Отчёт Чайникова Константина для компании Тинькофф 21.06.21

Задача:

1) Сделать классификатор интентов

- Для обучения использовать датасет https://github.com/PolyAI-LDN/task-specific-datasets/banking_data/
- Сделать небольшой отчет в свободной форме для оценки качества решения (tf-idf baseline на тесте выбивает от 0.9 f1-score если не получится улучшить - не критично, но выше бейзлайна будет большим плюсом)
- Цель: написать более менее адекватную архитектуру, построить читаемый отчет
- * Реализовать Self-adjusting Dice Loss из https://www.aclweb.org/anthology/2020.acl-main.45.pdf Сравнить с Cross Entropy
- * Реализовать механизм семплинга батчей, чтобы компенсировать несбалансированность классов в датасете Сравнить с обычным семплингом

2) Обернуть классификатор в REST сервис

- Метод POST /classify
- На вход подается текст примера
- В ответ возвращается строковый тег интента
- Сервис должен быть завернут в docker контейнер

Ссылка на ноутбук в гугл колаб: (заранее извиняюсь за ошибки в тексте)

https://colab.research.google.com/drive/1b_eMPqw5lsbXYV8YHIYuDsx39kO9EAWi?usp=sharing

Часть 1 Анализ данных.

Первое. Я нашёл оригинальный датасет тестовой выборки с лэйблами. Поэтому будем смотреть на нём валидировться, а на тренировочном обучаться

Посмотрим, что из себя представляю данные.

```
1 df_train.sample(3)
```

	text	category
1850	Is there a fee to make a transfer?	top_up_by_bank_transfer_charge
2177	I want to take back my transaction	cancel_transfer
2361	Why did I not get the amount of money I reques...	wrong_amount_of_cash_received

Пропусков нет

```
1 df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10003 entries, 0 to 10002
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   text      10003 non-null  object
 1   category  10003 non-null  object
dtypes: object(2)
memory usage: 156.4+ KB
```

77 классов.

Количество классов с тестовом и тренировачном наборе

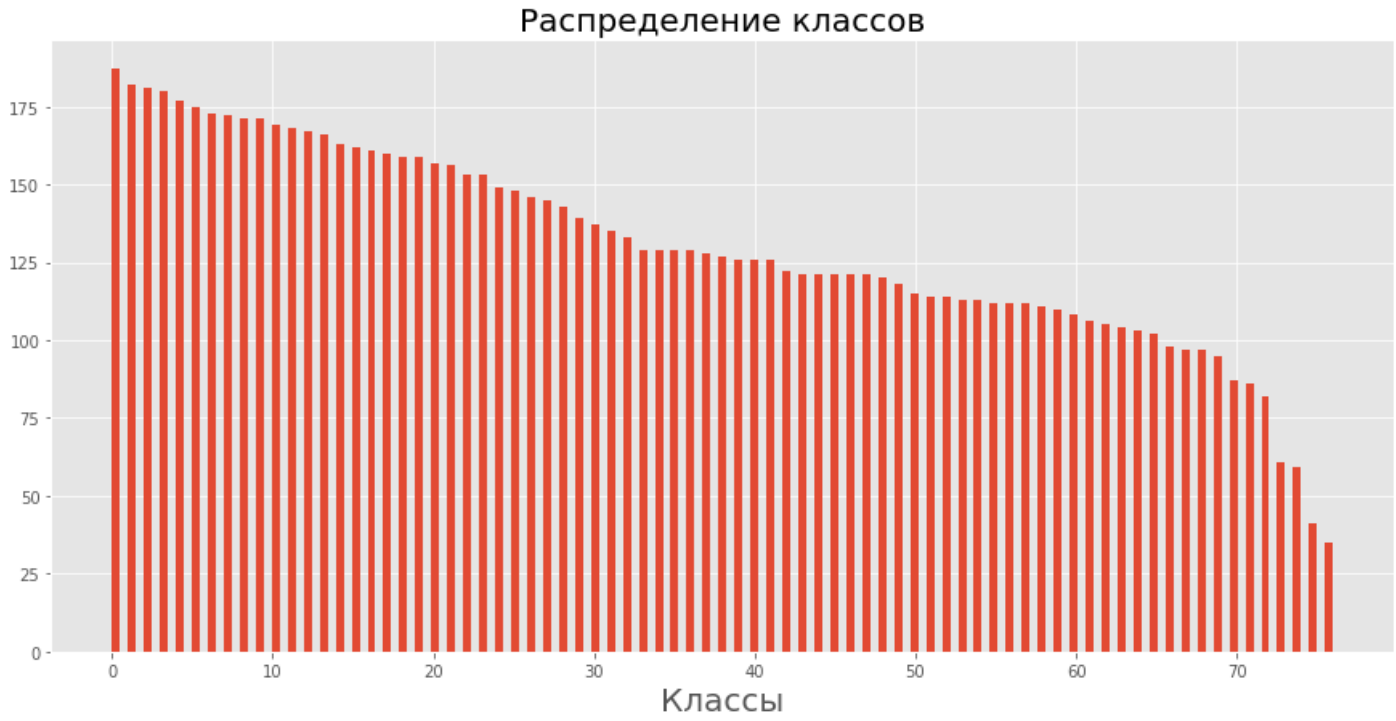
```
[9]  1 df_train.category.nunique()
```

```
77
```

```
[10]  1 df_test.category.nunique()
```

