Case Recommender

1st Guilherme Henrique de Souza Nakahata
Department of Informatics
State University of Maringá (UEM)
Maringá - Paraná - Brazil
guilhermenakahata@gmail.com

***Abstract*—Sistemas de recomendações estão sendo cada vez mais utilizados por diversas empresas para recomendar os melhores produtos para o seus clientes ao longo das últimas décadas. O estudo dessa área está em constante evolução e novos algoritmos vêm surgindo, com isso, a necessidade de *frameworks* para ajudar a desenvolver um sistema de recomendação vêm aumentando. Case Recommender é uma ferramenta implementada em Python que possui algoritmos dos paradigmas baseado em conteúdo, colaborativo, não personalizado e o híbrido. Esses algoritmos podem ser aplicados nos cenários de Item Recommendation e Rating Prediction. Um pequeno estudo de caso foi abordado, comparando os algoritmos ItemKNN e Most Popular nas bases de dados MovieLens e Anime Recommendation. Os resultados foram comparados utilizando as métricas implementadas no próprio *framework*. Case Recommender se mostrou uma ótima ferramenta, com algoritmos no estado da arte e com uma fácil utilização para a comunidade em geral.**

***Index Terms*—Case Recommender, sistemas de recomendação, frameworks de remendações**

I. Introdução

Com o crescimento de conteúdo e com uma fácil disponibilidade de informações *on-line*, as pessoas se deparam com uma grande diversidade de opções nos mais variados itens, tornando-se um desafio com o qual os usuários enfrentam diariamente [1], [2]. Para lidar com essas dúvidas e necessidades, podemos adotar diversas estratégias para esse problema, como recomendações passadas de forma direta por outra pessoa ou através de opiniões de revisores de filmes e jornais [3]. No entanto, durante as últimas décadas, estudos mostram que sistemas de recomendações vêm sendo utilizados e conseguindo bons resultados em aplicações reais [4], [5].

Os sistemas de recomendações podem ser utilizados para prever classificações de preferência ou gerar uma lista personalizada de recomendações que o usuário possa estar interessado [6]. Para conseguir realizar essas recomendações, é necessário o uso de informações relevantes ao usuário, como *feedback* explícitos e implícitos [7].

Existem diversas ferramentas e bibliotecas capazes de implementar e avaliar os algoritmos de recomendação como: Mahout [1], LensKit [2], RiVal [3] e MyMediaLite [4]. Essas ferramentas podem ser utilizadas em grandes volumes de dados, múltiplos algoritmos de recomendação e uma variedade de base de dados.

O presente trabalho descreve as principais características, paradigmas, licenças, métricas e algoritmos implementados pela *framework* chamada Case Recommender, é implementado também um estudo de caso para observar os aspectos operacionais. Nesse experimento, foram utilizadas as bases de dados MovieLens e Anime Recommendation, utilizando os algoritmos ItemKNN e Most Popular durantes os cenários de *item recommendation* e *rating prediction*.

Case Recommender se mostrou muito confiável e flexível para novos algoritmos e métricas de comparação e avaliação, também se notou uma facilidade de utilização através de suas chamadas simples e que podem ser originadas de diferentes aplicações, como linhas de comandos, classes e funções. Durante o estudo de caso, o algoritmo ItemKNN mostrou uma melhor performance no cenário de *item recommendation*, enquanto o Most Popular teve um melhor resultado durante o cenário de *rating prediction*.

Esse estudo está organizado na seguinte sequência: a Seção II apresenta a ferramenta Case Recommender e suas características. Durante a Seção III é apresentado o estudo de caso, e as conclusões são apresentadas na Seção IV.

II. Ferramenta Case Recommender

Case Recommender é uma *framework* desenvolvida por Arthur F. da Costa e Marcelo G. Manzato [2] que busca proporcionar flexibilidade e extensibilidade em ambientes de pesquisa, tendo a preocupação em maximizar as utilidades e desempenho para pesquisas e educação. Essa ferramenta também foi projetada para oferecer suporte a uma ampla variedade de algoritmos de recomendações em diferentes cenários e métricas de avaliações, durante essa seção será abordada as principais características, algoritmos e paradigmas utilizado por essa ferramenta, a tabela II mostra um resumo das principais caraterísticas desse *framework*.

