UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Sistemas de Recomendação para *Marketplace* de Empregos Domésticos

Elisa Jorge Marcatto

São Carlos – SP

Sistemas de Recomendação para *Marketplace* de Empregos Domésticos

Elisa Jorge Marcatto

Orientador: **Prof.º Dr.º André Carlos Ponce de Leon Ferreira de Carvalho**

Monografia final de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como requisito parcial para obtenção do título de Bacharel em Engenharia de Computação.
Área de Concentração: Inteligência Artificial

USP – São Carlos
Novembro de 2017

Marcatto, Elisa Jorge
Sistemas de Recomendação para *Marketplace* de Empregos Domésticos / Elisa Jorge Marcatto. - São Carlos - SP, 2017.
65 p.; 29,7 cm.

Orientador: André Carlos Ponce de Leon Ferreira de Carvalho.
Monografia (Graduação) - Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos - SP, 2017.

1. Sistemas de Recomendação. 2. Aprendizado de Máquina. 3. Ciência de Dados. I. Carvalho, André Carlos Ponce de Leon Ferreira de. II. Instituto de Ciências Matemáticas e de Computação (ICMC/USP). III. Título.

A todos que contribuíram para este momento de alguma forma,
e para aqueles que se sentirem inspirados por este trabalho.

AGRADECIMENTOS

Agradeço a minha família por todo apoio durante os anos de estudo. Sem dúvida alguma, tudo o que conquistei é por causa de vocês. Obrigada por serem tão especiais!

Com muito carinho, agradeço ao Breno. Obrigada pelo amor e pela paciência nas noites mal e bem dormidas, você fez, e sempre faz, a diferença!

Agradeço, também, à "Zuera"! Sem vocês a graduação não seria a mesma, eu não seria a mesma. Obrigada por me fazerem crescer em tantos sentidos!

Gostaria de agradecer ao Rafael pelo apoio, pela paciência e pelos conselhos. Tudo seria impossível sem sua ajuda, obrigada!

Finalmente, agradeço ao Prof.º André Carlos Ponce de Leon Ferreira de Carvalho pela fundamental orientação. Obrigada pela paciência e pela atenção de sempre!

“A frase mais prejudicial em um idioma é: "Sempre foi feito dessa maneira".”

(Grace Hopper)

RESUMO

MARCATTO, E. J.. **Sistemas de Recomendação para *Marketplace* de Empregos Domésticos**. 2017. 65 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

O *Eletronic Commerce*, ou *e-commerce*, é um modelo de negócios que, cada vez mais, ocupa um importante papel no cenário econômico. Através dele, é possível oferecer uma grande variedade de produtos para qualquer que seja a necessidade de um usuário. Diante dos inúmeros itens que podem ser comercializados neste modelo, surge a necessidade de personalizar o processo de compra para cada tipo de cliente. Assim, os sistemas de recomendação têm sido a solução de grandes *e-commerces* para melhorar a experiência de seus clientes e, consequentemente, maximizar suas vendas. Neste trabalho, foi desenvolvido um sistema de recomendação para o *marketplace* de empregos domésticos Casa e Café. O intuito foi favorecer contratações de profissionais que melhor se enquadravam às necessidades do contratante.

Palavras-chave: Sistemas de Recomendação, Aprendizado de Máquina, Ciência de Dados.

ABSTRACT

MARCATTO, E. J.. **Sistemas de Recomendação para *Marketplace* de Empregos Domésticos**. 2017. 65 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

The Eletronic Commerce, or e-commerce, is a business model that increasingly plays an important role in the economic scenario. Through it, a wide variety of products can be offered for any need of a user. Given the numerous items that can be marketed in this model, there is a need to customize the purchasing process for each type of customer. Thus, recommender systems have been the solution of important e-commerces to improve the experience of their customers and consequently maximize their sales. In this work, a recommender system was developed for a marketplace called Casa e Café. The aim was to favor the hiring of professionals that best fit the contractor's needs.

Key-words: Recommender Systems, Machine Learning, Data Science.

LISTA DE ILUSTRAÇÕES

Figura 1 – Cauda longa: lojas físicas oferecem o que é popular, enquanto lojas *online* oferecem qualquer tipo de produto 22
Figura 2 – Comparação entre a classificação tradicional e o problema de filtro colaborativo 28
Figura 3 – Comparação entre a classificação tradicional e o problema de filtro colaborativo 33
Figura 4 – Relação entre complexidade do modelo, variação e distorção 39
Figura 5 – Histograma de atributo "CEP"para o conjunto de dados do contratante . . . 42
Figura 6 – Histograma de atributo "Segmento"para o conjunto de dados do contratante 43
Figura 7 – Histograma de atributo "CEP"para o conjunto de dados do profissional . . . 43
Figura 8 – Histograma de atributo "Segmento"para o conjunto de dados do profissional 44