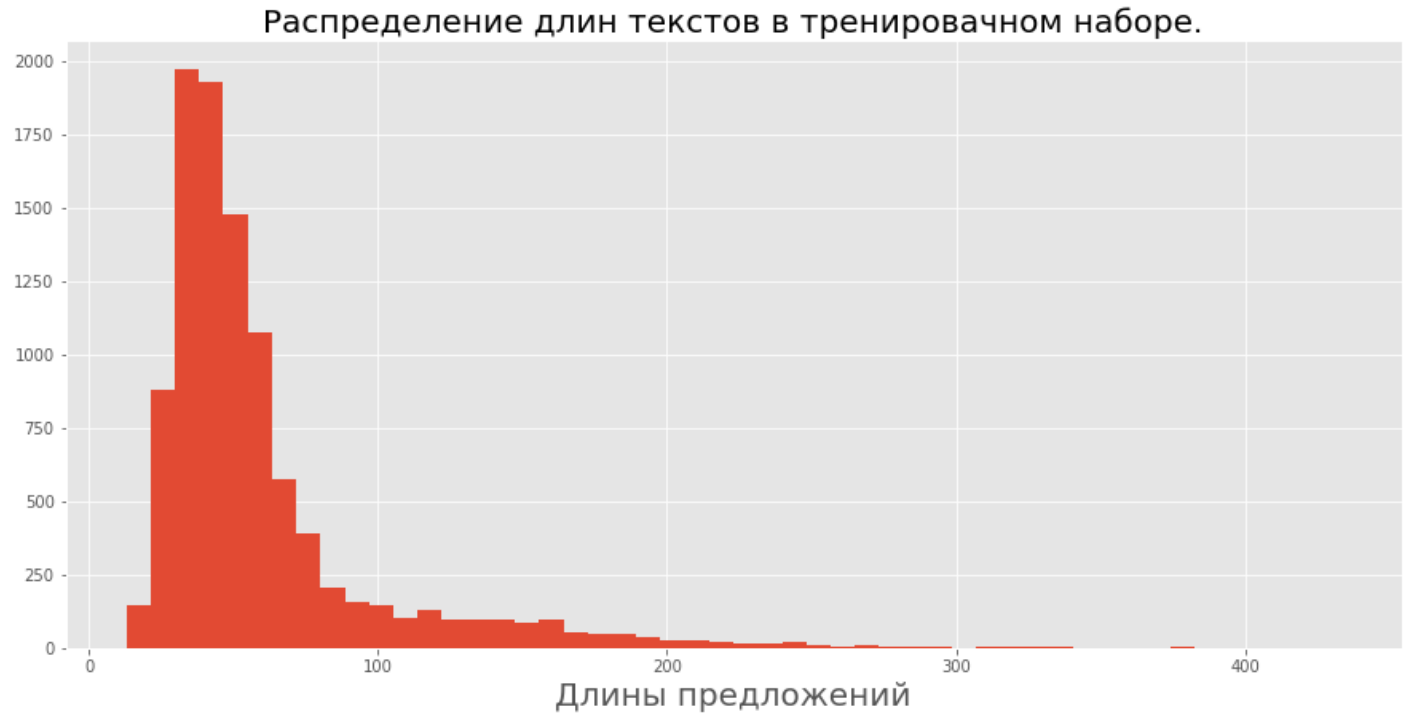
```
77
```

Распределение классов.



Видим дисбаланс.

Распределение длин предложений



Распределение длин слов в предложениях

Также был проведен анализ частотности слов и частотности символов. Подробности в ноутбуке по ссылке выше.

Часть 2 Baseline Классический ML

В данной части рассматривалась возможности предобработки данных (tf-idf и counter) и их влияние на качество на