A. Linguagem e Instalação

A *framework* Case Recommender foi implementada em Python 3 utilizando bibliotecas como Scipy [5], Numpy [6], Pandas [7] e Scikit-learn [8]. A instalação dessa *framework* pode ser realizada em diferentes sistemas operacionais como Linux e Windows através do seguinte comando pip:

[1]https://mahout.apache.org/
[2]https://lenskit.org/
[3]http://rival.recommenders.net/
[4]http://www.mymedialite.net/
[5]https://www.scipy.org/
[6]https://numpy.org/
[7]https://pandas.pydata.org/
[8]https://scikit-learn.org/

TABLE I
PRINCIPAIS CARACTERÍSTICAS DO *framework*.

Características	Sim	Não
Possiblidade de inclusão de novos algoritmos	x	
Realiza Validação Cruzada automaticamente	x	
Realiza validação cruzada informando os N arquivos de fold	x	
Tratamento da esparsidade dos dados	x	
Mecanismo para lidar com grandes volumes de dados	x	
Documentação	x	
Uso em ambiente de produção		x
Uso em ambiente acadêmico	x	
Disponibilidade de suporte		x
Algoritmos do Paradigma não personalizado	x	
Algoritmos do Paradigma de filtragem colaborativa	x	
Algoritmos do Paradigma baseado em conteúdo	x	
Algoritmos do Paradigma baseado em conhecimento		x
Algoritmos do Paradigma Híbrido	x	
Algoritmos do Paradigma Sensível ao Contexto		x
Lida com diferentes formatos de entrada (conjunto de dados)		x

- $ pip install caserecommender.

Para a versão mais recente, pode-se instalar diretamente a versão do github.

- $ pip install -U git+git://github.com/caserec/ CaseRecommender.git.

B. Vantagens e desvantagens

A ferramenta apresenta pequenos problemas relacionado as atualizações das bibliotecas e alguns dos métodos utilizado precisaram ser modificado para conseguir realizar o estudo de caso, além disso, a restrição na entrada de dados faz com que a base de dados passe por uma adequação antes de ser utilizada, a falta de suporte para predições com *feedback* implícitos, como números de cliques e quantidade de visualização também se mostrou uma dificuldade. No entanto, por esse *framework* ainda estar em versões recentes, todas essas desvantagens e problemas podem ser corrigidas e se juntar com diversas vantagens encontradas, entre elas destacam-se:

- Cálculo de recomendações em larga escala;
- Otimizações em relação ao armazenamento de dados e resultados temporários.
- Contínuo melhoramento e restruturação buscando suportar cálculos otimizados utilizando bibliotecas cientificas em Python;
- Recomendações de alta escala, dinâmicas e rápidas utilizando chamadas simples;
- Flexibilidade para implementar novos algoritmos de recomendação utilizando estruturas e processos presentes no *framework*.

C. Paradgimas e algoritmos

A presente *framework* possui quatro principais paradigmas: baseado em conteúdo, responsável por seleção de informação baseada na filtragem de conteúdo; colaborativo baseado em explorar informações sobre o comportamento passado ou opiniões de usuários; não personalizado na qual não é utilizado nenhuma informação sobre o usuário e o híbrido na qual permite a combinação de algoritmos de diferentes paradigmas [8].

Os algoritmos implementados pela *framework* pode ser dividido em *Rating Prediction* e *Item Recommendation*. De acordo com os diferentes cenários de aplicação é possível combinar múltiplos algoritmos de recomendações utilizando técnicas de *ensemble* disponíveis na literatura [9]–[11]. A tabela II mostra os algoritmos disponíveis no *framework* e sua divisão.

TABLE II
ALGORITMOS UTILIZADO PELO CASE RECOMMENDER.

Rating Prediction	Item Recommendation
Matrix Factorization	BPRMF
SVD	ItemKNN
Non-negative Matrix Factorization	Item Attribute KNN
SVD++	UserKNN
ItemKNN	User Attribute KNN
Item Attribute KNN	Group-based
UserKNN	Paco Recommender
User Attribute KNN	Most Popular
Item NSVD1	Random
User NSVD1	Content Based
Most Popular	
Random	
gSVD++	
Item-MSMF	
(E)CoRec	

A ferramenta tem destaque para abordagens baseadas em *Neighborhood* e *matrix factorization*, esses algoritmos podem ser implementadas com diferentes métricas de similaridade, como: cosine, tanimoto, pearson e euclediana.

