleaned.ipynb

Атрибут	Смысл
id	Уникальный идентификатор комментария
comment_text	Текст комментария
toxic	Метка, указывающая, является ли комментарий токсичным
severe_toxic	Метка, указывающая, является ли комментарий крайне токсичным
obscence	Метка, указывающая, содержит ли комментарий непристойные выражения
threat	Метка, указывающая, содержит ли комментарий угрозы
insult	Метка, указывающая, содержит ли комментарий оскорбления
identity_hate	Метка, указывающая, содержит ли комментарий ненависть к определённой группе людей

Разметка данных

Датасет русских комментариев содержал всего лишь текст токсичных комментариев. Без указания класса токсичности.

Задача:
По тексту комментария определить одну из 6 меток toxic / severe_toxic / obscence / threat / insult / identity_hate

Решение:
Использована библиотека https://github.com/unitaryai/detoxify для автоматической классификации.

Процесс:
Была получена вероятность для каждой метки. Порог 0.65 был установлен, чтобы бинарно проставить 1 для метки, иначе метка имела значение 0.

```python
import pandas as pd
import time
from detoxify import Detoxify

# Функция для анализа токсичности с помощью Detoxify
def analyze_toxicity(text):
    try:
        # Загрузка модели Detoxify
        model = Detoxify('original')  # Используем предобученную модель
        results = model.predict(text)  # Анализ текста

        # Преобразуем значения в бинарные (1 или 0) в зависимости от порога 0.65
        binary_results = {key: 1 if value > 0.65 else 0 for key, value in results.items()}

        return binary_results
    except Exception as e:
        print(f"Ошибка при анализе текста: {e}")
        return {
            'toxic': 0,
            'severe_toxic': 0,
            'obscene': 0,
            'threat': 0,
            'insult': 0,
            'identity_hate': 0
        }
```

Удаление дубликатов

Датасет может содержать дубликаты по тексту комментария `comment_text`.

Процесс:
Было удалено 56к дублей из данных

```
2025-01-14 09:13:35,119 - Removed 55985 duplicate rows based on 'comment_text'.
2025-01-14 09:13:35,119 - Data shape after removing duplicates: (3886577, 8)
```

Задача:
Удалить все записи содержащие один и тот же текст

Решение:
Использована библиотека pandas
`df.drop_duplicates(subset='comment_text', keep='first')`.

```python
# Удаление дубликатов по столбцу comment_text
initial_count = len(df)
df = df.drop_duplicates(subset='comment_text', keep='first')
removed_count = initial_count - len(df)
logging.info(f"Removed {removed_count} duplicate rows based on 'comment_text'.")
logging.info(f"Data shape after removing duplicates: {df.shape}")
```

Лемматизация текста

Приведение текста `comment_text` к нормальной форме.

Задача:
Каждое слово должно быть приведено к начальной форме

Решение:
Использована библиотека natasha. Каждое слово проходит токенезацию, морфологический анализ, лемматизацию

Процесс:
Было лемматизировано 3 млн 850 тыс строк

```python
def preprocess_text(text):
    global processed_rows_counter  # Используем глобальный счетчик

    # Увеличиваем счетчик строк
    processed_rows_counter += 1

    # Печать в консоль каждые 50 000 строк
    if processed_rows_counter % 50000 == 0:
        print(f"Лемматизировано {processed_rows_counter} строк.")

    # Приведение к нижнему регистру
    text = text.lower()

    # Удаление пунктуации
    text = text.translate(str.maketrans('', '', string.punctuation))

    # Удаление стоп-слов
    text = ' '.join([word for word in text.split() if word not in stop_words])

    # Лемматизация с использованием Natasha
    doc = Doc(text)
    doc.segment(segmenter)  # Токенизация
    doc.tag_morph(morph_tagger)  # Морфологический анализ

    # Извлечение лемм
    for token in doc.tokens:
        token.lemmatize(morph_vocab)  # Лемматизация каждого токена
    lemmas = [token.lemma for token in doc.tokens]
    text = ' '.join(lemmas)

    return text
```

Результат предобработки данных

Каждый из файлов датасетов был соединен в один csv файл представляющий датасет для дообучения с помощью скрипта https://github.com/Stepan5024/itmo-ai-stream/blob/main/connect_datasets.ipynb

