Автор: Темур Холматов
Задание: Сервис для Рекомендации Новостных Статей

Репорт по Исследованию

Выбор Датасета

Для начала, я решил выбрать один датасет из предложенных в описании задания, чтобы сфокусироваться, ибо задачи так немало. Я решил использовать "Russian news articles" датасет с сайта Webhose по следующим соображениям:

1. Данные хорошо структурированы в JSON формате.
2. Текстовые данные предоставлены в хорошем качестве для последующей обработки.
3. Имеется большой пул различных дополнительных признаков, которые могли бы быть использованы в системе рекомандаций. Например, даты публикаций могут помочь предлагать наиболее свежие новости (спойлер: до этого дело не дошло).
4. Датасет является русскоязычным. Хотя я не заметил жестких требований для выбора языка, я посчитал, что выбор данных на русском языке был бы более релевантным (все ж в России живем) и менее шаблонным (меньшее количество готовых туториалов, ждущих пока их "скопипастят" по сравнению, например, с английским языком).
5. Достаточно большое количество документов - 291,584. Конечно, в датасете "Rossiya Segodnya" документов намного больше - 1,003,869. Однако, на данный момент, я не вижу смысла выбирать большой датасет, по крайней мере, в силу невысокой производительности технических средств.
6. В дополнение к придыдущему пункту, нужно отметить, что в датасете "Rossiya Segodnya" тексты содержат много лишних примесей в виде html

тэгов и подобного. Не то, чтобы я избегаю предобработку данных, однако, это могло бы занять достаточно времени, которое я предпочел бы потратить на более важные задачи.

Эмбеддинг

Я использовал FastText эмбеддинг в комбинации с Tfidf Weighted эмбеддингов для векторизации предложений. Соглашусь, что это не лучшее решение, ибо можно было взять ELMo, но никто не просил выбирать только лучшее решение, а следовательно, есть выбор. FastText по сути улучшенный Word2Vec (с буквенными n-граммами, что позволяет обрабатывать неизвестные словарю слова). Минус FastText а в том, что он не context-senstive и, конечно, результаты соответственно хуже, чем у ELMo. Но оставим ELMo для лучших времен, мы ведь пришли сюда исследовать неизведанное, а не лучшее.

Кластеризация

Для baseline решения было предложено использовать KMeans. Это конечно неплохое решение, однако, его можно улучшить. Я нашел хорошее решение - использовать алгоритм Approximate Nearest Neighbors, который по сути входит в тот же класс алгоритмов, что и KMeans, но позволяет аппроксимировать результаты, при этом давая прирост в скорости и меньшее потребление памяти.

API (или подобное)

Было предложено написать REST API на Flask. Однако, я не имею опыта работы с Flask и посчитал, что на самом деле не имеет значения какой API я решу написать. Изучить работу на Flask не сложно (тем более у меня есть опят работы на Django

(разные вещи, я знаю)), но нужно ли это? Думаю нет, ибо это скорее второстепенная задача и более нацелена на то, чтобы изобразить свою модели работающей в реальном продукте. А это можно сделать и иным способом - просто создав небольшой Python проект и симитировав работы системы, что я и сделал.

Дополнение

Конечно, это не все, что мне пришлось сделать и исследование на самом деле было большим и долгим. Больше подробностей вы сможете найти в ноутбуках и в коде. Там я комментирую работу и объясняю некоторые важные моменты. В частности, предлагаю начать с Data-Analysis.ipynb.

Также можете вглянуть на мой Trello board, который я использовал для мониторинга процесса своей работы. Все же порой приятно видеть, что ты перетаскиваешь таски в Done, это дает стимул работать дальше.
Ссылка: https://trello.com/b/RMEgi2XQ/news-recommendation-system