Отчет

Решение задачи классификации отзывов к фильмам

Выполнил Глазков. Н. А

Цель работы:

- Обучение модели для классификации отзывов к фильмам
- Создание приложения на Django

Задачи:

- Выбрать оптимальную модель для обучения на датасете Large Movie Review Dataset v1.0
- Обучить модель
- Создать приложение Django для классификации отзывов и развернуть его в открытом доступе

Датасет:

В качестве датасета представлен Movie Review Dataset v1.0. Он содержит позитивные и негативные отзывы (12500 позитивных и 12500 негативных для train и test составляющих) и словарь.

Модель:
В качестве модели была избрана ELECTRA. Используем предобученную модель. В датасете отсутствуют оценки 5 и 6, так что обучаем модель для классификации 8 классов

```
tokenizer = ElectraTokenizer.from_pretrained('data/tokinizer')

model = ElectraForSequenceClassification.from_pretrained('data/tokinizer', num_labels=8)
model.load_state_dict(torch.load('data/frozen2.pth', weights_only=True))
```

Создаем датасет pytroch:

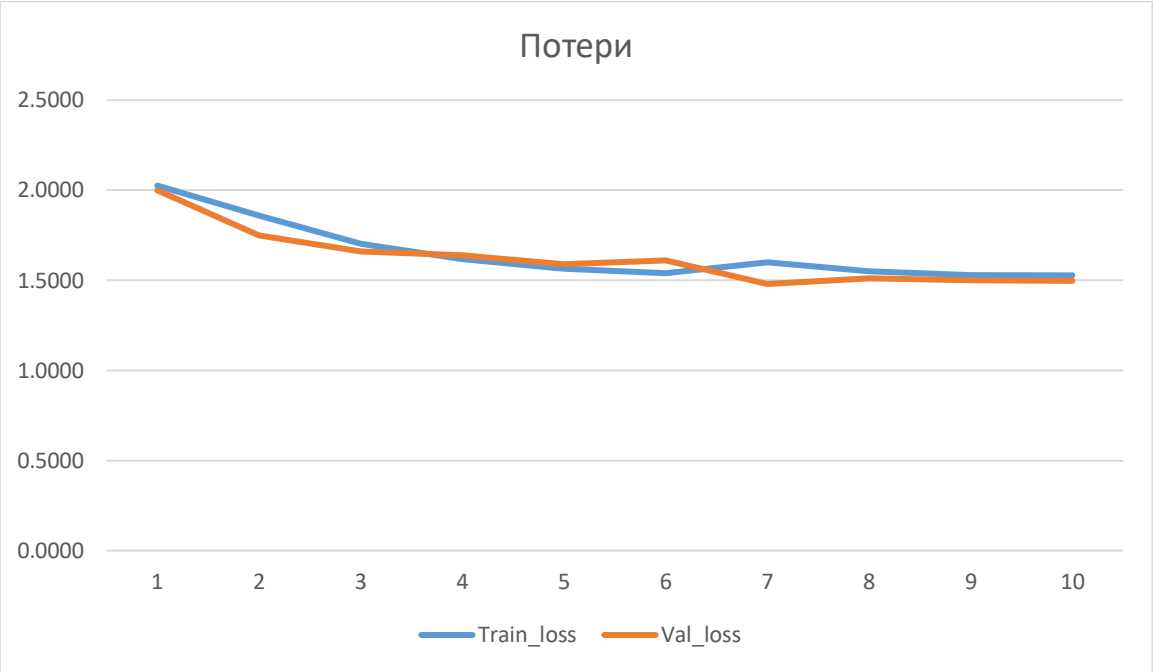
```
class TextDataset(Dataset):
    def __init__(self, data_dir, tokenizer, max_length=512):
        self.data_dir = data_dir
        self.tokenizer = tokenizer
        self.max_length = max_length
        self.file_paths = [os.path.join(data_dir, f) for f in os.listdir(data_dir) if f.endswith('.txt')]

    def __len__(self):
        return len(self.file_paths)

    def __getitem__(self, idx):
        file_path = self.file_paths[idx]
        with open(file_path, 'r', encoding='utf-8') as file:
            text = file.read()

        label = int(os.path.basename(file_path).split('_')[-1].split('.')[0])

        encoding = self.tokenizer(
            text,
            padding='max_length',
            truncation=True,
            max_length=self.max_length,
            return_tensors='pt'
        )

        classes = {1:0, 2:1, 3:2, 4:3, 7:4, 8:5, 9:6, 10:7} #label -> class idx

        input_ids = encoding['input_ids'].squeeze()
        attention_mask = encoding['attention_mask'].squeeze()
        return {
            'text' : text,
            'input_ids': input_ids,
            'attention_mask': attention_mask,
            'label': torch.tensor(classes[label], dtype=torch.long)
        }
```