Файл: final_combined_toxicity_data.csv

```python
# Загрузка данных
df1 = pd.read_csv('jigsaw-toxic-comment-train.csv')
df2 = pd.read_csv('jigsaw-toxic-comment-train-google-ru-cleaned.csv')
df3 = pd.read_csv('jigsaw-unintended-bias-train_ru_clean.csv')
df4 = pd.read_csv('russian-language-toxic-comments.csv')

# Приведение столбцов к единому формату
# Для первого датасета
df1['toxic'] = df1[['severe_toxicity', 'obscene', 'threat', 'insult', 'identity_attack']].max(axis=1)
df1['severe_toxic'] = df1['severe_toxicity']
df1['obscene'] = df1['obscene']
df1['threat'] = df1['threat']
df1['insult'] = df1['insult']
df1['identity_hate'] = df1['identity_attack']

# Для второго датасета
df2['toxic'] = df2['toxic']
df2['severe_toxic'] = df2['severe_toxic']
df2['obscene'] = df2['obscene']
df2['threat'] = df2['threat']
df2['insult'] = df2['insult']
df2['identity_hate'] = df2['identity_hate']

# Для третьего датасета
df3['toxic'] = df3[['severe_toxicity', 'obscene', 'threat', 'insult', 'identity_attack']].max(axis=1)
df3['severe_toxic'] = df3['severe_toxicity']
df3['obscene'] = df3['obscene']
df3['threat'] = df3['threat']
df3['insult'] = df3['insult']
df3['identity_hate'] = df3['identity_attack']

# Выбор нужных столбцов
df1 = df1[['id', 'comment_text', 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']]
df2 = df2[['id', 'comment_text', 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']]
df3 = df3[['id', 'comment_text', 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']]
```

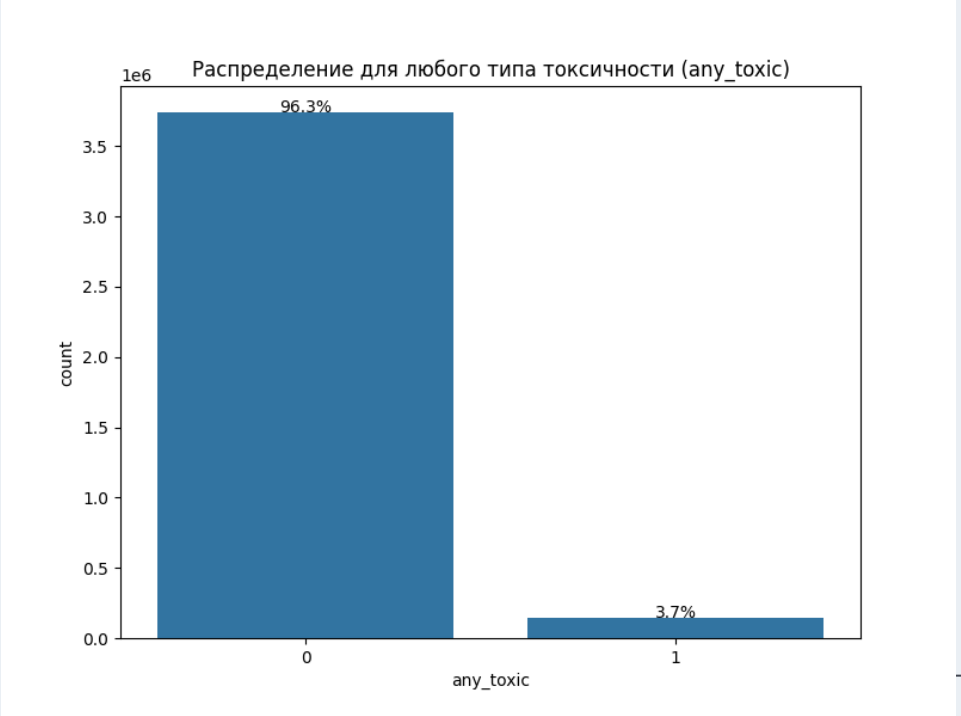
03

╣ģģėāĀĠ▪ ℓĤāėĮģĕĒēќ ℓĝℓėĒċќĀℓĝĝĴ Ĩ

Задачи исследования данных

1. Оценить сбалансированность меток токсичных и нетоксичных комментариев
2. Оценить распределение длины комментариев
3. Оценить распределение меток среди 6 классов
4. Просмотр облака слов нетоксичных комментариев
5. Просмотр облака слов токсичных комментариев
6. Оценка количества меток на один комментарий
7. Оценить распределение мультиязычности (русский и английский)

