Programming Assignment #2

Content-Based Movie Recommender

Luiza Campos
luiza.chagas@dcc.ufmg.br
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais, Brasil

1 INTRODUÇÃO

O objetivo deste trabalho é implementar um recomendador de filmes baseado em conteúdo. Conforme discutido em aula [2], diferentes opções de implementação afetam a qualidade das recomendações. As recomendações deviam ser enviadas ao Kaggle.

2 MÉTODO

O método de recomendação baseado em conteúdo é bem amplo, com diversas escolhas relativas às categorias. Em comum, diversas delas tem a similaridade do cosseno, que é dada por:

$$\cos(a,b) = \frac{ab}{\|a\|\|b\|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \quad (1)$$

Um valor alto de cosseno indica que uma entrada está intimamente relacionada ao item e, portanto, é boa candidata a ser recomendada ao usuário.

Para melhor entendimento, os métodos utilizados serão divididos por categoria presente nos dados do OMDB.

2.1 Plot

Primeiramente, é importante falar sobre o processo de tokenização dessa categoria:

- Toda a string é colocada em caixa-baixa utilizando `.lower()`;
- Converte-se números em palavras (variação de num2words);
- Faz-se a stemização - reduzir palavras flexionando-as à sua base ou raiz - utilizando-se o Snowball Stemmer [1][3];
- Remove-se qualquer tipo de acentuação;
- Remove as *stop-words* utilizando as referenciadas por NLTK.

Abaixo temos um exemplo do antes e depois:

```
"In Montréal, Françoise's 5th four-hour
über was only 12.89 dollars"
['montreal', 'francoises', 'fifth', 'four-hour',
'uber', 'twelve', 'eighty-nine', 'dollars']
```

O `plot` foi avaliado usando-se similaridade do cosseno e TF-IDF.

2.1.1 Term Frequency-Inverse Document Frequency(TF-IDF). . TF-IDF é usado em Information Retrieval para extração de *features*. O peso de cada termo t é dado por $TF * IDF$.

Term Frequency é a frequência da palavra no documento atual em relação ao número total de palavras no documento.

$$TF(t) = \frac{\text{Frequency occurance of term t in document}}{\text{Total number of terms in document}}$$

Inverse Document Frequency é o número total de documentos pela frequência de ocorrência de documentos que contenham a palavra. Indica a raridade da palavra e ajuda a dar pontuação mais alta a termos raros nos documentos.

$$IDF(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing term t}}\right)$$

2.2 Genre

É feito o *One-Hot-Encode* da lista de gêneros. Nesse caso, armazenam-se todos os gêneros diferentes em colunas que contêm 1 ou 0 - 1 mostra que um filme tem aquele gênero e 0 mostra que não. A comparação é feita com a similaridade do cosseno.

Abaixo temos um exemplo do antes e depois:

```
['Action', 'Thriller']
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0]
```

2.3 Decade

`Decade` é uma categoria criada a partir de `Year`, dada por $\lfloor \frac{Year}{10} \rfloor$. A maior distância possível entre décadas é 12 (2010 a 1890). Assim, a similaridade entre dois filmes é:

$$Sim(a,b) = 1 - \frac{w * |a-b|}{12}$$

Onde $w \in (0,1)$ é peso arbitrário que evita que a similaridade seja 0.

2.4 IMDb Rating

Um aspecto muito discutido em sala de aula foi o conceito de *wisdom of the crowds*. No trabalho, essa informação era dada por `imdbRating`. Contudo, em `imdbVotes` é possível notar uma discrepância: alguns filmes tinham milhares de votos, outros apenas 8. Dessa forma, foi criada uma nova categoria denominada `weighted_rate` que penaliza filmes com poucos votos e bonifica filmes com muitos.

A fórmula de classificação ponderada do IMDb é usada como 'Pontuação' para regularizar a métrica.

$$WR = \frac{vR + mC}{(v+m)} \quad (2)$$

Onde R é a `imdbRating` do filme, v é o número de votos do filme (`imdbVotes`), m é o mínimo de votos necessário (usa-se 143 resultado de NP.QUANTILE com $q = 0.1$) e C é a média de nota de todos os filmes.

2.5 Bias do usuário

Todo usuário tem um viés. O bias do usuário é dado pela média do viés do usuário para cada item que este já avaliou (posto que o item tenha `imdbRating`). Temos o bias bruto, dado por $R_u - R_{imdb}$ e o

percentual dado por $\frac{R_u - R_{imdb}}{R_{imdb}}$. Com esse bias podemos estimar a nota que o usuário daria para um item i.

2.6 Cold-start cases

Cold-start cases são casos onde falta algum tipo de informação. Para isso é bom definir alguns valores: MEAN RATING é a média de valores (válidos) de `imdbRating`, AVERAGE RATING é a média ponderada (por quantidade de votos) de `imdbRating`. A mesma lógica se aplica a WEIGHTED MEAN RATING e WEIGHTED AVERAGE RATING que usam `weighted_rate`.

A seguir estão descritos os métodos utilizados para lidar com esses casos. Os pesos utilizados em cada caso foram determinados experimentalmente.

