Programming Assignment Content-based Movie Recommendation

Paulo Henrique Maciel Fraga 2018054451

ABSTRACT

Utilizamos a métrica TFIDF para calcular as similaridades de roteiros de filmes para resolver o problema do “Content-based Movie Recommendation”.

1 IMPLEMENTATION

O problema proposto era: dado um conjunto de avaliações conhecidas de usuários à filmes e informações sobre os filmes, deveria-se gerar previsões sobre avaliações desconhecidas de um novo conjunto de usuários.

Para resolver a tarefa, pegamos o campo textual que representava enredo do filme (plot), o adaptamos para um modelo bag of words usando uma função de tokenização própria, calculamos o TFIDF dos filmes utilizando apenas manipulações em pandas DataFrame e geramos a similaridades entre itens para calcular as recomendações.

Durante a tokenização, palavras terminadas em "sses", "ies", "ss", "s" foram modificadas para, respectivamente, "ss", "i", "ss", "", seguindo a ideia do algoritmo de Porter. Também removemos terminações em "ly", "ing" e "ed".

Provavelmente devido a utilização de uma função própria, a tokenização gerava uma "linguagem" (conjunto de palavras únicas presentes em todos os enredos), muito extensa. Para tentar lidar com o problema, tiramos palavras que apenas apareciam em um único enredo do modelo, visto que nada impactam no cálculo de similaridades e que poderiam representar erros no parser das palavras ou erros na grafia das palavras do enredo (por se tratar de um campo textual aberto). Ainda sim, a linguagem se mostrou bastante extensa o que impactou negativamente a performance do código.

Assim, com o TFIDF calculado carregamos o dado em uma matriz numpy, de dimensão: filmes x tamanho da linguagem, para calcular a matriz de similaridades dos filmes. Para isso multiplicamos a matriz de filmes x palavra por ela mesma transposta e normalizamos os resultados.

O cálculo dessa matriz representa mais de 90% do gasto temporal do programa e, devido ao tamanho da linguagem já citado acima, o cálculo demora bastante excedendo o tempo estipulado pela especificação (o algoritmo gastou cerca de 10min). Possivelmente um processamento de linguagem natural mais sofisticado que considere apenas a raiz das palavras resolveria nosso problema temporal, uma vez que a dimensão da matriz que representa palavras únicas seria consideravelmente menor.

Assim, com as similaridades geradas já era possível fazer as recomendações. Carregamos os ratings conhecidos numa matriz numpy e carregamos e manipulamos os targets, novamente utilizando Data Frames pandas. Geramos os ratings previstos, multiplicando o vetor que representa as notas dadas pelo usuário a filmes pelo vetor de similaridades entre enredos, isso tudo normalizado pelas similaridades utilizadas no cálculo.

Dessa maneira, para todos os filmes que o processo era possível geramos os ratings previstos, caso não fosse retornamos a nota média de todos os filmes no conjunto de treino.

2 CONSIDERAÇÕES FINAIS

Com o programa implementado foi possível resolver o problema proposto seguindo os requirements.txt especificados, com um erro médio de 1,8, segundo placar de colocações do keggle. O tempo delimitado de 5 minutos, porém, foi excedido. Como já ressaltado a diminuição do tamanho da linguagem com um parsey melhor provavelmente resolveria o problema.

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