Сбалансированность меток токсичных и нетоксичных комментариев

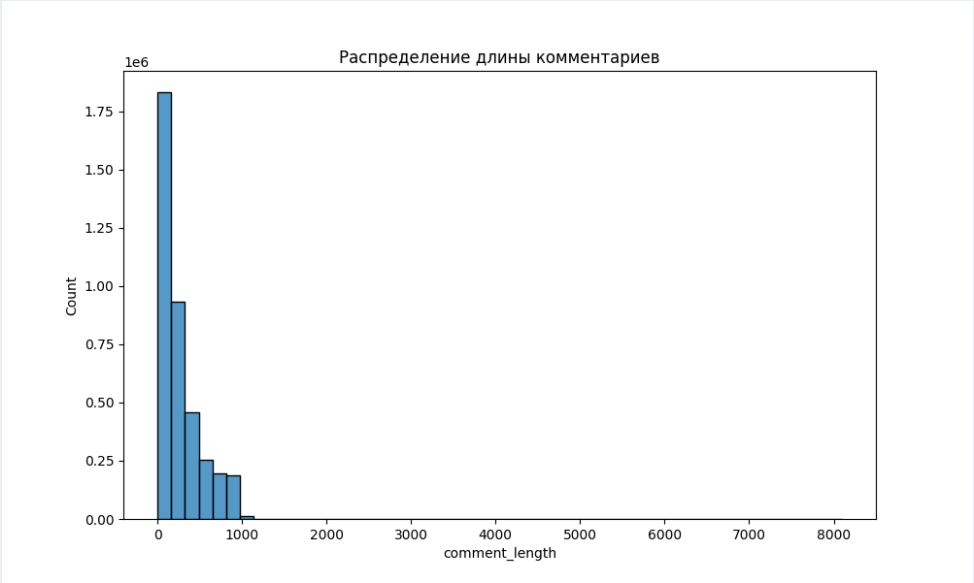


В собранном наборе данных из 3 млн. комментариев 96% принадлежат к нетоксичным сообщениям

Вывод:
В процессе обучения могут быть сложности предсказывания токсичных комментариев

Решение:
Для обучения взять 400 тыс. токсичных комментариев и срез 1 млн. нетоксичных комментариев.

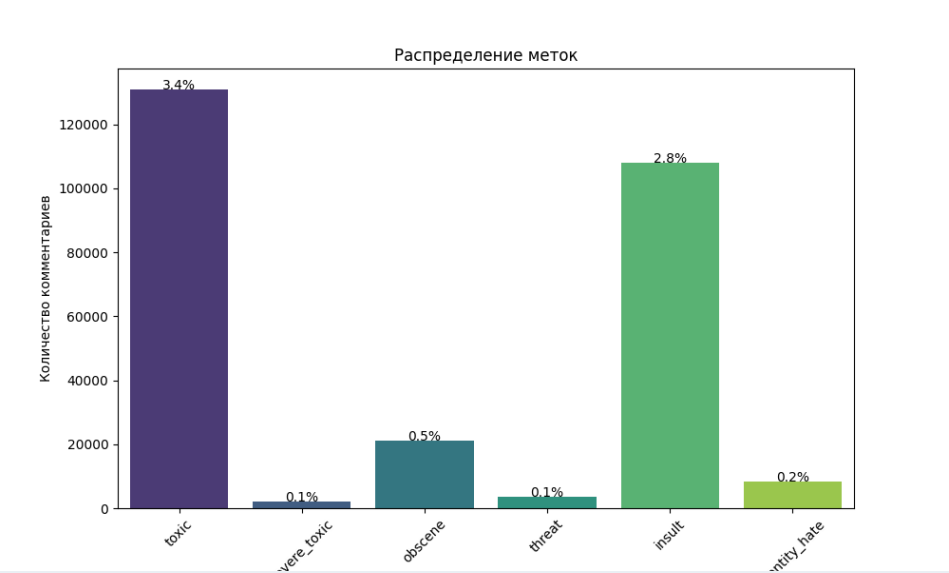
╪Ī ā ĝ ě ℓ ќĀė Ēĝ ĝ Ĵ ѣ ĠĜĜā ĝ Ĥℓ Ģ Ēā ▪



В собранном наборе данных длина составляет от 2 символов до 8к

Вывод:
Большая часть данных находится размером до 1тыс символов. Это корректно, оставить слишком длинные комментарии в 8 раз. Они предоставляют больший контекст обработки моделью

