Programming Assignment Collaborative Movie Recommendation

Paulo Henrique Maciel Fraga 2018054451

ABSTRACT

Utilizamos um recomendador híbrido, user-based e item-based, baseado em matrix factorization e cosine similarity para resolver o problema do “Collaborative Movie Recommendation”.

1 IMPLEMENTATION

O problema proposto era: dado um conjunto de avaliações conhecidas de usuários a itens, deveria-se gerar previsões sobre avaliações desconhecidas de um novo conjunto de usuários e itens.

Para resolver a tarefa, agrupamos e filtramos os dados utilizando Data Frames da biblioteca pandas. Assim, determinamos quantas avaliações existiam de um determinado usuário e quantas avaliações existiam de um determinado item e as filtramos baseado num valor arbitrário de avaliações. Na versão final consideramos apenas usuários que tinham realizado pelo menos 5 avaliações e itens que tinham recebido pelo menos 5 avaliações (u5 i5).

Assim separamos as filtragens em dois Data Frames distintos, um que filtrava os dados baseado nas quantidade de avaliações de um usuário, para ser usado no modelo user-based, e um que filtrava baseado na quantidade de vezes que um item foi avaliado, para o modelo item-based. As operações de filtragem tinham complexidade aparentemente irrelevantes para o tempo total de funcionamento do algoritmo.

Desse modo, com os dados separados em Data Frames únicos em seguida carregamos os ratings únicos em uma matriz numpy. Assim, utilizando a matriz montada calculamos as cosine similarities multiplicando a matriz por ela mesma transposta e normalizamos os resultados. Repetimos os processos separadamente para calcular as similaridades de usuário e itens. Parte mais custosa do programa e diretamente ligada às dimensões das matrizes de ratings escolhidas, por isso o processo de filtragem foi essencial.

Assim, com as similaridades geradas já era possível fazer as recomendações. Carregamos e manipulamos os targets, novamente utilizando Data Frames pandas, e geramos os ratings previstos, multiplicando o vetor que representa as notas recebidas pelo item/usuário pelo vetor de similaridades entre item/usuários, isso tudo normalizado pelas similaridades utilizadas no cálculo.

Dessa maneira, para todos os itens/usuários que o processo era possível geramos os ratings previstos. Caso fosse possível calcular a predição com os dois modelos retornamos a média entre as notas, caso apenas um fosse possível retornaremos a predição do modelo e, por fim, caso nenhum dos modelos se aplicasse a predição em questão, apenas retornamos a predição média de todo o conjunto de treino pré filtragem.

2 CONSIDERAÇÕES FINAIS

Com o programa implementado foi possível resolver o problema proposto respeitando o tempo delimitado de 5 minutos, seguindo os requirements.txt especificados e ,com um erro médio de 1,74, segundo placar de colocações do keggle.

Além disso, foram testados diversos tamanhos diferentes de filtro, e o menor erro médio encontrado com filtros (u3 i10) (e = 1,74490), porém com um tempo um pouco maior de cálculo comparado ao (u5 i5) (1,74768). Filtros maiores ou menores não impactam expressivamente o erro médio das predições.

Houve uma tentativa não inclusa no código final de normalização das notas pré computação de similaridades, que falhou.

Por fim, pandas Dataframes e numpy arrays foram primordiais para a eficiência da implementação.