D. Licença e Distribuições

Case Recomender possui a licença GPLv3 [9] se caracterizando como *software* livre [10]. Nessa licença é necessário respeitar as seguintes liberdades:

- Liberdade do uso do *software* para qualquer finalidade;
- Liberdade para mudar o *software* de acordo com a necessidade;
- Liberdade para compartilhar o *software* e suas modificações com a sociedade;

O projeto pode ser visto na integra e gratuitamente no repositório do GitHub [11], permitindo assim o uso, teste e contribuição da comunidade.

E. Entrada de dados

É possível realizar conexões de aplicações com o *framework* através de linhas de comandos, classes e funções, permitindo que os pesquisadores não necessitem criar suas próprias ferramentas de recomendações.

Para essas conexões as entradas dos dados devem estar no formato: **u_id i_id *rating***, onde:

[9]https://www.gnu.org/licenses/quick-guide-gplv3.pt-br.html
[10]https://www.gnu.org/licenses/
[11]https://github.com/caserec/CaseRecommender

- u_id: Número inteiro referindo a identificação dos usuários;
- i_id: Número inteiro referindo a identificação do item;
- *Rating*: Número de ponto flutuante referindo o quanto o usuário gostou do item.

Esses dados podem estar separado por espaço, tabs ou vírgulas, caso exista colunas adicionais o *framework* irá ignorá-las.

Os algoritmos de recomendação são bastantes similares com o de predição, no entanto a maioria dos algoritmos de recomendação podem omitir ou ignorar o parâmetro de *rating* [2], porém, existe algoritmos que podem ignorar esses valores em ambas as abordagens.

F. Métricas de avaliação e validações

Case Recommender possui métricas de avaliações implementadas, como *Root Mean Square Error* (RMSE) e *Mean Absolute Error* (MAE), porém, para tarefas de recomendação é possível utilizar as medidas de *precision*, *recall*, *mean average precision* (MAP) e *Normalized Discounted Cumulative Gain* (NDCG), para ambos os cenários a framework possui o cálculo de esparsidade da base de dados. Para a comparação dos resultados é possível utilizar a análise estatística *two-sided paired t-test* [12].

Para a validação o *framework* possui suporte para o *k-folds-cross-validation*, também é possível utilizar o protocolo *All But One* [13] durante essa validação. A separação dos conjuntos de testes e treinos durante o k-folds acontece de forma aleatória.

III. Estudo de Caso

Durante essa seção será abordada as etapas adotadas para a realização do estudo de caso, em primeiro momento foi necessário decidir os algoritmos, tarefas e base de dados que seriam utilizados, após esses elementos definidos foi possível realizar as validações, avaliações e análise dos resultados obtidos, o fluxograma 1 representa essas etapas.

A. Algoritmos

Os Algoritmos utilizados durante o estudo de caso foram ItemKNN e Most Popular, essa escolha se deu pelos fatores:

- Possuírem paradigmas e características diferentes;
- Poderem ser aplicados na tarefa de *rating predictions* e *item recommendation*;
- Poderem ser utilizados com as informações que as bases de dados possuem.

O algoritmo Most Popular (Não personalizado) possui a característica de recomendar o item que foi mais popular entre os usuários durante um certo período de tempo, enquanto o algoritmo ItemKNN (Colaborativo) tenta encontrar similaridade entre itens para realizar as predições ou recomendações. Durante a implementação foi necessário somente chamar as funções desses algoritmos, o código utilizado pode ser visto na Figura 2, os parâmetros train_file e test_file são os diretórios da base de dados utilizadas, sendo divididas entre treino e teste.