Распределение меток среди 6 классов

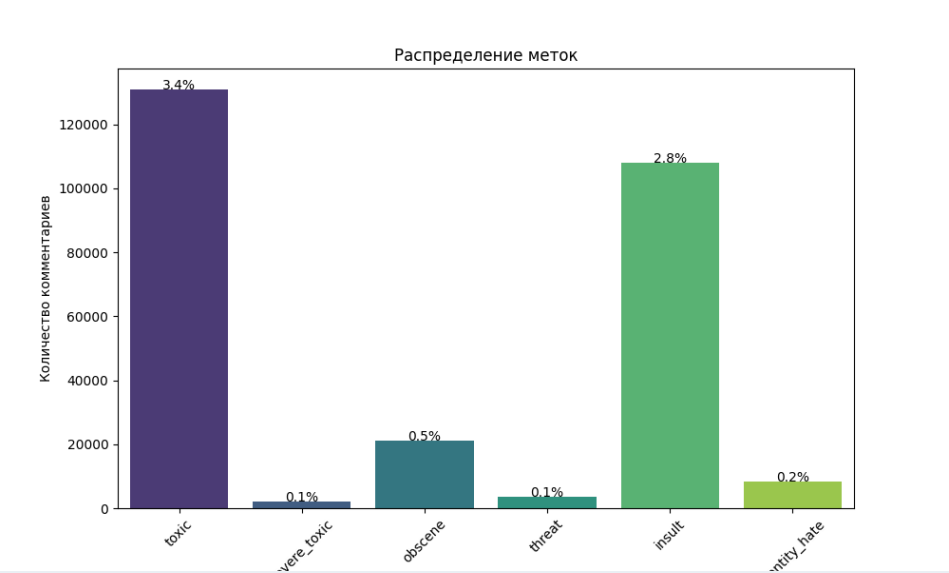


Основная часть токсичных данных связана с классом toxic и threat

Вывод:
По результатам обучения предсказания классов toxic и threat могут быть более вероятны по сравнению с другими классами.

Решение:
На продакшене важно получить сам факт токсичности, принадлежность к метке задача очень субъективная

Распределение меток среди 6 классов



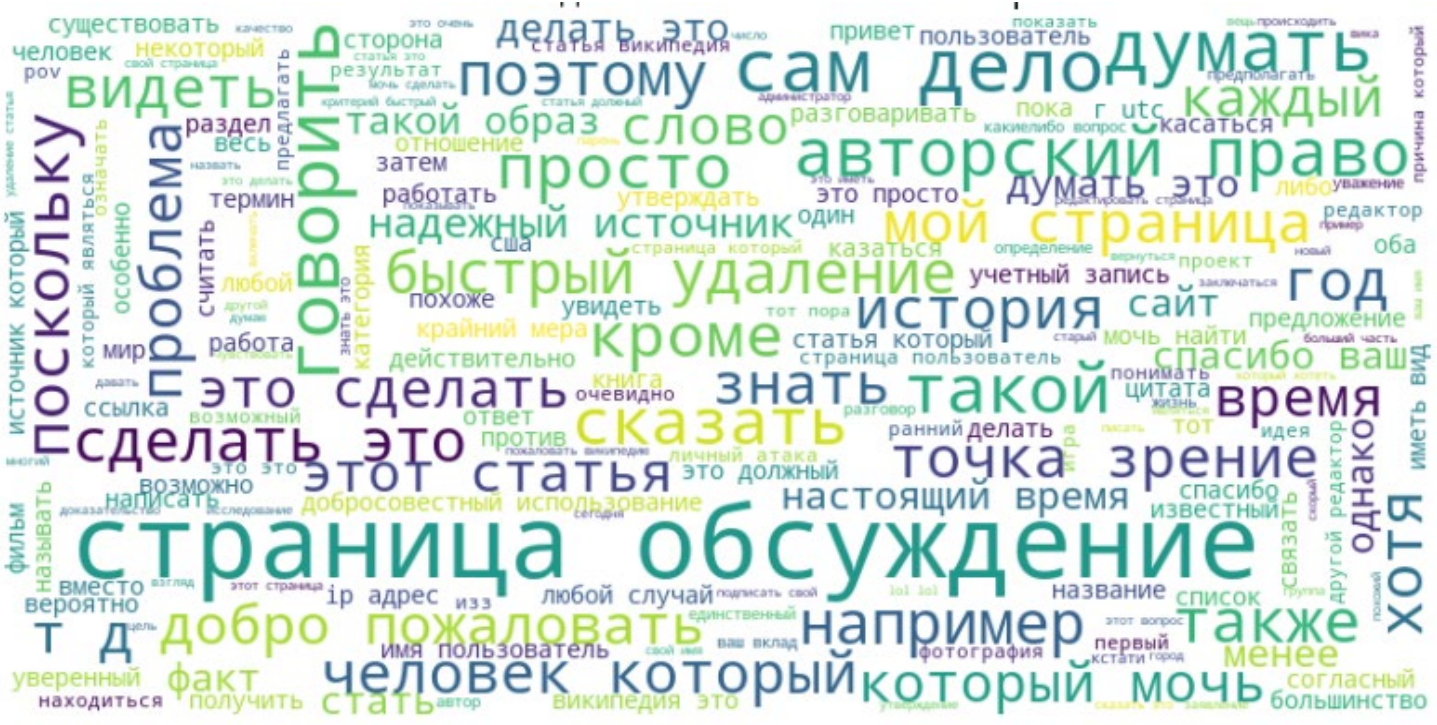
Основная часть токсичных данных связана с классом toxic и threat

Вывод:
По результатам обучения предсказания классов toxic и threat могут быть более вероятны по сравнению с другими классами.