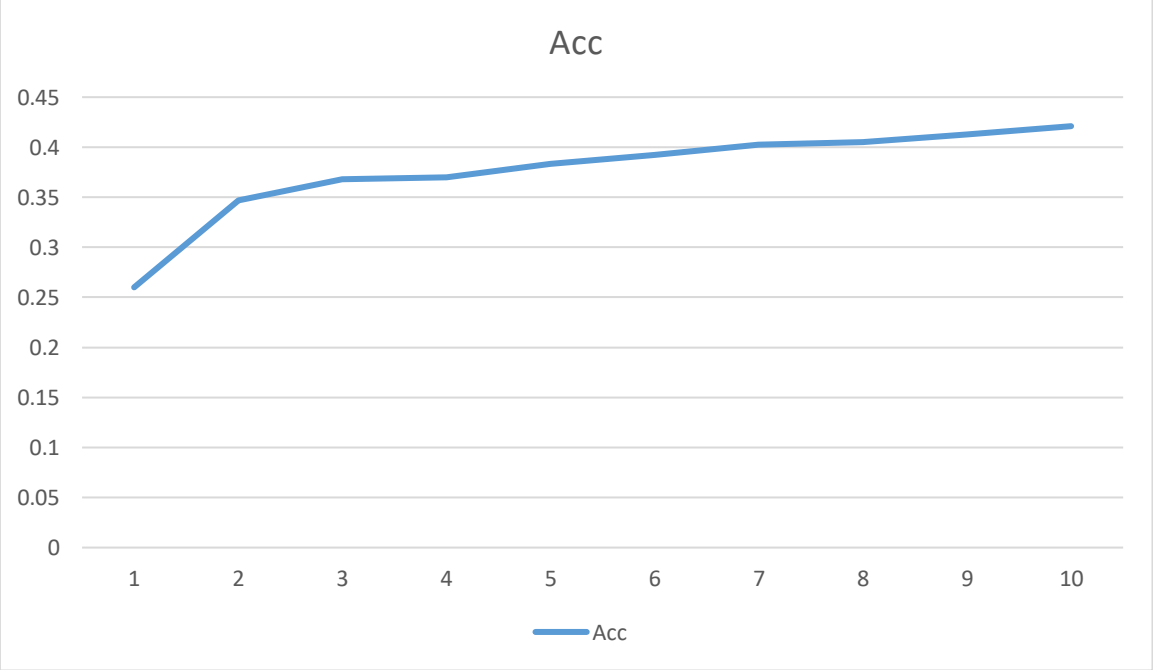
Загружаем DataLoader:

```
train_loader = DataLoader(train_dataset, sampler=sampler, batch_size=32)
test_loader = DataLoader(test_dataset, shuffle=False, batch_size=32)
```

Используем семплер чтобы нивелировать несбалансированность датасета

В качестве функции потери используем CrossEntropyLoss т.к. она хорошо подходит для многоклассовой классификации. В качестве оптимизатора используем AdamW. На валидации проходимся по всему тестовому датасету, метрика – Accuracy. Делим количество верных ответов на общее количество экземпляров в датасете.


Потери
2.5000
2.0000
1.5000
1.0000
0.5000
0.0000
1
2
3
4
5
6
7
8
9
10
Train_loss
Val_loss


Acc
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1
2
3
4
5
6
7
8
9
10
Acc

Выводы:

- Модель не была дообучена из-за нехватки времени и ресурсов
- Была достигнута точность ~40%
- Приложение Django было развернуто в общем доступе