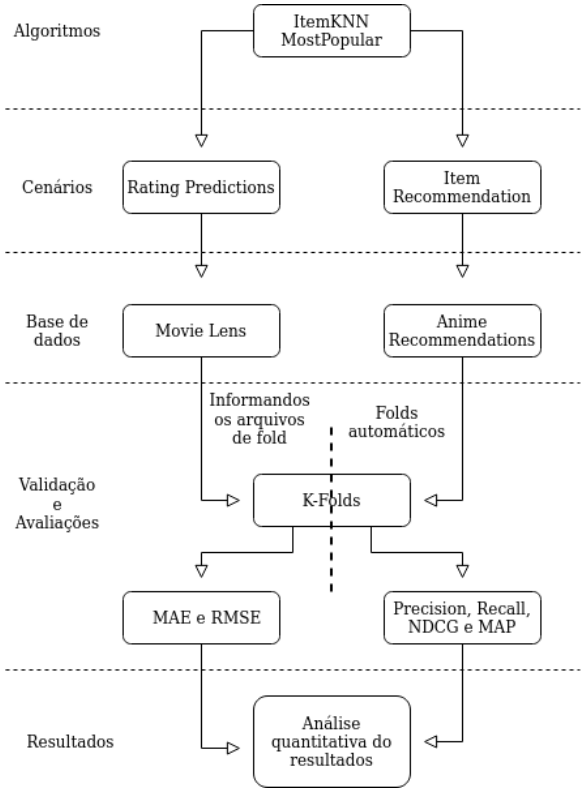

Fig. 1. Fluxograma estudo de caso.

```
from caserec.recommenders.item_recommendation.itemknn import ItemKNN
from caserec.recommenders.item_recommendation.most_popular import MostPopular

train_file = "train1.dat"
test_file = "test1.dat"

ItemKNN(train_file, test_file).compute()
MostPopular(train_file, test_file).compute()
```

Fig. 2. Código utilizado para chamada dos algoritmos.

B. Cenários

Durante o estudo de caso foi utilizado os dois tipos de cenários disponíveis no Case Recommender: *Rating Prediction* e *Item Recommendation*, os algoritmos utilizado em cada um desses cenários pode ser visto na tabela II.

O cenário *rating prediction* é um tipo popular de *feedback* explícito, na qual busca estimar uma classificação desconhecida a partir de um determinado conjunto de classificações conhecidas. Essas classificações previstas podem indicar o quanto um usuário irá gostar ou não de um item [2], [14].

Na tarefa de *item recommendation* é criado uma lista de itens que são prováveis de serem escolhidos pelo usuário [15], para o presente trabalho serão recomendadas listas de 5 e 10 itens.

C. Base de Dados

Para o estudo de caso foi adotado duas bases de dados, a base de dados *Movie Lens* e *Anime Recommendations*, as principais características dessas bases de dados podem ser vistos na tabela III. Essa variação da base de dados é proposital,

buscando abordar características e cenários diferentes com os algoritmos disponíveis no *framework*.

TABLE III
PRINCIPAIS CARACTERÍSTICAS DAS BASES DE DADOS

Características	Anime Recommendations	Movie Lens
Coletado por:	CooperUnion	GroupLens
Adaptado por:	Arthur Fortes	X
Inicio Coleta:	Dezembro de 2016	Setembro de 1997
Termino Coleta:	Dezembro de 2016	Abril de 1998
Quantidade Usuários:	4714	943
Quantidade Itens:	7157	1682
Faixa de avaliação:	1 – 10	1 – 5
Informações dos Usuários:	X	Idade, genero, ocupação e zip
Informações dos Itens:	Nome, genêro, tipo, episódios, avaliação e membros	Título, Ano de lançamento e Genêro

D. Validação e Avaliações

Para realizar a partição dos dados entre treinamento e teste, foi utilizado a validação cruzada k-folds. No total foram separados em 5 folds cada uma das bases de dados, no entanto essa divisão se originou de formas distintas conforme mostrado na Figura 1.

Buscando explorar o *framework* foi utilizado a partição automática dos *folds* na base de dados Anime Recommendations e a inserção manual dos arquivos de *folds* na base de dados Movie Lens, esses arquivos de *folds* foram disponibilizado pelos próprios autores da base de dados.

A implementação da validação cruzada automática na ferramenta é bem simples e utilizada 4 parâmetros, o diretório da base dados, o algoritmos de recomendação, o diretório que será salvo os folds e por fim a quantidade de folds que será criada, a Figura 3 mostra essa implementação. Para a inserção manual dos aquivos de folds é necessário somente apontar o diretório do fold com sua divisão de treino e teste, um exemplo dessa implementação pode ser visto na Figura 2.

```
from caserec.recommenders.rating_prediction.most_popular import MostPopular
from caserec.utils.cross_validation import CrossValidation

db = 'u.data'
folds_path = ''

recommender = MostPopular()
CrossValidation(input_file=db, recommender=recommender, dir_folds=folds_path, header=1, k_folds=5).compute()
```

Fig. 3. Código utilizado para a validação cruzada.

