Отчёт за 30 марта.

Сухочев Александр

Достижения:

1)Создал словарь с помощью API Яндекс.Переводчика, переводящий русскоязычные леммы на английский язык. Применив его к датасету, получил повышение accuracy алгоритма примерно на 3-4%, сейчас accuracy на датасете с sure=True и отелями, которые искали по крайней мере 5 раз, составляет примерно 0.854. (эксперименты приложены в виде html и ipynb файлов в директории data и source соответственно на Bitbucket-e).

2)Написал код для удобного встраивания дополнительной предобработки текста и для будущего голосования нескольких классификаторов. Пока планируется устроить голосование между kNN и Naive Bayes, как двумя разнородными классификаторами, которые гипотетически должны показать неплохую производительность на множестве с таким количеством классов, как у нас. Написал обёртку для Naive Bayes из scikit-learn, которая пригодится для оценки ошибочных классификаций и голосования(также добавлены на Bitbucket-e в папку source).

Замечания:

1)Стоит отметить, что словарь был составлен немного не правильно, потому что, к примеру, в ответ на запрос перевести слово “отель” API Яндекс.Переводчик-а могла выдать строку “the hotel”, вместо “hotel”. Элементы “the hotel” в последствии конкатенировались(это было ошибкой), что давало новый признак ‘thehotel”, никак не связанный с “hotel” это никак не улучшало классификацию. Исходя из этого, можно предположить, что при правильном составлении словаря качество классификации может стать лучше. Проверить это прямо сейчас к сожалению сложно, потому что программно редактировать уже существующий словарь представляется мало возможным, так как, например, не всегда понятно, нужно ли в слове “ahead” убирать первую гласную “a”. Создание нового словаря займёт где-то два дня из-за длительности создания и ограничения по количеству слов за день.

2)Также качество классификации может повыситься, если правильно разбивать выборки на обучающую и тестовую. Сейчас они просто разбиваются случайно, и из-за большого обилия небольших классов может произойти так, что в тестовую выборку попадут объекты класса, не представленного в тренировочной выборке. Пока думаю над тем, как статистически грамотно их разбить.

3)Стоит также отметить, что все нынешние результаты получены без настройки гиперпараметров, как препроцессоров, так и классификаторов. Такая настройка зачастую проводится с помощью grid search и кросс-валидации, и может дать прирост качества на 1-3%. Она требует больших вычислительных и(или) временных ресурсов. На моём нынешнем компьютере это вообще представляется мало возможным. Может быть можно в рамках проекта получить какую-нибудь высокопроизодительную виртуальную машину, чтобы там всё настроить?

4)Также стоит осторожно относиться к показателям времени работы и качества классификации. Эти показания в лучшем случае делались по паре-тройке экспериментов. Если accuracy алгоритма изменяется слабо на уровне тысчяного порядка, то вот 90%-ые квантили времени обработки одного запроса меняются от запуска к запуску от 0.05 до 0.1. Для более надёжной оценки этих данных также желательно иметь какую-нибудь мощную виртуальную машину под рукой, чтобы можно было запустить несколько экспериментов в разных процессах, и, собрав достаточно данных, оценить нужные показатели, как положено, с доверительными интервалами.

Проблемы:
Сложно организовать взаимодействие между инструментами scikit-learn-а и большими матрицами данных типа scipy.sparse.csr_matrix, в которых вынужденно обучающее и тестовое множество. Я думал решить эту проблему для Naive Bayes с помощью генератора chunk-ов и использовать опцию partial_fit, как указано в их примерах, но это вызвало новую непонятную ошибку, в причине которой я пока не могу разобраться. (этот неудачный эксперимент также можно найти в виде html и ipynb файлов на Bitbucket-е. На stackoverflow ответов не нашёл, буду рад совету на этот счёт).

Планы на будущее:
Создать взвешенное по качеству голосование разнородных не ресурсоёмких классификаторов. Провести более аккуратное оценивание качества классификации и времени обработки запроса.