LISTA DE TABELAS

Tabela 1 – Matriz de avaliação representando notas de livros em uma escala 1-5 27
Tabela 2 – Vantagens e Desvantagens das Técnicas de Filtragem Colaborativa Baseada em Memória 31
Tabela 3 – Características dos Métodos de Fatorização de Matrizes 35
Tabela 4 – Vantagens e Desvantagens do Modelo de Filtragem Baseada em Conteúdo . 36
Tabela 5 – Comparação dos Modelos de Sistema de Recomendação 37
Tabela 6 – Características Gerais dos Conjuntos de Dados de Contratante 44
Tabela 7 – Características Gerais dos Conjuntos de Dados de Profissional 44
Tabela 8 – Atributos que compõem o perfil do item 45
Tabela 9 – Atributos que compõem o perfil do usuário 46
Tabela 10 – Perfis dos profissionais que serviram como base para a obtenção dos valores de distância de referência para a avaliação das recomendações. 47
Tabela 11 – Matriz que contém os valores da distância euclidiana para cada par de profissionais 47
Tabela 12 – Parcela dos dados após cálculo de similaridade entre profissionais escolhidos e recomendados 49
Tabela 13 – Resultados das avaliações para o modelo final 50

LISTA DE ABREVIATURAS E SIGLAS

ALS ..... *Alternating Least Squares*
AM ..... Aprendizado de Máquina
BI ..... Baseado em Item
BME ..... Baseado em Memória
BMO ..... Baseado em Modelo
BU ..... Baseado em Usuário
EMA ..... Erro Médio Absoluto
FBC ..... Filtragem Baseada em Conteúdo
FCBM ... Filtragem Colaborativa Baseada em Modelo
FM ..... Fatorização de Matrizes
IA ..... Inteligência Artificial
MD ..... Mineração de Dados
MFL ..... Modelo de Fatores Latentes
MFL ..... Modelos de Fatores Latentes
REQM ... Raiz do Erro Quadrático Médio
SGD ..... *Stochastic Gradient Descent*
SR ..... Sistemas de Recomendação
SRH ..... Sistema de Recomendação Híbrido
SVD ..... *Singular Value Decomposition*

SUMÁRIO

1 INTRODUÇÃO 21
1.1 Motivação e Contextualização 21
1.2 Objetivos 23
1.3 Organização 23

2 REVISÃO BIBLIOGRÁFICA 25
2.1 Considerações iniciais 25
2.2 Sistemas de Recomendação 25
2.2.1 Estruturação dos Dados *25*
2.2.2 Modelos e Técnicas *27*
2.2.2.1 Sistema de Recomendação de Filtragem Colaborativa *27*
2.2.2.1.1 Filtragem Colaborativa Baseada em Memória 29
2.2.2.1.2 Filtragem Colaborativa Baseada em Modelo 31
2.2.2.2 Sistema de Recomendação de Filtragem Baseada em Conteúdo *35*
2.2.2.3 Sistemas de Recomendação Híbridos *36*
2.2.3 Métricas e Fatores de Avaliação *37*

3 DESENVOLVIMENTO 41
3.1 Considerações Iniciais 41
3.2 Atividades realizadas 41
3.2.1 Coleta, Análise e Escolha dos dados *41*
3.2.2 Escolha, Implementação e Avaliação do Modelo *45*
3.3 Resultados Obtidos 49
3.4 Dificuldades, limitações e trabalhos futuros 51

4 CONCLUSÃO 53
4.1 Considerações sobre o curso 53
4.2 Contribuições 53

REFERÊNCIAS 55

5 CÓDIGOS IMPLEMENTADOS 57

Capítulo 1

INTRODUÇÃO

1.1 Motivação e Contextualização

Uma das atividades mais populares na *Web* é a compra. O desenvolvimento de *websites* destinados à comercialização de produtos é uma tendência que vem sendo observada ao longo dos anos, caracterizando um modelo de negócios denominado *Eletronic Commerce* (comumente chamado de *e-commerce*).

A história dos *e-commerces* se iniciou em 1991 quando a *Internet* foi aberta para uso comercial. Entretanto, naquele período, a definição de *e-commerce* se restringia à execução de transações comerciais. Apenas em 2000, quando os procolos de segurança (como o HTTP) foram consolidados, o significado de *e-commerce* passou a ser a ação de comprar produtos e serviços por meio de meios de pagamentos eletrônicos.

De acordo com dados disponíveis, o crescimento desse modelo de negócio fez com que, no final de 2007 nos Estados Unidos, as compras e as vendas realizadas por meio de *e-commerce* assumissem 3,7% do total computado (LAND, 2008). Diante da visibilidade adquirida ao longo dos anos, logo foram aparecendo as vantagens de um *e-commerce* em relação às tradicionais lojas físicas, como a facilidade em procurar por diferentes produtos e a possibilidade de acompanhar as preferências de um cliente.

Uma loja física possui espaço limitado nas prateleiras e apenas pode apresentar uma parcela de todo o estoque existente. Em contrapartida, uma loja *online* exibe todos os produtos comercializados aos clientes. Assim, enquanto uma loja física oferece milhares de itens, uma loja *on-line* oferece milhões de itens.