As avaliações dos cenários tiveram métricas diferentes, as medidas de MAE e RMSE foram utilizados para *Rating Prediction* e as medidas de precision@N, Recall@N , NDCG@N e MAP@N para *Item Recommendation*, também foi utilizado a média e o desvio padrão encontrado entre os *folds* para se ter uma avaliação mais completa dos resultados finais. Durante a implementação podemos escolher quais métricas serão calculadas pela *framework*, para isso é necessário criar uma array com os nomes dessas métricas e passar de parâmetro para a função compute, a Figura 4 mostra esse procedimento.

```
from caserec.recommenders.item_recommendation.itemknn import ItemKNN
from caserec.recommenders.item_recommendation.most_popular import MostPopular

train_file = "train1.dat"
test_file = "test1.dat"

metrics = ('PREC', 'RECALL', 'NDCG', 'MAP')

ItemKNN(train_file, test_file).compute(metrics= metrics)
MostPopular(train_file, test_file).compute(metrics= metrics)
```

Fig. 4. Código utilizado para a geração das métricas.

E. Resultados

Os resultados dos algoritmos durante o cenário de *item recommendation* foram separados em listas de 5 e 10 recomendações, conforme mostrado nas tabelas IV e na V. Com essa separação foi possível avaliar os top-5 e top-10 recomendações através das métricas definidas na Seção III-D.

TABLE IV
RESULTADOS CENÁRIO DE *Item Recommendation* COM LISTA DE 5 RECOMENDAÇÕES.

Anime Recommendation	Algoritmos	Média	Desvio Padrão
Precision	ItemKNN	0.3188	0.0034
	Most Popular	0.1444	0.0015
Recall	ItemKNN	0.1414	0.0026
	Most Popular	0.0587	0.0015
MAP	ItemKNN	0.5237	0.0046
	Most Popular	0.2819	0.0037
NDCG	ItemKNN	0.6068	0.0059
	Most Popular	0.3545	0.0053

TABLE V
RESULTADOS CENÁRIO DE *Item Recommendation* COM LISTA DE 10 RECOMENDAÇÕES.

Anime Recommendation	Algoritmos	Média	Desvio Padrão
Precision	ItemKNN	0.2571	0.0011
	Most Popular	0.1223	0.0016
Recall	ItemKNN	0.2064	0.0026
	Most Popular	0.0977	0.0031
MAP	ItemKNN	0.4904	0.00421
	Most Popular	0.2751	0.0029
NDCG	ItemKNN	0.5962	0.0050
	Most Popular	0.3729	0.0048

Com posse dos resultados notou-se uma baixa precisão e revogação do algoritmo Most Popular quando comparado com *ItemKNN*, mostrando que o algoritmo do paradigma não supervisionado teve problemas na recomendação em situações de falsos positivos e falsos negativos. Notou-se também que conforme a lista continha mais recomendações os resultados dessa métrica também abaixava, porém, como a complexidade de recomendação aumenta essa repentina baixa de valores é totalmente aceitável.

Apesar da baixa precisão e revogação encontrada nos resultados dos algoritmos, os valores de NDCG e MAP para o algoritmo *ItemKNN* mostram que a lista de recomendações é composta por uma boa parte da melhor ordenação e *ranks* possíveis, no entanto o algoritmo Most Popular não teve bons

resultados, esses valores se manteve próximo nas listas de 5 e 10 recomendações.

Para o cálculos do erro médio nos cenário de *Rating Predictions* foi utilizado as métricas MAE e RMSE, os resultados desses cálculos pode ser visto na tabela VI.

TABLE VI
RESULTADOS CENÁRIO DE *Rating Prediction*.

Movie Lens	Algoritmos	Média	Desvio Padrão
MAE	ItemKNN	0.8027	0.0042
	Most Popular	0.8154	0.0042
RMSE	ItemKNN	1.0445	0.0065
	Most Popular	1.0222	0.0060

A pequena variância entre as medidas MAE e RMSE mostra que existe uma pequena variância nos erros encontrados. Quando analisado os resultados dos algoritmos é possível notar que o algoritmo ItemKNN apesar de possuir o menor MAE, o valor de RMSE supera o do Most Popular, essa mudança pode ser justificada com possíveis outliers que ocorreram durante recomendação no algoritmo ItemKNN, visto que a medida RSME busca penalizar esses outliers.