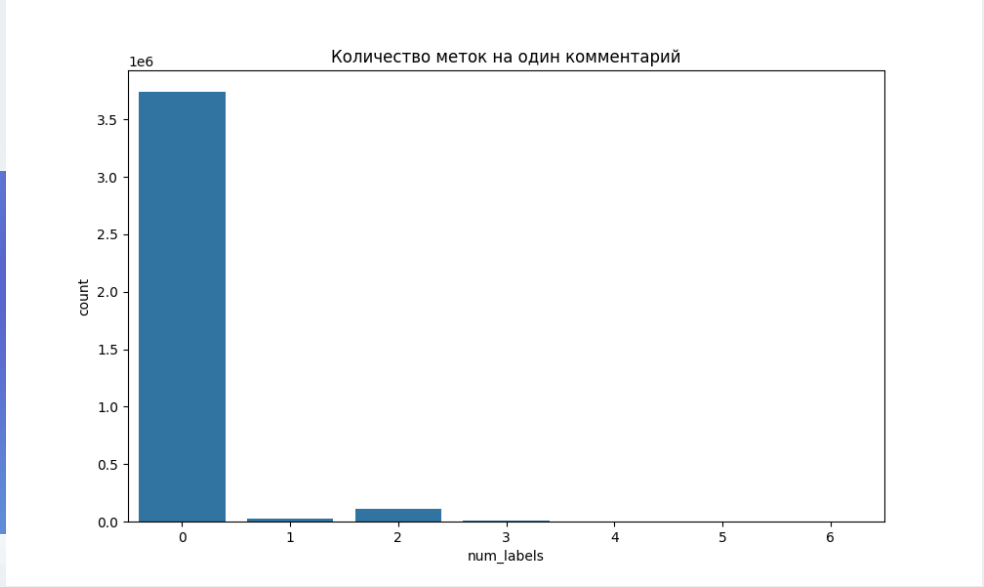
Решение:
На продакшене важно получить сам факт токсичности, принадлежность к метке задача очень субъективная

Облако слов для токсичных комментариев

Облако слов для не токсичных комментариев



Оценка количества меток на один комментарий



Основная часть токсичных комментариев имеет одну метку

Вывод:
Наличие двух, а то и 6 классов одновременно не является «плохими» данными из-за сложной субъективной оценки эмоций

╫Ī ā ĝ ě ℓ ŕ Ģ ℓ ģ ġ Ģ ā Ā ā ė ā ĝ Ē Ō ŕ Ĝ ĥ ė Į Ĥ Ē Ō ċ Ĵ ĩ ĝ Ġ ģ Ĥ Ē

Данные по разным языкам распределены +- равномерно

Вывод:
Обучение для русского и английского языка будет сбалансированным

04

Определение и обоснование метрик качества данных

Метрики качества данных

Бизнес задача:

1) Поддерживать хранение и добавление новых комментариев в БД
2) Иметь таблицу с данными готовыми для обучения toxic BERT

Метрики:

1) Доля комментариев с нестандартными символами
2) Доля комментариев с неподдерживаемыми языками
3) Доля пропущенных токсичных комментариев
4) Доля ложных срабатываний

┘Ġė ŌĩẽĠĜĜā ĝĤℓĢĒā▪ ŧĝŧĝā ģĤℓĝĀℓĢĤĝĴ ĜĒŧĝĒĜ▪ Ġė ℓĜĒ

Негативные примеры:

- Эмодзи (😀, 💙).
- Специальные символы (@, #, $, %, &).
- HTML-теги (
,).
- Повторяющиеся символы ("!!!", "???").
- Нечитаемые символы (например, "").

Почему это важно?

- Шум в данных.
- Искажение смысла.
- Проблемы с токенизацией.
- Влияние на классификацию.

Алгоритм проверки качества данных по этой метрике:

Использую регулярное выражение для поиска нестандартных символов в поступающей строке. После добавления новых данных пересчитывается доля содержания не стандартных символов

Доля комментариев с неподдерживаемыми языками

Негативные примеры:

- Языки не поддерживаемые библиотекой. Сейчас доступны русский, английский, французский, испанский и еще 5 языков. Не корректные: Kaixo, nola zaude, 重写代码片段.
- Смешанные языки("Hello, 你好, cómo estás").
- Нечитаемые символы (например, "").

Почему это важно?

- Шум в данных.
- Модель, обученная на определённых языках, может некорректно обрабатывать текст на неподдерживаемых языках, что приводит к ошибкам классификации.Проблемы с токенизацией.
- Текст на неподдерживаемых языках сложно анализировать и интерпретировать,

Алгоритм проверки качества данных по этой метрике:

Использую библиотеку langdetect для определения языка каждого комментария. Далее фильтрация через список поддерживаемых языков. Вычисление доли попавших в БД текстов на не поддерживаемых языках

Доля пропущенных токсичных комментариев

Измеряет, сколько токсичных комментариев не было обнаружено модель. И модератор пометил комментарий вручную как токсичный.

Почему это важно?

- Качество авто модерации могло упасть.
- Упала репутация сервиса.
- Упала удовлетворенность потребителей,

Алгоритм расчета метрики:

Вычисление доли пропущенных токсичных комментариев относительно всех токсичных комментариев.

