Инструкции и материалы к шестому домашнему заданию

Представьте, что вам нужно реализовать алгоритм анализа тональности отзывов о банках. Задача изначально стоит в самом неопределенном виде: научиться отличать позитивные отзывы и негативные. Нет ни более четкой постановки, ни данных, но нужно как-то продемонстрировать возможность решения задачи, потратив на это не слишком много времени.

1. Формализуйте постановку задачи, ответив на вопросы: a) какая задача будет решаться (классификация/регрессия/кластеризация/еще что-то); b) какими будут целевые значения (например, классы в задаче классификации) - почему именно такими, как прогнозы будут показываться в демонстрации (например: каким-то числом, текстом, цветом); c) как измерять качество, чтобы было более-менее интуитивно понятно, высокое оно или не очень, d) на каких отзывах должна работать демонстрация (язык, длина, наличие ошибок и сленга в тексте). Возможно вам потребуется частично справиться со следующим пунктом задания, чтобы постановка задачи была совместима с данными, которые вы можете достать.

2. Соберите данные: тексты отзывов и какую-то разметку (желательно, не менее 5000 примеров), позволяющую судить о том, какие отзывы позитивные, а какие нет. Например, можно попробовать найти готовую выборку, либо распарсить тексты с каких-нибудь сайтов (лучше - с помощью scrapy или какой-то другой библиотеки, предназначенной для краулинга, но можно с помощью requests и какой-нибудь библиотеки для парсинга html-кода).

3. Постройте какую-нибудь несложную модель в Ipython Notebook с помощью sklearn. Не стоит слишком сильно увлекаться чисткой данных или усложнением модели, помните, что исходная цель - сделать простой прототип, который будет как-то сносно работать в большом числе случаев, но может не очень удачно

обрабатывать сложные ситуации. Сохраните модель с помощью pickle или joblib, чтобы в демонстрации просто загружать обученную модель и vectorizer.

4. Реализуйте демонстрацию работы вашего алгоритма, сделав простую веб-страничку, например на Flask. Логика должна быть предельно простой: текст отзыва вставляется в текстовое поле, пользователь нажимает кнопку и получает прогноз тональности. Поиграйтесь с получившейся демонстрацией, возьмите какие-нибудь запросы для тестирования и сделаете скриншоты результатов.

- **Для сдачи задания** нужно прислать краткий отчет по каждому пункту выше (что делали, к каким выводам пришли, можно отдельным документом, можно прямо в теле письма) и код (кроме пункта 1 - в нем код не нужен). В пункте 3 не забудьте указать значения метрики качества, и как вы его измеряли. В пункте 4 прикрепите несколько скриншотов с примерами работы вашей демки. Также нужны инструкции, как запустить парсер и демку (в пунктах 2 и 4, соответственно).
- Собранную выборку присылать не надо, обученный классификатор и vectorizer - лучше прислать, чтобы демонстрацию можно было запустить, но это остается на ваш выбор - если по каким-то причинам они весят 100+ Мб - можно не присылать.
- При оценке задания каждый пункт из 4 оценивается 5 баллами, суммарно - 20 баллов. За то, что какой-то пункт выполнен "не так идеально, как хотелось бы" баллы сниматься не будут - максимум соответствует просто нормальному выполнению всех пунктов.
- В конце отчета укажите пожалуйста, сколько примерно вашего времени потребовалось для выполнения задания (т.е. если вы запустили на ночь парсер и пошли спать - ночь к этому времени не прибавляется :). В идеале должно получиться несколько часов, если не отвлекаться от выполнения задания, но очень хочется проверить, что это действительно так.
- Все вопросы по заданию пишите на адрес: **xead@yandex-team.ru**

- Готовые работы присылайте туда же.

Материалы:

- Код парсера, использующего scrapy, и примеры собранных таким образом выборок: ithappens_dataset.tar.bz2 @romovpa 172
- Блокнот с примером того, как парсить страницы с помощью requests: PageParsing.ipynb (но лучше разберитесь с scrapy - получится намного быстрее)
- Блокнот, в котором прогнозируется количество лайков по тексту поста: IT_Happens_model.ipynb
- Хорошее введение во Flask: https://ru.wikibooks.org/wiki/Flask - буквально на час-полтора можно научиться делать все базовые вещи (для формального выполнения задания может не понадобиться, т.к. далее в материалах дается вполне самодостаточный код для демки - нужно только запустить)
- Код демонстрашки для сентимент-анализа: simple_demo.zip (без задампленного классификатора - его вам нужно будет подготовить самостоятельно, возможно переписав затем sentiment_classifier.py под свою постановку задачи)