Diante dos resultados aqui apresentados, foi possível notar através do desvio padrão que não ouve uma grande dispersão em torno das médias em nenhum dos resultados apresentados, também notou-se que o algoritmo ItemKNN representou melhor desempenho para o cenário de *Item Recommendation*, porém, para o cenário de *Rating Prediction* o algoritmo Most Popular apresentou os melhores resultados. Notou-se também que existe a possibilidade de melhorias, visto que foi usado somente dois algoritmos de dois diferentes paradigmas. Todo o código utilizado para se chegar aos resultados aqui apresentado pode ser visto na íntegra no repositório do github do autor [12].

IV. CONSIDERAÇÕES FINAIS

Nesse trabalho foi explorado a ferramenta Case Recommender. Nela foram sintetizadas as principais características, paradigmas, licenças, métricas e algoritmos implementos pela *framework*. Através desses dados, foi possível identificar vantagens e desvantagens do uso da ferramenta bem como suas limitações relacionadas à falta de suporte para predições com *feed-back* implícitos.

Como demonstrado no estudo de caso, foi possível utilizar os algoritmos ItemKNN e Most Popular em diferentes cenários e base de dados. Com base nos resultados, notou-se um baixo desempenho desses algoritmos, possibilitando que técnicas mais avançadas como a de *ensemble* possa ser aplicada e comparada com os resultados aqui apresentados. Nota-se também que a *framework* demonstrou estabilidade e consistências durante a partição dos *folds*, os cálculos avaliativos se mostraram corretos e bem escolhidos de acordo com o cenário seguindo criteriosamente o estado da arte.

Case Recommender se mostrou uma ferramenta confiável e que se preocupa em proporcionar um ótimo ambiente para estudo e pesquisa, podendo ser um grande aliado para que futuros problemas sejam estudados e novas hipóteses sejam implementadas.

[12]https://github.com/GuilhermeNakahata/CaseRecommender.git

REFERENCES

[1] S. C. Cazella, M. Nunes, and E. Reategui, "A ciência da opinião: Estado da arte em sistemas de recomendação," *André Ponce de Leon F. de Carvalho; Tomasz Kowaltowski..(Org.). Jornada de Atualização de Informática-JAI*, pp. 161–216, 2010.

[2] A. da Costa, E. Fressato, F. Neto, M. Manzato, and R. Campello, "Case recommender: a flexible and extensible python framework for recommender systems," in *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018, pp. 494–495.

[3] U. Shardanand and P. Maes, "Social information filtering: Algorithms for automating "word of mouth"," *Conference on Human Factors in Computing Systems - Proceedings*, vol. 1, 02 1995.

[4] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: a survey," *Decision Support Systems*, vol. 74, pp. 12–32, 2015.

[5] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender systems handbook*. Springer, 2011, pp. 257–297.

[6] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender systems handbook*. Springer, 2011, pp. 1–35.

[7] Q. Zhao, F. M. Harper, G. Adomavicius, and J. A. Konstan, "Explicit or implicit feedback? engagement or satisfaction? a field experiment on machine-learning-based recommender systems," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 1331–1340.

[8] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender systems: an introduction*. Cambridge University Press, 2010.

[9] G. Brown, "Ensemble learning." *Encyclopedia of machine learning*, vol. 312, pp. 15–19, 2010.

[10] L. Tang, Y. Jiang, L. Li, and T. Li, "Ensemble contextual bandits for personalized recommendation," in *Proceedings of the 8th ACM Conference on Recommender Systems*, 2014, pp. 73–80.

[11] A. F. da Costa and M. G. Manzato, "Exploiting multimodal interactions in recommender systems with ensemble algorithms," *Information Systems*, vol. 56, pp. 120–132, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437915300818

[12] T. Mitchell, "Machine learning. macgraw-hill companies," *Inc., Boston*, 1997.

[13] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, p. 43–52.

[14] H. Steck, "Evaluation of recommendations: Rating-prediction and ranking," 10 2013, pp. 213–220.

[15] P. Symeonidis, E. Tiakas, and Y. Manolopoulos, "Product recommendation and rating prediction based on multi-modal social networks," 10 2011, pp. 61–68.