Министерство науки и высшего образования Российской Федерации

Московский физико-технический институт

Промышленное машинное обучение

Осень

2022

Лабораторная работа №2

CONTENT-BASED РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА ФИЛЬМОВ – SPARK DATAFRAMES

Цель работы:

По имеющимся данным: https://files.grouplens.org/datasets/movielens/ml-20mx16x32.tar построить content-based рекомендации по образовательным курсам. **Запрещено** использовать библиотеки pandas, sklearn и аналогичные.

Рекомендации по выполнению работы:

Для подбора рекомендаций следует использовать меру TFIDF, а в качестве метрики для ранжирования — косинус угла между TFIDF-векторами для разных курсов.

Что такое TFIDF? TF — это term frequency: по сути, сколько раз слово встречается в этом документе. Если мы сделаем такой word count по каждому документу, то получим вектор, который как-то характеризует этот документ.

мама - 2

мыла - 1

раму - 1

лапу - 1

роза - 2

упала - 3

Если мы сравним вектора, рассчитав дистанцую между ними, то получим вывод – насколько похожи эти тексты. Назовем этот подход наивным.

Этот подход наивен, потому что мы как бы присваиваем одинаковый вес каждому слову, которое у нас есть в тексте. А что, если мы попробуем как-то повысить значимость тех слов, которые часто встречаются только в этом тексте? Для этого мы посчитаем DF – document frequency: по сути, число документов, в которых есть вхождение этого слова. Мы хотим "штрафовать" слово за частое появление в документах, поэтому делаем инверсию этой величины – буква I в TFIDF. Теперь для каждого слова мы будем считать TF и делить на IDF. Так мы получим другой вектор для нашего документа. Он может быть более правильным для наших задач.

TFIDF нужно считать для описаний курсов (desc). При извлечении слов из описания словом считаем то, что состоит из латинских или кириллических букв или цифр, знаки препинания и прочие символы не учитываются.

Для поиска слов можно использовать такой код на Python (может быть проблема с распознаванием юникода).

```
regex = re.compile(u'[\w\d]{2,}', re.U)

regex.findall(string.lower())
```

Сам TFIDF реализован в Spark, писать с нуля вычисления не требуется. При вычислении TF с помощью HashingTF использовалось число фичей: 10000. То есть: tf = HashingTF(10000).

Основное задание: дать рекомендации по случайному набору (от 10 до 100 штук) ID фильмов (поле Movie Id).

Дополнительное задание: построить полный CI/CD цикл модели с использованием Docker/Jenkins/подходов тестирования (см. работу №1).