классических алгоритмах таких как Логистическая Регрессия и Случайный лес.

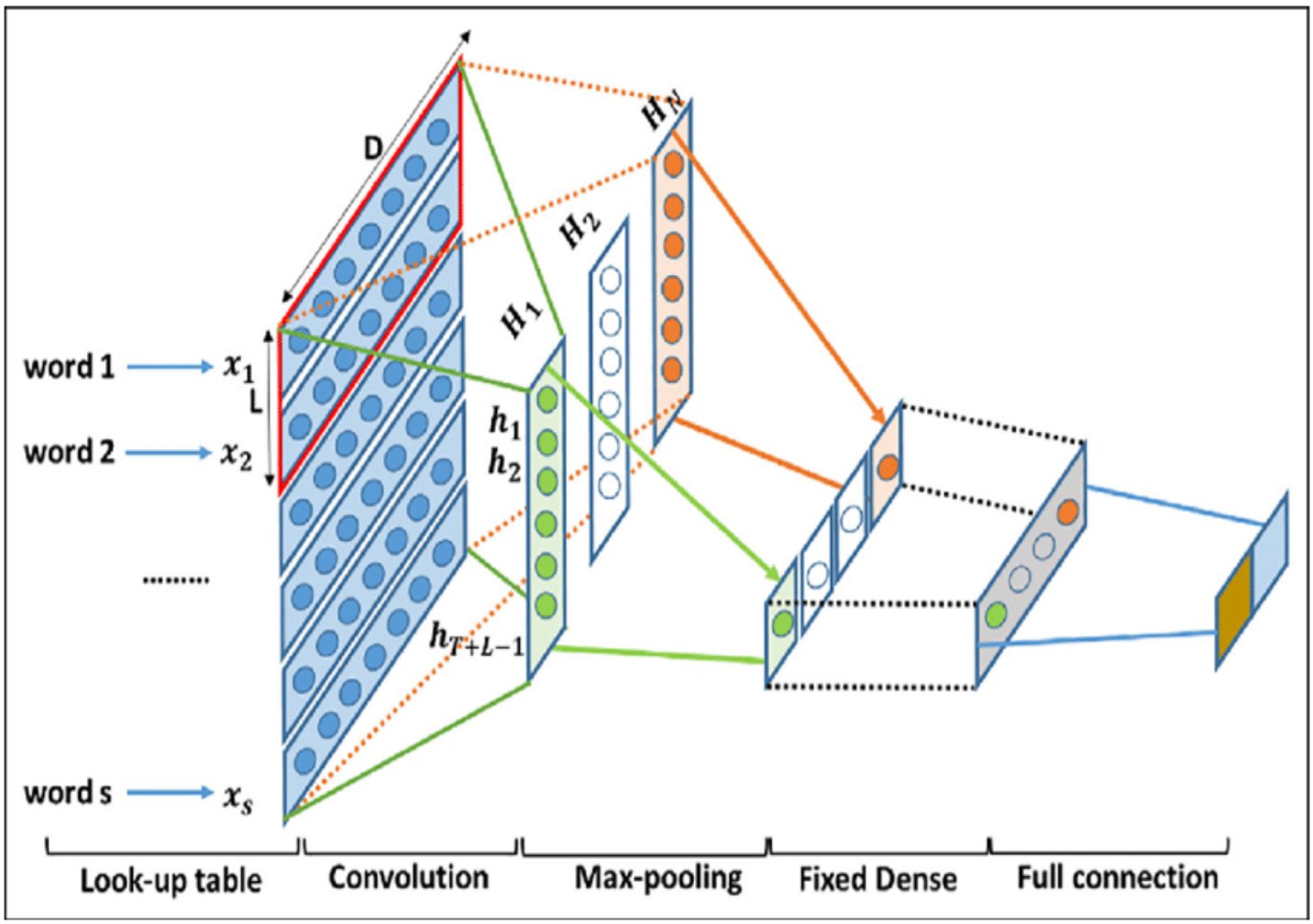
Наилучший результат показала предобработка автокорректировщиком и регулярными выражениями и снова автокорректировщиком, достигнув 0.89 F1 на валидации.

```
Tfidf
F1 для логрегрессии    : 0.879228
F1 для случайного леса : 0.862191
---
Count
F1 для логрегрессии    : 0.890145
F1 для случайного леса : 0.861347
```