Доля ложных срабатываний

Измеряет, сколько нетоксичных комментариев были ошибочно помечены как токсичные моделью и стримеру пришлось снять пометку о токсичности.

Почему это важно?

- Качество авто модерации могло упасть.
- Упала репутация сервиса.
- Упала удовлетворенность потребителей,

Алгоритм расчета метрики:

Вычисление доли ложных срабатываний токсичных комментариев относительно всех токсичных комментариев.

µς

■ℓċĢℓ□ĠĤĕℓќ□ℓċĴ ќ

Āℓĝĝ Ĵ Ĩ ќĀėŌĨĢℓĝāĝĒŌќ

Āℓĝĝ Ĵ Ĩ

Требования к БД

1) Хранение данных, пригодных для дообучения модели
2) Сохранения данных полученных в режиме реального времени со стрима

Выбор хранилища – SQLite

Причины:
1) Минимальная настройка
2) Легковесность
3) Работа с реляционными данными (триггеры, таблицы)
4) Бесплатное использование

Таблицы БД

Таблица: cleaned_comments
Цель: Хранение сырой информации комментариев со стримов.

#	Column	Dtype
0	id	object
1	comment_text	text
2	toxic	int64
3	severe_toxic	int64
4	obscene	int64
5	threat	int64
6	insult	int64
7	identity_hate	int64
8	stream_id	object
9	is_wrong_classificated	bool
10	data_created	datetime
11	data_modified	datetime

Каждая таблица имеет триггер, для обновления полей date_created и date_modified

Таблица: streams
Цель: Хранение информации о стримах.

#	Column	Dtype
0	id	text
1	start_time	datetime
2	end_time	datetime
3	chanel_name	text
4	data_created	datetime
5	data_modified	datetime

Таблица: row_comments
Цель: хранение пригодных данных для дообучения модели

#	Column	Dtype
0	id	object
1	comment_text	object
2	toxic	int64
3	severe_toxic	int64
4	obscene	int64
5	threat	int64
6	insult	int64
7	identity_hate	int64
8	data_created	datetime
9	data_modified	datetime

Поток данных

В случае, если модератор убрал признак токсичности или наоборот проставил его в таблице **row_comments,** то проставляется признак ошибочного распознавания is_wrong_classificated = 1 и актуализируется запись в **cleaned_comments**

06

формление пунктов 1,2 и 5 в отдельный пайплайн для автоматизации

Задачи автоматизации

1) Бизнес-задача обогащение данных для дообучения модели toxic bert

Ряд технических задач:
1) Подключение к стримам Twitch
2) Сбор данных в реальном времени в таблицу row_comments
3) Предобработка комментария
4) Классификация токсичности, языка
5) Запись чистого комментария в таблицу cleaned_comments
6) Удаление обработанной записи из таблицы row_comments

Скрипты автоматизации собраны в папке application
https://github.com/Stepan5024/itmo-ai-stream/tree/main/application

╬ĠĀĕėŋ ĩāĝĒāŗĕģĤĢĒĜℓĜõ⌊ẑűfl Š

Была использована библиотека `twitchio`

Сервис позволяет разместить в конфигурации список каналов, на которых будут отслеживаться комментарии.

В текущей версии в конфигурации заложено использование канала автора работы stepycdragon и двух тестовых `leekbeats и wipr.`

```python
    async def event_ready(self):
        print(f'Бот подключен как | {self.nick}')
        print(f'ID пользователя | {self.user_id}')

    async def event_message(self, message):
        print(f'[{message.channel.name}] {message.author.name}: {message.content}')

        # Сохраняем сообщение в таблицу row_comments
        save_message(message)

        # Обрабатываем команды
        await self.handle_commands(message)

    @commands.command()
    async def hello(self, ctx: commands.Context):
        await ctx.send(f'Привет, {ctx.author.name}!')
```

Автоматическое устранение расхождения в данных

В ходе работы приложения могут быть случаи записи «грязных» данных в таблицу cleaned_comments. Чтобы гарантировать сбор пригодных для дообучения данных, необходимо автоматизировать процесс проверки и очистки данных.

Для этого написан скрипт, отрабатывающий каждые три часа
https://github.com/Stepan5024/itmo-ai-stream/blob/main/application/check_bad_data.py

