Recomendação de séries de TV utilizando Redes Bayesianas

Arthur Teixeira Jardim e Marcelo Marchioro Cordeiro

8 de Dezembro de 2019

1 Introdução

Com o grande consumo de informações e produtos via internet, os algoritmos de recomendação passaram a ser importantes para o auxílio na tomada de decisão. Os *streamings* de séries e filmes são modelos de negócio que dependem desses algoritmos para prender o usuário na plataforma e possuir maiores índices de audiência. Por outro lado o usuário também é beneficiado com as recomendações, não perdendo muito tempo no processo de decisão.

Com a quantidade de informações e com a disponibilidade facilitada das mesmas pelo acesso a Internet, as pessoas se deparam com uma diversidade muito grande de opções. Muitas vezes um indivíduo possui pouca ou quase nenhuma experiência pessoal para realizar escolhas dentre as várias alternativas que lhe são apresentadas [2]. Os sistemas de recomendações são ferramentas e técnicas que fornecem sugestões ao usuário, essas que estão relacionadas a diversos processos de tomada de decisão como por exemplo decidir uma série para assistir. De uma maneira geral, as recomendações são oferecidas como listas de classificação de itens, que após serem feitas são utilizadas para tentar prever quais são os interesses mais adequados ao usuário em questão [4].

A Netflix, popular serviço de *streaming*, estima a probabilidade de um usuário assistir a um filme ou série em particular do catálogo em alguns fatores como: interações com o serviço, assinantes com gostos similares e informações sobre os títulos. Além disso leva em conta informações do usuário como o horário e o tempo assistido [1].

As redes bayesianas são modelos utilizados para tomada de decisão baseado em condições de incerteza, podendo ser aplicada para a construção de um sistema de recomendação de séries de televisão. Analisando informações do usuário torna viável a construção de um modelo buscando ranquear séries com um maior grau de recomendação.

2 Objetivos e Metodologia

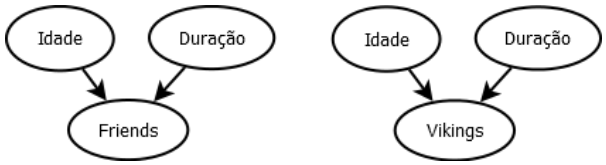
O modelo de recomendação de séries de TV foi construído a partir da utilização das redes bayesianas com o propósito de raciocinar em condições de incerteza. O sistema tem como objetivo analisar os dados do usuário e a partir disso listar as séries mais recomendadas para aquele usuário, para tal algumas séries devem ser mapeadas buscando encontrar informações que podem ser associadas aos parâmetros dos usuários. Com o intuito de consolidar os assuntos vistos em sala de aula, aplicamos em um modelo prático de recomendação de séries utilizando-se dos conceitos de redes bayesianas. Para a construção do sistema foi utilizada a plataforma *Anaconda Distribution* [3], desenvolvendo em *Jupyter* com a linguagem de programação *Python*. A biblioteca *pomegranate* [5] foi a utilizada para gerar os modelos de redes bayesianas, onde foram criadas as variáveis de distribuição discreta e de probabilidade condicional.

3 Desenvolvimento

Em um modelo de redes bayesianas cada nodo é associado com uma função de probabilidade que recebe como entrada um conjunto de valores para as variáveis pais do nodo e retorna a probabilidade (ou distribuição de probabilidade) das variáveis representadas pelo nodo. Utilizamos em nossa modelagem a conexão convergente onde dois nodos apontam para um específico, ou seja, o nodo que recebe as transições tem a probabilidade calculada a partir dos dois nodos de entrada.

Para a criação do sistema de recomendação foram levadas em conta as variáveis duração de episódio, que descreve o tempo médio de episódio assistido pelo usuário, e a variável idade, que busca classificar a faixa etária do usuário. Foram inseridas duas séries de TV no sistema para a realização dos testes e confirmação do funcionamento do modelo, foram elas: *Friends* e *Vikings*. As varáveis duração de episódio e idade são de distribuição discreta e para as probabilidades tornarem-se coerentes com cada série foram criados dois modelos, onde cada um representa uma série de TV (Figura 1).

