Сычева Татьяна
tg: @Owl_toto

WB winter school

Репутация пользователей: задача классификации ненормативной лексики

1. Постановка задачи

Ваше решение должно определить, содержит ли текст нецензурные выражения, и выставить соответствующий бинарный флаг:

1 — в тексте есть мат.
0 — в тексте нет мата.
Вам предстоит использовать набор данных с текстовыми комментариями и отзывами, где каждый текст уже помечен флагом о наличии или отсутствии мата. Ваша задача — построить модель, которая сможет эффективно решать эту задачу на новых, ранее невиданных данных.

Пример:

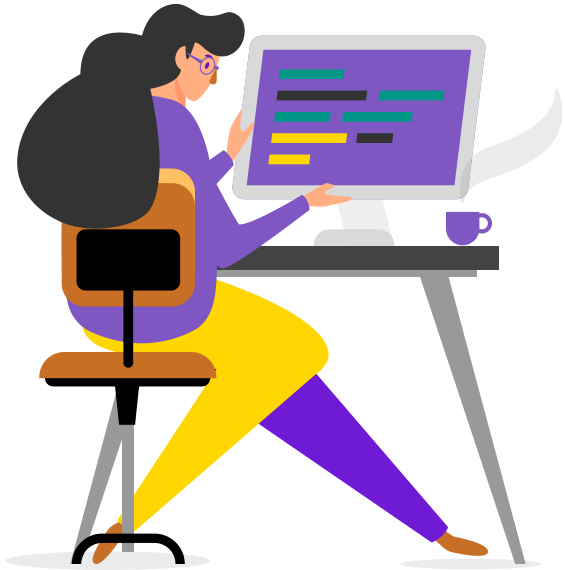
Оригинальный текст: "Эта юбка полное г..."
Ожидаемый флаг: 1 (есть мат).
Оригинальный текст: "Эта юбка мне не понравилась."
Ожидаемый флаг: 0 (мата нет).
Основная метрика

Решение будет оцениваться с использованием метрики F1-score. Эта метрика учитывает как точность, так и полноту предсказаний, что особенно важно в задачах с несбалансированными классами (когда, например, количество текстов с матом может быть меньше, чем без него).

Задачи:

Разработать модель для бинарной классификации текстов, которая предсказывает наличие мата.
Оптимизировать модель с фокусом на повышении метрики F1-score.
Обработать различные виды нецензурных выражений, включая завуалированные и сокращенные формы.
Ожидаемый результат
Модель должна точно предсказывать, содержит ли текст нецензурные выражения, позволяя онлайн-платформам быстро выявлять потенциально неприемлемый контент.

