Relatório - Sistema de Recomendação

Algoritmos para Big Data **Docentes:** Adriano Lopes e João Oliveira

Mestrado em Engenharia Informática

ISCTE - Instituto Universitário de Lisboa **Alunos:** Diogo Santos Nº 96332 e João Faria Nº 78980

Introdução

Foi desenvolvido um sistema de recomendação com o algoritmo ALS e um dataset que contém avaliações de produtos da Amazon submetidas por utilizadores entre Maio 1996 e Julho 2014. O dataset na sua totalidade contém não só as avaliações dos produtos como a metadata dos mesmos (descrições, categorias, preço, marca e imagens) e outros produtos comprados recentemente. Para além deste dataset são ainda disponibilizados diferentes subsets do mesmo. Para o algoritmo ALS só precisamos do identificador do utilizador, do item e a avaliação atribuída pelo que escolhemos o subset "*rating only"*, com um tamanho de 3.2GB, onde consta a informação que precisamos mais a coluna *timestamp* que iremos remover posteriormente.

O dataset que utilizámos encontra-se disponível para descarregar no seguinte *link:*
https://drive.google.com/file/d/1tC1NoeWFKFaFp_MDFYTXQMj_YfOEtlRs/view

Por fim, o nosso trabalho está dividido em três *notebooks*:

- *Data Preparation* - Ler o *dataset* para um *dataframe* e analisar o seu conteúdo, nomeadamente a sua estrutura, a existência de valores nulos ou incorretos na tabela de rating, visto que apenas podem ser números inteiros de 1 a 5. Mudança de nome das colunas para melhor legibilidade, bem como indexação dos valores de No final guardamos o *dataframe* no formato *parquet.*
- *Creating Model and Computing results:* Ler o dataset criado pelo notebook anterior de volta para um *dataframe* de modo a criar o modelo ALS. Divisão do dataset em treino e teste. Treinar o modelo para gerar as recomendações.
- *Results:* Um *notebook* "*frontend*" para apresentar os resultados, nomeadamente através da execução de comandos sql nas tabelas persistentes criadas pelo *notebook* anterior.

Data Preparation

O dataset que escolhemos trazia apenas informação relevante, uma coluna com o identificador do utilizador, o identificador do produto que avaliou, a avaliação e o timestamp da mesma. Começamos por selecionar uma sample do dataset inicial, entre 1 a 5%. Verificámos a estrutura do *dataset* e o *datatype* de cada coluna. Alteramos o nome das mesmas para serem facilmente identificáveis, de [“_c0”,”_c1”,”_c2”,”_c3”] para [“*Reviewer*”,”*Item*”,”*Rating*”,”*Timestamp*”] . Verificámos a existência de valores nulos em qualquer uma das colunas. Analisámos a coluna dos ratings para valores fora do rating 1-5 habitual da Amazon. Como não existiam constrangimentos, procedemos à conversão da tabela dos ratings para Integer, visto que os ratings são valores inteiros. Analisámos a coluna dos ratings para valores fora do rating 1-5 habitual da Amazon.

Como o algoritmo ALS trabalha apenas com valores numéricos e as colunas *Reviewer* e *Item* são identificadores, mas no formato *string* foi necessário usar a função *StringIndexer* definida no *package pyspark.ml* que converte o valor num índice númerico. Nesta fase foi onde nos deparámos com algumas dificuldades quando tentávamos selecionar uma sample maior, por volta dos 10%. Encontrámos diferentes e diversos erros, nomeadamente *java.OutOfMemory - java heap space*, *KryoSerializer.BufferOverflow* e o *kernel* simplesmente deixar de responder após algum tempo. Tentamos várias alternativas como aumentar o espaço de armazenamento da máquina virtual, aumentar o *spark.driver.memory*, *spark.executor.memory*, *spark.kryoserializer.buffer.max* atraves do *SparkContext* e *SparkSession* sem sucesso. Tentámos também máquina virtual com 450GB de armazenamento, 12GB de Ram e 12 cores onde ocorreram os mesmos erros. No *AWS*, também não nos foi possível correr o dataset na sua totalidade devido ao erro *KryoSerializer.BufferOverflow.* Tentamos como meios de resolução aumentar *spark.kryoserializer.buffer.max* através da SparkSession, SparkContext e SparkConf, todas sem efeito. Por fim, removemos a coluna “*TimeStamp*” e guardamos o novo *dataframe* localmente no formato *parquet* e numa tabela.

Creating Model and Computing Results

Nesta fase voltamos a carregar o *dataset* num *dataframe* através de uma *sql query* ou do comando *spark.read.load* e indicando o nome da tabela. Dividimos o *dataframe* em dois, um *dataframe* de treino e outro de teste. Definimos a função que cria o modelo ALS indicando as colunas *ReviewerID, ItemID, Rating* e o *coldStart Strategy = drop* para remover linhas nulas, apesar de sabermos que as mesmas não existem devido à nossa análise do *dataset* inicial no *notebook* anterior. Antes de dar *fit* verificamos se existe um modelo guardado localmente e se sim usamos esse modelo, se não houver damos *fit* do modelo e guardamos localmente para na próxima iteração já existir um modelo e pouparmos tempo e recursos saltando este passo. Avaliamos o modelo através da computação do *RMSE*(*Root Mean Square Error)* no *dataframe* de teste que nos indica a qualidade do modelo através da medição dos valores previstos pelo modelo e os reais. Finalmente, computamos as primeiras cinco recomendações para todos os itens e utilizadores, guardando os resultados em tabelas persistentes.

Results

Neste último *notebook* apresentamos os resultados calculados através de *sql queries* nas tabelas que guardámos no *notebook* anterior. Guardar os dados em tabelas persistentes permite que os dados possam ser acedidos através de, por exemplo, um website e neste caso acedidos noutro notebook. Este *notebook* serve esse mesmo propósito, apresentamos a tabela das recomendações de todos os utilizadores e itens, e por fim apenas para um utilizador e um item.

Nota: Como mencionado acima, encontrámos vários erros que não nos permitiram correr a maior parte do dataset apesar dos nossos esforços para os tentar corrigir. Devido a isto a interpretação do modelo e dos resultados é bastante limitada.

References

McAuley, J. (n.d.). Amazon review data. Amazon Product Data. Acedido a 14, Maio de 2021, from http://jmcauley.ucsd.edu/data/amazon/links.html

***Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering** R. He, J. McAuley WWW, 2016*