Figura 1 – Modelos das séries *Friends* e *Vikings*



Fonte: autores, 2019.

No modelo de *Friends* as probabilidades nas variáveis idade e duração de episódio (Figura 2) foram definidas dessa maneira pelo fato de que o público da série é majoritariamente adolescente e adulto e a série tem em média episódios

de 22 minutos. No modelo de *Vikings* as probabilidades nas duas variáveis (Figura 3) foram definidas dessa maneira porque a série tem um público mais adulto e os episódios possuem uma duração média de 47 minutos. A variável idade tem as possíveis classificações: *baby*, criança, pré-adolescente, adolescente, adulto 1, adulto 2 e idoso. Sendo que cada série possui seu público alvo. Na variável duração de episódio a classificação é dividida em: baixo, médio, longo e muito longo. Em cada série é definida as classificações possíveis.

Figura 2 – Variáveis série *Friends*

```
idade_f = DiscreteDistribution({'pre-adolescente': 0.1, 'adolescente': 0.45, 'adulto 1': 0.45})

duracao_f = DiscreteDistribution({'baixo': 0.8, 'medio': 0.15 , 'longo': 0.05})
```

Fonte: autores, 2019.

Figura 3 – Variáveis série *Vikings*

```
idade_v = DiscreteDistribution({'pre-adolescente': 0.05, 'adolescente': 0.1, 'adulto 1': 0.7,
                                'adulto 2': 0.15})

duracao_v = DiscreteDistribution({'baixo': 0.05, 'medio': 0.15 , 'longo': 0.7,
                                  'muito longo': 0.1})
```

Fonte: autores, 2019.

As variáveis de probabilidade condicional foram definidas como *Friends* e *Vikings* já que vão ser o resultado do sistema de recomendação, esses nodos são justamente os que recebem as transições da conexão convergente.

Nas Figuras 4 e 5 são demonstradas as tabelas das condições de probabilidade da série Friends e de Vikinkgs respectivamente.

Figura 4 – Variável *Friends*

```
friends = ConditionalProbabilityTable(
        [['pre-adolescente', 'baixo',       'friends', 0.6],
         ['pre-adolescente', 'medio',       'friends', 0.4],
         ['pre-adolescente', 'longo',       'friends', 0.0],
         ['adolescente',     'baixo',       'friends', 0.6],
         ['adolescente',     'medio',       'friends', 0.3],
         ['adolescente',     'longo',       'friends', 0.1],
         ['adulto 1',        'baixo',       'friends', 0.6],
         ['adulto 1',        'medio',       'friends', 0.3],
         ['adulto 1',        'longo',       'friends', 0.1]], [idade_f, duracao_f])
```

Fonte: autores, 2019.

Figura 5 – Variável *Vikings*

```
vikings = ConditionalProbabilityTable(
          [['pre-adolescente', 'baixo',       'vikings', 0.1],
          ['pre-adolescente', 'medio',       'vikings', 0.2],
          ['pre-adolescente', 'longo',       'vikings', 0.5],
          ['pre-adolescente', 'muito longo', 'vikings', 0.2],
          ['adolescente',     'baixo',       'vikings', 0.1],
          ['adolescente',     'medio',       'vikings', 0.2],
          ['adolescente',     'longo',       'vikings', 0.5],
          ['adolescente',     'muito longo', 'vikings', 0.2],
          ['adulto 1',        'baixo',       'vikings', 0.1],
          ['adulto 1',        'medio',       'vikings', 0.15],
          ['adulto 1',        'longo',       'vikings', 0.6],
          ['adulto 1',        'muito longo', 'vikings', 0.15],
          ['adulto 2',        'baixo',       'vikings', 0.1],
          ['adulto 2',        'medio',       'vikings', 0.15],
          ['adulto 2',        'longo',       'vikings', 0.6],
          ['adulto 2',        'muito longo', 'vikings', 0.15]], [idade_v, duracao_v])
```

Fonte: autores, 2019.

A criação dos nodos e transições são mostradas nas Figuras 6 e 7. São criados os nodos com base nas variáveis mostradas anteriormente. Então são adicionados os nodos em cada modelo e por fim são criadas as transições entre os nodos que são definidas como: s1 → s3 e s2 → s3. O comando *bake* executa o modelo.

