levheimcube

masterkristall

July 2021

1 Введение

В данной работе предпринята попытка применения модели куба эмоций Лёвхейма в задаче анализа тональности текста. Куб эмоций Лёвхейма (рисунок 1) - теоретическая модель объяснения отношений между уровнем нейромедиаторов и испытываемой эмоцией. Модель была предложена в работе 2012 года Хуго Лёвхеймом.

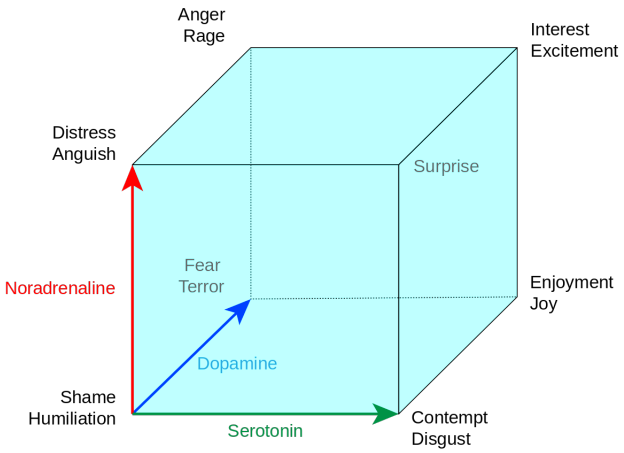

Рис. 1: Куб эмоций Лёвхейма

В современных системах автоматического определения эмоциональной оценки текста чаще всего используется одномерное эмотивное пространство: позитив или негатив (хорошо или плохо). Использование куба Лёвхейма позволяет перейти к многомерному пространству и свести задачу определения тональности к задаче регрессии для каждой из шкал эмоций:

- disgust-rage (отвращение-ярость)
- enjoyment-distress (удовольствие-страдание)

- fear-surprise (страх-удивление)
- shame-excitement (стыд-возбуждение)

2 Описание моделей

Для решения задачи регрессии расматривались следующие модели:

- Bert base multilingual cased - многоязычная модель BERT, предварительно обученная на 104 языкахс использованием маскового моделирования языка (Masked-Language Modeling). Модель чувствительна к регистру.
- DeepPavlov rubert base cased - модель, обученная на русскоязычной части Википедии и новостных данных. В качестве инициализации для RuBERT используется многоязычная версия BERT-base.
- XGBoostRegressor - модель регрессии на основе XGBoost. XGBoost - это реализация алгоритма градиентного бустинга деревьев, широко известного своей эффективностью и точностью прогнозирования.

3 Эксперименты

3.1 Setup лучших моделей

3.1.1 Bert base multilingual cased

- epochs = 5
- seed = 2021
- max length = 200
- batch size = 8

3.1.2 DeepPavlov rubert base cased

- epochs = 10
- seed = 2021
- max length = 200
- batch size = 8

Регрессионный модуль имеет вид:

```
class RegressionBert(torch.nn.Module):
    def __init__(self, freeze_bert=True):
        super(RegressionBert, self).__init__()
        self.bert = AutoModel.from_pretrained(BERT, num_labels=1)
        if freeze_bert:
            for p in self.bert.parameters():
                p.requires_grad = False
        self.fc0 = torch.nn.Linear(768, 768)
        self.fc1 = torch.nn.Linear(768, 1)
        self.dropout = torch.nn.Dropout(0.25)

    def forward(self, x, att=None):
        x = self.bert(x, attention_mask=att)[0]
        x = x[:, 0, :]
        x = self.fc0(x)
        x = self.fc1(x)
        x = x.flatten()
        return x
```

Введение дополнительных слоев в регрессионный модуль (Linear, Dropout) ухудшает качество на тестовой выборке.

3.1.3 XGBoostRegressor

Гиперпараметры получены с использованием hyperopt - библиотеки для последовательной и параллельной оптимизации.

- colsample_bytree = 0.61,
- eta = 0.04
- gamma = 0.13
- max_depth = 5.0
- min_child_weight = 3.0
- n_estimators = 106.0
- subsample = 0.86

disgust-rage	enjoyment-distress	fear-surprise	shame-excitement
2.252	2.763	2.345	2.498

Таблица 1: Значения кроссвалидационной метрики для xgboost

3.2 Результаты экспериментов

Модель	disgust-rage	enjoyment-distress	fear-surprise	shame-excitement
DeepPavlov rubert	**2.206**	**2.563**	**2.195**	**2.335**
Bert base	2.251	2.788	2.290	2.400
XGBoostRegressor	2.264	2.759	2.247	2.495

Таблица 2: Значения метрик на валидационной выборке

По таблице 2 видно, что лучшие значения на тестовой выборке показывает модель DeepPavlov rubert.