Главная функция приложения

```python
from twitchio.ext import commands
from db_utils import save_message, update_cleaned_comments
from process_message import preprocess_text, is_toxic
from config import TOKEN, INITIAL_CHANNELS

class Bot(commands.Bot):
    def __init__(self):
        super().__init__(
            token=TOKEN,
            prefix='?',
            initial_channels=INITIAL_CHANNELS
        )

    async def event_ready(self):
        print(f'Бот подключен как | {self.nick}')
        print(f'ID пользователя | {self.user_id}')

    async def event_message(self, message):
        print(f'[{message.channel.name}] {message.author.name}: {message.content}')

        # Сохраняем сообщение в таблицу row_comments
        save_message(message)

        # Обрабатываем команды
        await self.handle_commands(message)

    @commands.command()
    async def hello(self, ctx: commands.Context):
        await ctx.send(f'Привет, {ctx.author.name}!')

# Запуск бота
if __name__ == "__main__":
    bot = Bot()
    bot.run()
```

```python
Base = declarative_base()

class Stream(Base):
    __tablename__ = 'streams'
    id = Column(String, primary_key=True)
    start_time = Column(DateTime)
    end_time = Column(DateTime)
    chanel_name = Column(String)

class RowComment(Base):
    __tablename__ = 'row_comments'
    id = Column(String, primary_key=True)
    comment_text = Column(String)
    toxic = Column(Integer)
    severe_toxic = Column(Integer)
    obscene = Column(Integer)
    threat = Column(Integer)
    insult = Column(Integer)
    identity_hate = Column(Integer)

class CleanedComment(Base):
    __tablename__ = 'cleaned_comments'
    id = Column(String, primary_key=True)
    comment_text = Column(String)
    toxic = Column(Integer)
    severe_toxic = Column(Integer)
    obscene = Column(Integer)
    threat = Column(Integer)
    insult = Column(Integer)
    identity_hate = Column(Integer)
    stream_id = Column(String)
    is_wrong_classificated = Column(Boolean)
```