2.6.1 User e Item no treino.

Neste caso temos informações do usuário e item, mas não sabemos se os itens comparados tem todas as informações desejadas. Em um caso normal (sem nenhum tipo de cold-start), a predição será dada pela ponderação de: NOTA MÉDIA USUÁRIO, MÉDIA DE IMDBRATING E WEIGHTED_RATE DO ITEM, NOTA DO ITEM COM BIAS DO USUÁRIO e os valores de PLOT RATING, GENRE RATING e YEAR RATING (que são dados por 10 vezes a similaridade).

Contudo, pode ser que PLOT RATING, GENRE RATING e YEAR RATING não estejam disponíveis. No caso de não haver nenhum dado, a similaridade retornada é -1 e os pesos do vetor são modificados, zerando a posição do valor. Há o caso da similaridade ser 0, nesse caso o valor retornado é o da MÉDIA DE IMDBRATING E WEIGHTED_RATE DO ITEM.

2.6.2 User no treino.

Neste caso temos informações do usuário. Calculamos o bias dele (o percentual teve resultados superiores) e aplicamos esse bias em WEIGHTED AVERAGE RATING e AVERAGE RATING. A predição é a média dos valores.

2.6.3 Item no treino.

Neste caso temos apenas informações do item que o usuário deseja consumir. Além de WEIGHTED AVERAGE RATING e AVERAGE RATING utilizamos GENRE AVERAGE RATING e DECADE AVERAGE RATING e os valores de `imdbRating` e `weighted_rate` do item. O peso de cada valor varia de acordo com o item a ser previsto. Os valores são alterados caso ele tenha (ou não) uma determinada categoria.

2.6.4 User e Item fora do treino.

Neste caso não temos informação para recomendar de forma personalizada. Assim, a predição é dada pela média entre WEIGHTED AVERAGE RATING e AVERAGE RATING.

3 IMPLEMENTAÇÃO

O trabalho foi implementado em Python 3, utilizando as bibliotecas padrão e as permitidas em `requirements.txt`. Um grande desafio da abordagem foi o limite de memória pois o cálculo do TF-IDF demanda bastante (especialmente quando se tem muitos *tokens* distintos).

Uma medida importante foi a utilização de listas e dicionários, estruturas nativas do Python que diminuem o tempo de computação. Além disso, poucos valores são calculados ao longo de funções, com a maioria pré-calculada e armazenada na classe `Content`.

3.1 Classes

Foram criadas três classes. A classe `setup` tem as informações sobre a configuração de treino: informações de tokenização, categorias a manter, pesos do caso com usuário e item e como calcular o bias. A classe `Dados` guarda: dicionário de usuários, dicionário de itens e um dicionário com usuário, itens e *ratings*. A classe `Content` guarda informações sobre o conteúdo: itens, média de notas e dicionário TF-IDF.

3.2 Algoritmos

Foram utilizados os algoritmos de TF-IDF e similaridade do cosseno. O TF-IDF tem a maior complexidade: $O(T_N * N)$ onde N é o número total de documentos e T_N é o número total de termos. Na prática, o número de documentos em que um determinado termo aparece é muito menor e, portanto, o tempo gasto será muito inferior a isso.

4 RESULTADOS

Todos os resultados exibidos a seguir foram obtidos em uma máquina `Intel® Core™ i7-6500U CPU @ 2.50GHz × 4` com 7.7 GB de memória em uma distribuição `Linux Ubuntu 18.04.5 LTS`. Todos os testes executaram em menos de 5 minutos, medidos utilizando-se o comando `time python3 main.py content.csv ratings.csv targets.csv > submission.csv`.

Muito dos resultados foram análises experimentais, com a mudança de pesos feita com base na intuição. Para vias de comparação, aqui estão resultados onde apenas os pesos do caso usuário e item foram modificados.

Table 1: Resultados Obtidos

run	user	item_avg	item_bias	plot	genre	year	RMSE
47	3/40	11/40	10/40	6/40	7/40	3/40	1.60028
49	0.05	0.5	0.2	0.09	0.11	0.05	1.60319
50	1/40	12/40	10/40	7/40	7/40	3/40	1.59776
51	0/40	12/40	11/40	7/40	7/40	3/40	1.59634

Como se pode observar, a média ponderada do item é muito influente, mas a média do usuário teve pouca influência - muito provavelmente por causa da nota do item com viés. Gênero foi a categoria que mostrou maior importância.

5 CONCLUSÃO

Ao final do trabalho era esperado um recomendador com bom RMSE. Essa tarefa foi concluída. Acredito que o valor atual de erro poderia ser melhorado adicionando-se um grafo de atores, diretores e escritores, mas ficaria muito grande e demorado para ser feito em 5 minutos.

REFERENCES

[1] Martin Porter. 2002. The English (Porter2) stemming algorithm. http://snowball.tartarus.org/algorithms/english/stemmer.html.

[2] Rodrygo L. T. Santos. 2021. Content-based Recommendation. https://www.youtube.com/watch?v=M6zMgMthxjo

[3] C.J. van Rijsbergen, S.E. Robertson, and M.F. Porter. 1980. New models in probabilistic information retrieval. (1980).