Figura 6 – Criação modelo *Friends*

```
s1 = Node(idade_f, name="idade_f")
s2 = Node(duracao_f, name="duracao_f")
s3 = Node(friends, name="friends")

model = BayesianNetwork("Friends")
model.add_states(s1, s2, s3)
model.add_edge(s1, s3)
model.add_edge(s2, s3)
model.bake()
```

Fonte: autores, 2019.

Figura 7 – Criação modelo *Vikings*

```
s1 = Node(idade_v, name="idade_v")
s2 = Node(duracao_v, name="duracao_v")
s3 = Node(vikings, name="vikings")

model2 = BayesianNetwork("Vikings")
model2.add_states(s1, s2, s3)
model2.add_edge(s1, s3)
model2.add_edge(s2, s3)
model2.bake()
```

Fonte: autores, 2019.

4 Resultados e Discussões

Para testar o modelo e poder medir resultados sobre o nosso sistema, foi realizada uma simulação com o modelo criado a partir das redes bayesianas. Como entradas do sistema optamos por escolher uma pessoa que seja classificada com a idade de adulto 1 e o tempo de duração de episódio baixa.

Os resultados dos testes retornaram para série *Friends* (Figura N) uma probabilidade de 0,216 e para a série *Vikings* (Figura N) uma probabilidade de 0,0035.

Figura 8 – Resultados série *Friends*

```
model.probability(['adulto 1', 'baixo', 'friends'])
```

```
0.21599999999999997
```

Fonte: autores, 2019.

Figura 9 – Resultados série *Vikings*

```
model2.probability(['adulto 1', 'baixo', 'vikings'])
```

```
0.003500000000000001
```

Fonte: autores, 2019.

O modelo acerta ao ter uma probabilidade maior para a série *Friends* pelo fato de que as informações do usuário coincidem com as especificações da série e não com as de *Vikings* que possui informações distintas comparadas a do usuário. Por outro os valores resultantes mostraram-se baixos em comparação com a realidade, para fins de comparação o modelo criado com lógica fuzzy para esse mesmo problema retornou resultados melhores.

Por possuir modelos para cada uma das séries, provavelmente escalando para um número grande de séries o custo computacional do nosso sistema seja alto. Abordar outros pontos criando um modelo de redes bayesianas mais complexo talvez seja mais preciso e tenha um custo menor.

5 Considerações Finais

O desenvolvimento desse trabalho foi fundamental para auxiliar no processo de entendimento da modelagem de uma rede bayesiana, mesmo a modelagem elaborada no trabalho sendo simples e considerando apenas a idade e a duração dos episódios, foi possível observar o comportamento dela na prática. O sistema

comportou-se como o esperado retornando um resultado coerente baseado no que foi visto nas diferentes bibliografias. A inferência obtida na aplicação dessa rede pode se tornar ainda mais precisa se for adicionado mais parâmetros, como por exemplo os gêneros de cada seriado, os atores principais, histórico de séries vista pelo o usuário entre outras variáveis. Isso tornaria a rede mais complexa com uma demanda computacional maior, porém a confiabilidade dos resultados seriam melhores e as indicações mais precisas.

Referências

[1] Como funciona o sistema de recomendações da netflix. https://help.netflix.com/pt/node/100639. Acesso em: 05 dez. 2019.

[2] S. C. Cazella, M. A. Nunes, and E. B. Reategui. A ciência da opinião: Estado da arte em sistemas de recomendação. *XXX Congresso da Sociedade Brasileira de Computação — Jornada de Atualização em Informática (JAI)*, 2010.

[3] Anaconda Inc. Anaconda versão 2019.10. https://www.anaconda.com/. Acesso em: 05 dez. 2019.

[4] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. http://www.inf.unibz.it/ ricci/papers/intro-rec-sys-handbook.pdf, 2011. Acesso em: 05 dez. 2019.

[5] J. Schreiber. Pomegranate versão 0.12.0. https://pomegranate.readthedocs.io/en/latest/index.html. Acesso em: 05 dez. 2019.