Пайплайн задачи



01 Постановка задачи

02 Описание данных

03 Предобработка данных

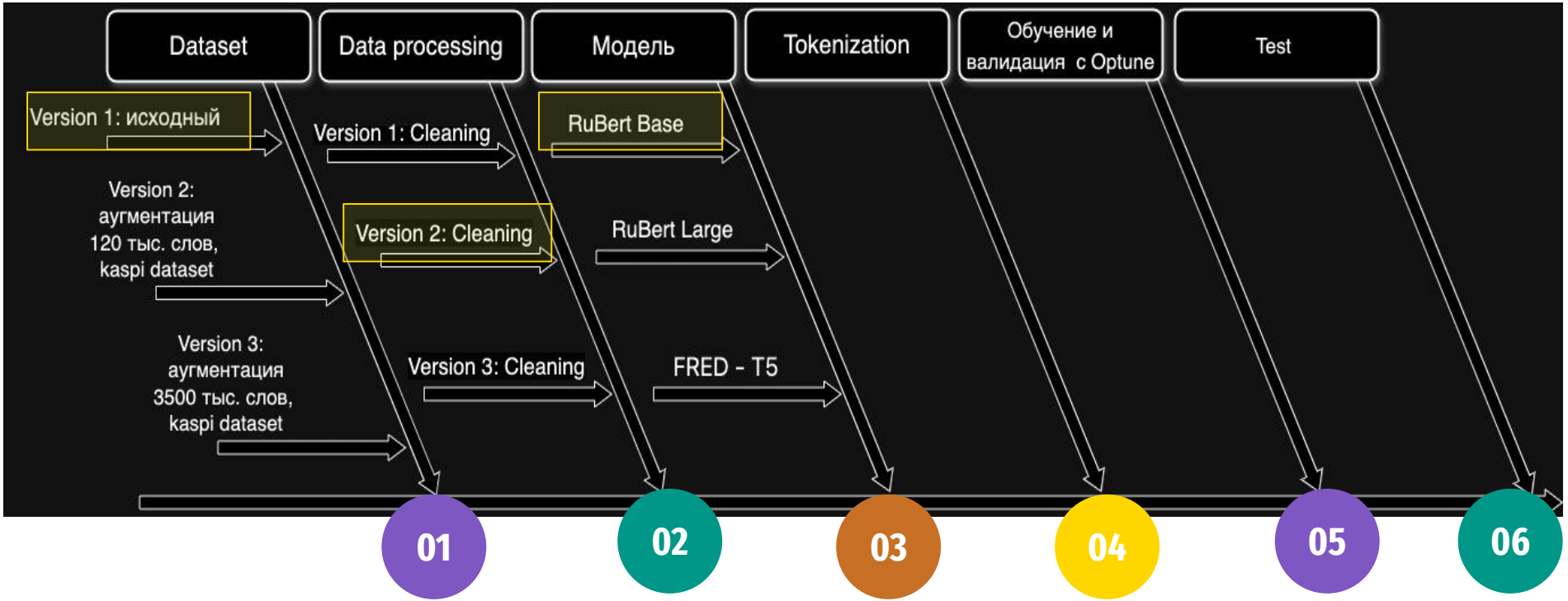
04 Разделение данных

05 Токенизация

06 Обучение модели

07 Валидация и тестирование

08 Fine-tuning модели


Dataset
Data processing
Модель
Tokenization
Обучение и валидация с Optune
Test
Version 1: исходный
Version 1: Cleaning
RuBert Base
Version 2: аугментация 120 тыс. слов, kaspi dataset
Version 2: Cleaning
RuBert Large
Version 3: аугментация 3500 тыс. слов, kaspi dataset
Version 3: Cleaning
FRED - T5
01
02
03
04
05
06

Очистка данных | *regex*

- lowercase
- замена нескольких пробелов на одинарный
- замена некоторых букв, например 'ё' на 'е'
- удаление эмодзи

Веса для классов

```
class_counts = train_d['label'].value_counts()
class_weights = [len(train_d) / class_counts[i] for i in range(len(class_counts))]
class_weights = torch.tensor(class_weights).to(device).float()
```

```
criterion = CrossEntropyLoss(weight=class_weights) # определяем функцию потерь с учетом весов классов
```

Выбор модели

Word embedding

FastText

- легкая модель
- быстро обучается
- работает с n-gramm, что хорошо для завуалированных слов

- низкий скор
- не работает с контекстом

Contextual embeddings

ruBert - base

- сравнительно небольшая и быстро обучается (по сравнению с другими моделями трансформеров)
- достаточно простая
- поддерживает контекст за счет Bert архитектуры

- меньше параметров, чем у больших моделей

ruBERT - large

- ресурсоемкая модель

FRED - T5

- оочень ресурсоемкая модель

Токенизация:

AutoTokenizer

Гиперпараметры:

Количество скрытых слоёв	**12**
Количество голов внимания для каждого уровня внимания	**12**
Dropout	**0,1**
Optimizer	**AdamW**
Learning rate	**6.6875e-06**
β 1 (AdamW)	**0.9**
β 2 (AdamW)	**0.999**
Epsilon	**1e-8**
Scheduler	**get_linear_schedule_with_warmup**
Warmup Steps:	**170**
Loss Function	**CrossEntropyLoss(weight=class_weights)**
Training Epochs	**3**
Batch Size	**8**

Результаты

- Валидационный F1-score: `0.980`
- Test F1 Score: `0.981`
- Test F1 Score Leaderboard : `0.921`

Спасибо за внимание