Отчёт

«Разработка Web-приложения – классификатора обзоров на фильмы на базе фреймворка Django»

Отчёт выполнен согласно тестовому заданию, которое было предоставлено компанией Greenatom (https://greenatom.ru/).

Подготовил:

Маношин Андрей Андреевич

2020

Введение.

Серия заданий, о по поводу выполнения которых был написан этот отчёт, заключалась в:

1. Обучить модель на языке Python для классификации отзывов.

2. Разработать веб-сервис на базе фреймворка Django для ввода отзыва о фильме с автоматическим присвоением рейтинга (от 1 до 10) и статуса комментария (положительный или отрицательный).

3. Развернуть сервис в открытом доступе для оценки работоспособности прототипа.

4. Подготовить отчет о работе с оценкой точности полученного результата на тестовой выборке.

5. Отправить ответным письмом ссылку на прототип сервиса, ссылку на открытый репозиторий github с исходным кодом проекта, отчет о проделанной работе в формате pdf.

Для обучения классификатора был предоставлен датасет, составленный из 50 000 отзывов к различным фильмам. Отзывы делятся на положительные (рейтинг больше семи) и отрицательные (не больше четырёх) для упрощения обучения. И тех и других поровну, так что неравенство классов отсутсвует.

Построение модели и обучение

У меня был достаточно большой выбор архитектур нейросети. Выбирал я между архитектурами CNN, RNN (LSTM в частности) и BERT. Были отобраны именно эти архитектуры потому, что они могут учитывать контекст и относительно легко и дёшего обучаемы. К сожалению, самую, на мой взгляд, лучшую по соотношению цена-качество BERT обучить из-за технических возможностей (мало памяти) было затруднительно. (Если это будет необходимо, я могу устранить проблему железа в самом ближайшем будущем). Поэтому я остановился на архитектуре **LSTM** из-за свойственной всем рекуррентным сетям последовательной обработки данных, хотя CNN тоже достаточно привлекательна собственной возможностью учитывать контекст и наименьшей из списка сложностью обучения.

Модель Bag of Words, предполагаемую авторами датасета, я рассматривать не стал. Моей целью было построить нейросеть, способную хоть как-то учитывать контекст. Ещё нужно отметить, что я решил не пользоваться разбиением 50/50 (тренировка/тест), изначально присутствовавшим в наборе данных, так как мне кажется такое разбиение малоэффективным и иногда может быть подвержено bias ошибкам при тестировании на реальных данных из-за отсутствия валидационного датасета. Я остановился на разбиении 80/10/10 тренировка/валидация/тест.

После того, как сеть была выбрана, мной была построена относительно простая и неглубокая двухслойная модель. Точнее, были выбраны парметры $num\_layers = 2$ и $hidden\_size = 8$. Размер эмбеддинга был выбран равным 32.

При выборе времени обучения я остановился на 10-ти эпохах, так как где-то в районе этого момента модель начинала переобучаться и потери валидационной выборки начинали расти. Тренировка занимала в среднем 4 – 5 часов.

```
Epoch: 10/10 Step: 7300 Training Loss: 0.2030 Validation Loss: 0.1920
Epoch: 10/10 Step: 7400 Training Loss: 0.0349 Validation Loss: 0.0169
Epoch: 10/10 Step: 7500 Training Loss: 0.0576 Validation Loss: 0.0209
Epoch: 10/10 Step: 7600 Training Loss: 0.1251 Validation Loss: 0.1894
Epoch: 10/10 Step: 7700 Training Loss: 0.1126 Validation Loss: 0.1372
Epoch: 10/10 Step: 7800 Training Loss: 0.0558 Validation Loss: 0.0462
Epoch: 10/10 Step: 7900 Training Loss: 0.2247 Validation Loss: 0.2045
Epoch: 10/10 Step: 8000 Training Loss: 0.1495 Validation Loss: 0.1619
```

По результатам обучения была достигнута точность (метрика accuracy) на тестовой выборке равная **85%**. При такой неглубокой сети, как мне кажется, это хороший результат.

```
Test Loss: 0.5193
Test Accuracy: 0.85
```

Теперь к оценкам рейтинга. Это, наверное, самая спорная часть. Я реализовал эту оценку простым преобразованием вероятности положительной оценки в число от 1 до 4 и от 7 до 10. Классификатор не обучался на нейтральных отзывах, поэтому его тесты на таких отзывах могут давать плохие результаты. Также у меня была идея как-то сделать здесь регрессию, но, учитывая, что два разных человека, написав одинаковый отзыв, могут поставить две разные оценки, я отмёл эту идею.

Django приложение и его последующее развёртывание

Наверное, это была самая нетривиальная часть задания для меня, так как с этим фреймворком ещё пару дней назад совсем не был, да и вообще с Web-разработкой. Тем не менее, ознакомление с фреймворком было осуществлено, после чего было реализовано простое Django – приложение, применяющее уже обученную модель.

Для развёртки приложения я выбрал платформу Heroku из-за её, конечно, бесплатности и относительной простоты развёртки используя git.

https://simplelstmmodel.herokuapp.com/

Заключение

По итогу работы был обучен LSTM классификатор, имеющий точность 85%, а также было создано Web-приложение на основе Django, применяющее эту модель к введённому Пользователем отзыву к фильму для последующего сентимент анализа.