Часть 3 Нейронки

Я попытался решить задачу типичной архитектурой для классификации текстов, которая помогала мне в прошлом. (Картинка ниже). Плюс я добавил LSTM для того чтобы в эмбедингах токенов была информация о соседних токенах. Токенами выступали символы из тренировочного набора.

Однако результаты оказались неутешительными сетка медленно училась и в не вошедших экспериментах достигла максимум в 0.4 F1. Плохой результат.



Часть 4 Bert

Когда мало данных возьмём предобученную сеть и дообучим под свою задачу.

В данном случае это берт. Провёл 4 эксперимента. Порядок обучения во всех один и тот же.

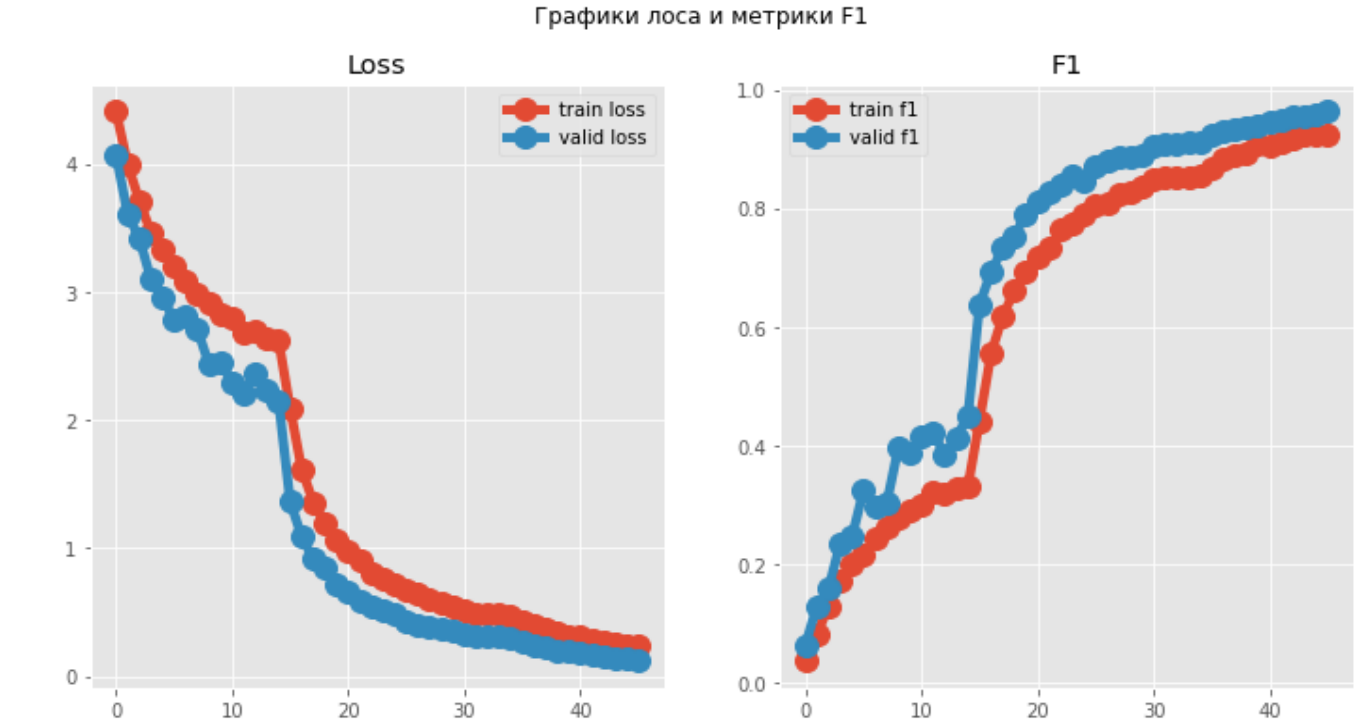
1. Сначала замораживаем веса берта и обучаем классификатор.
2. Когда классификатор выходит на плато, размораживаем веса берта и обучаем дальше.
3. Повторить 1 раз пункт 1 и 2

Были рассмотрены 4 эксперимента.

1. Кроссэнтропия с весами для компенсации дисбаланса классов
2. Кроссэнтропия без весов
3. Балансированное сэмплирование
4. Лосс из статьи.

Лучшим оказался первый эксперимент, графики обучения вы можете увидеть ниже.

```
Лучшее значение F1 на трейне: 0.9243 | на валидации: 0.9627
```



Лосс из статьи

Отдельно хочу остановиться на лоссе из статьи. Моя реализация не заставила модель учиться.

Я нагуглил, что этот лосс уже реализован и реализован сотрудником Тинькофф (Привет Ивану Фурсову от тинькофф поколения). Ссылка ниже.

https://github.com/fursovia/self-adj-dice/blob/4d70d87a05afa154d1002acebd95848f833db342/README.md

Я сравнил с матфорулами из статьи и вроде как они идентичны.

НО. Модель совершенно не хочет на нём учиться.

REST сервис

Для запуска выполните следующие команды.

docker build -t service

sudo docker run -p 5000:5000 service

sudo curl --header "Content-Type: application/json" --data '{"text" : "How do I find the exchange rate?"}' http://127.0.0.1:5000/classify

Итоги

Итого берт с лосом кросс энтропии с весами показал наилучшие результаты, превзойдя даже классические модели ML. Моя реализация лосса из статьи не работает. Реализован Rest сервис.

Возможные улучшения:

- Можно попробовать дальше поэкспериментировать с комбинированием подходов, например взять в самом начале балансированное сэмплирование и обычным лоссом, потом вернуться к обычному сэплмированию, но взять лосс с весами.
- Взять берт побольше или дольше обучить этот берт
- Возможно стоит попробовать обучить сеть на tfidf/count кодировке предложений.
- Реализовать Labelsmoothing