3.3 DeepPavlov rubert - детали обучения

На рисунках 2, 3, 4, 5 отражены кривые обучения/валидации и сравнение распределения предсказанных значений с исходным.

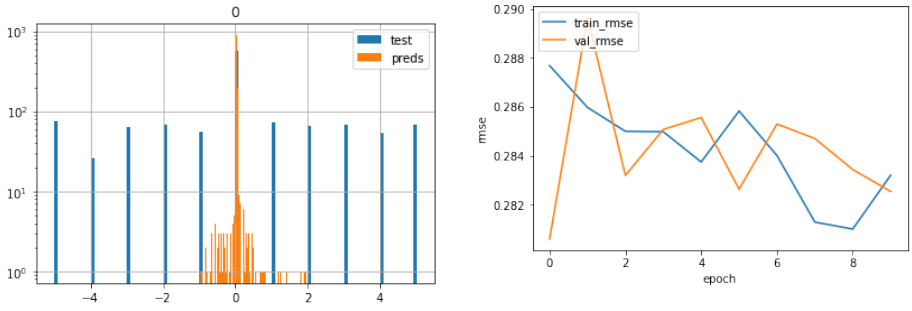

Рис. 2: Распределение предсказанных значений и кривые обучения для шкалы disgust-rage

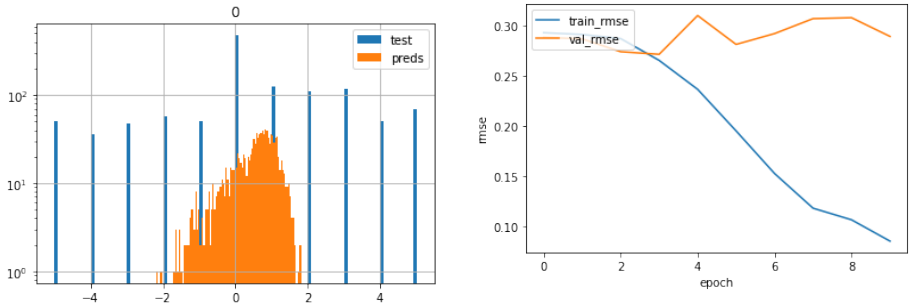

Рис. 3: Распределение предсказанных значений и кривые обучения для шкалы fear-surprise

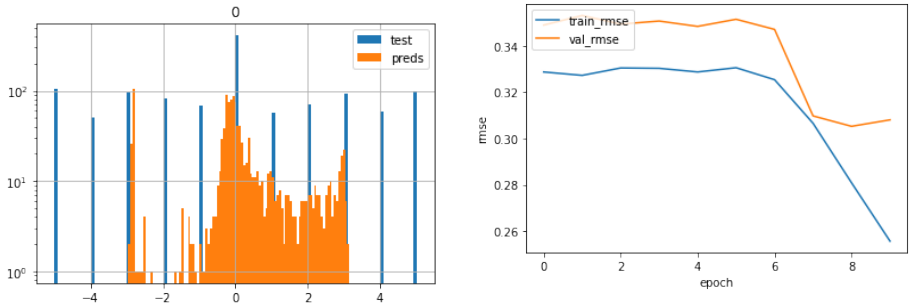

Рис. 4: Распределение предсказанных значений и кривые обучения для шкалы shame-excitement

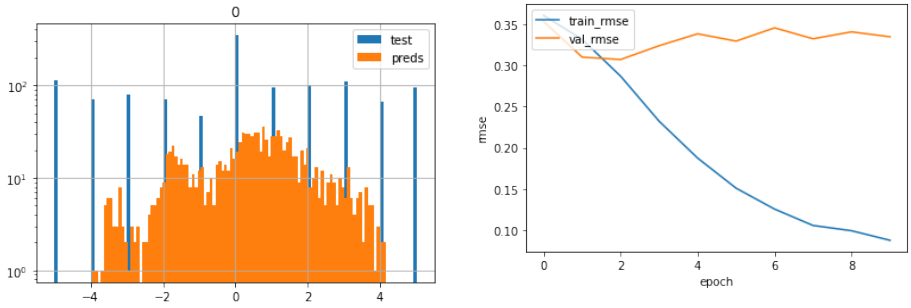

Рис. 5: Распределение предсказанных значений и кривые обучения для шкалы enjoyment-distress

4 Выводы

- Наилучшее качество показывает модель DeepPavlov rubert base cased.
- Модели довольно быстро начинают переобучаться, что препятствует улучшению метрик за счет увеличения количества эпох обучения или слоев в архитектуре.
- Модель градиентного бустинга показывает относительно неплохие результаты с учетом небольшого времени, необзодимого для обучения (30.7 секунд), а также имеет небольшое смещение относительно результатов кроссвалидации при обучении.