Модель данных

07

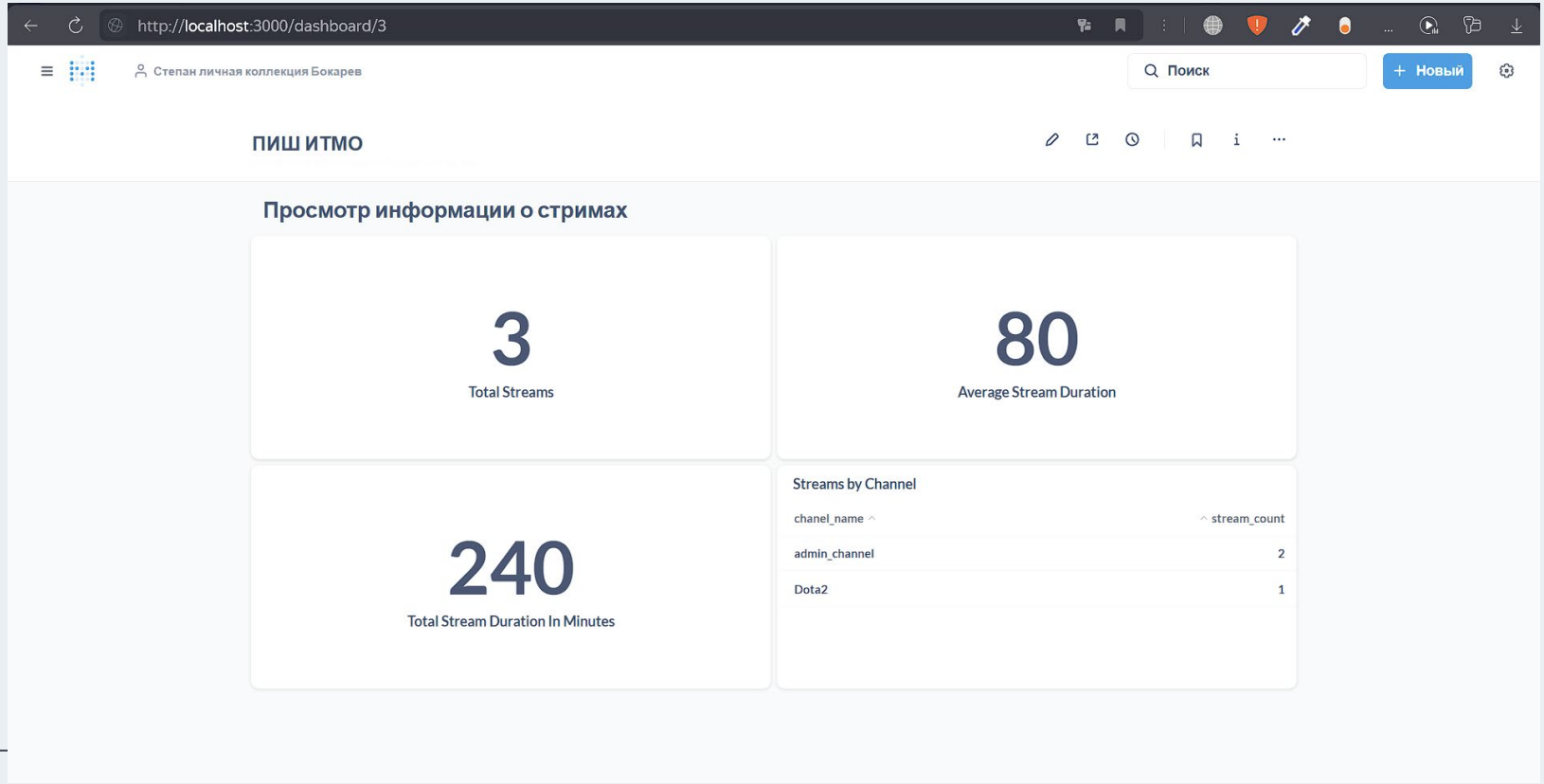
╠ħ ĠĢĜė ā ĝ Ēā ќ
ℓ ī □ĠĢĀℓ

Задачи дашборда

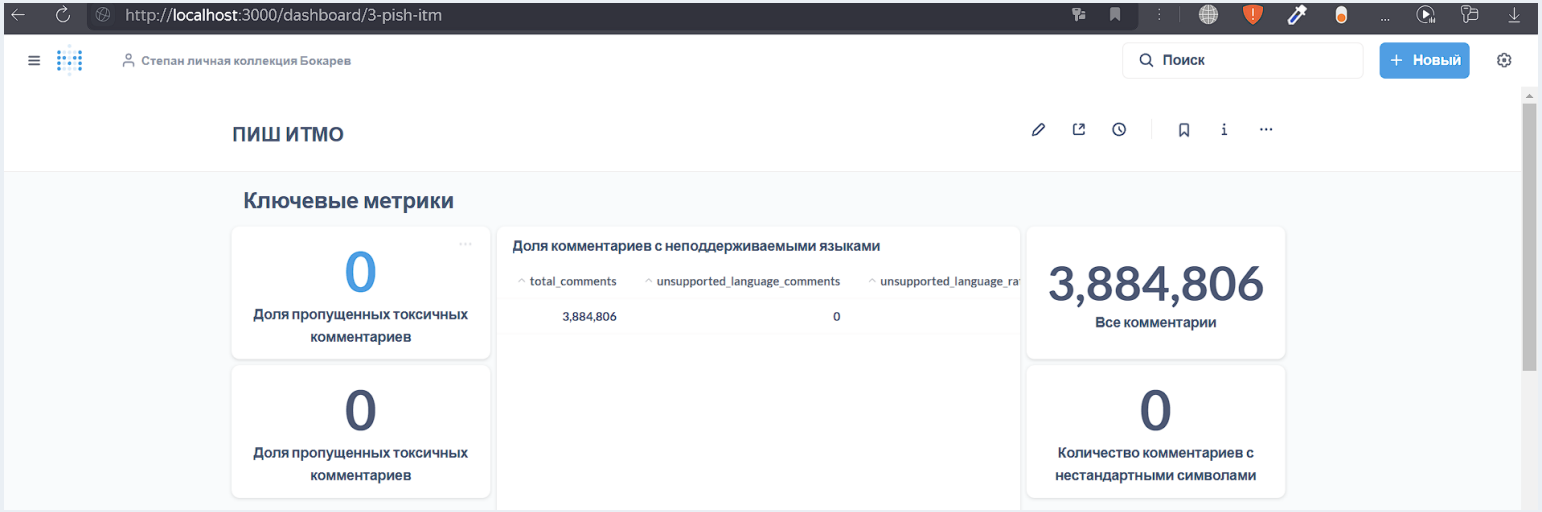
1) Отобразить ключевые показатели хранения и обработки данных
2) Визуализировать процесс хранения данных
3) Давать отчет о состоянии качества данных

Выбран инструмент **metabase.com**
Он развернут на локальном порту и предоставляет мощные возможности BI.

Таблица по стримам



Ключевые метрики



μψ
āċĥėĮĤℓĤℐ

Результаты разработки

В бизнес домене онлайн трансляций Twich разработан инструмент для сбора, предобработки, очистки, анализа комментариев на любых трансляциях.
1.Текущая работа позволила сделать пайплайн обработки данных
Этапы, которые проходят данные:
1. Соединение датасетов.
2. Извлечение из онлайн трансляций.
3. После сбора данные проходят этап очистки, где устраняются ошибки, пропуски и выбросы. Для этого используются инструменты на Python.
4. Далее предобработка данных.
5. После этого данные размечаются для задач классификации, что позволяет выявить токсичные комментарии, определить языковые предпочтения аудитории.
6.Подключена система BI визуализации метрик качества данных в metabase.com
7.Решение содержит процедуры автоматической проверки качества данных и устранения проблем в хранилище

Дальнейшие планы проекта

Текущий инструмент позволяет проводить ETL процесс работы с комментариями. Для дальнейшего развития проекта будет произведено дообучение модели toxic bert. Это позволит увеличить точность распознавания токсичных и негативных комментариев на русском языке по сравнению с существующими решениями

Спасибо за внимани е

Контакты для связи тг @StepanBokarev