Como se pode observar, não é possível adaptar uma loja física de acordo com a preferência de cada cliente, são oferecidos apenas os produtos que agradam o maior número de clientes. Porém, na loja *online*, podem ser exibidos somente os produtos de maior interesse de um cliente. Essa diferença entre o mundo comercial físico e o *online* é chamada de fenômeno da cauda longa (LESKOVEC; RAJARAMAN; ULLMAN, 2014). Esse termo é utilizado na Estatística para descrever distribuições de dados em que o volume de dados é decrescente.

Mais do que uma distribuição de dados, o termo cauda longa foi introduzido em (CHRIS, 2006) para descrever a estratégia de varejo de se vender uma grande variedade de itens em que cada membro vende pequenas quantidades (modelo *online*), ao invés de apenas os poucos itens

populares que vendem muito (modelo físico). A Figura 1 exibe o comportamento da distribuição de cauda longa, na qual o eixo vertical representa o número de vezes que um item é escolhido (popularidade) e o eixo horizontal representa os itens ordenados de acordo com sua popularidade.

As lojas físicas oferecem apenas os itens mais populares, ocorrendo uma restrição da variedade de produtos exibidos aos clientes. Dessa forma, elas estão representadas no lado esquerdo do eixo vertical na Figura 1. Já as lojas *online* conseguem oferecer toda a gama de itens, sendo representadas ao longo de toda a cauda da Figura 1.

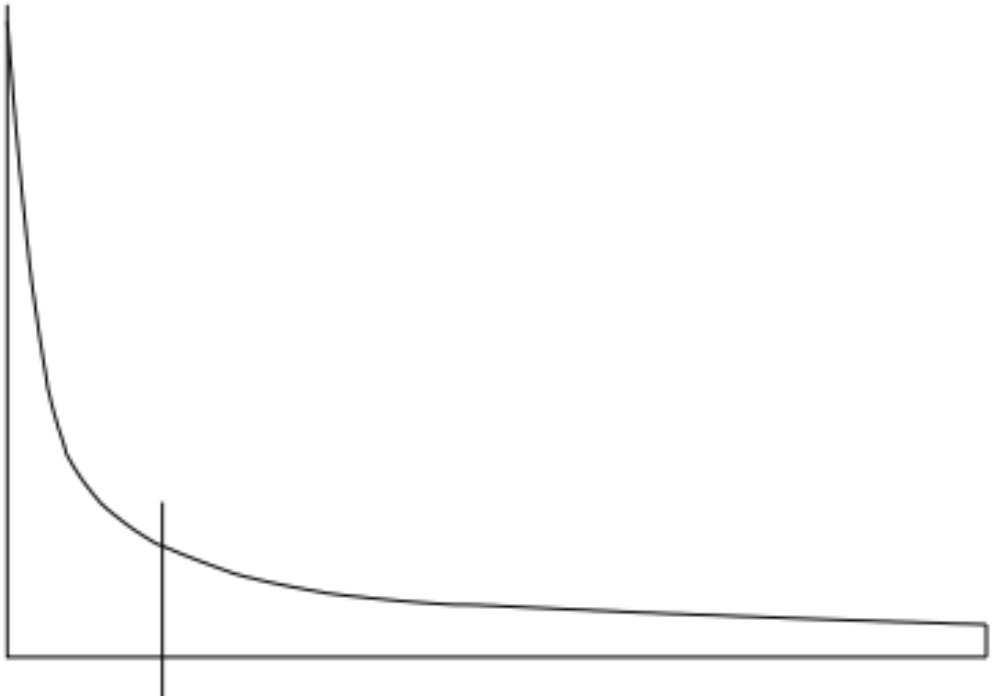

Figura 1 – Cauda longa: lojas físicas oferecem o que é popular, enquanto lojas *online* oferecem qualquer tipo de produto

Diante deste cenário, percebe-se que os *e-commerces* não conseguem exibir aos clientes todos os produtos disponíveis, pois é impossível apresentar milhões de itens a um usuário. Além disso, não se pode esperar que cada cliente tenha visto cada um dos produtos de interesse disponibilizados pela plataforma (LESKOVEC; RAJARAMAN; ULLMAN, 2014). Dadas essas condições, faz-se necessário que sejam recomendados itens de maior interesse de acordo com as preferências de cada usuário, surgindo os chamados sistemas de recomendação.

Sistemas de recomendação Sistemas de Recomendação (SR) são ferramentas que utilizam algoritmos de Inteligência Artificial Inteligência Artificial (IA) para prever produtos que um usuário pode ou não se interessar. Eles são aplicações de software utilizados em *e-commerces* para proporcionar uma experiência personalizada ao usuário através da análise de seu perfil e de suas compras passadas.

Com a expansão do volume e do aumento da complexidade dos dados na *Web*, a existência de sistemas de recomendação se tornou essencial para diversas aplicações utilizadas por um

usuário. Os sistemas de recomendações ajudam os usuários a superar o problema de sobrecarga de informações, expondo-os aos itens mais interessantes e relevantes (ZISOPOULOS *et al.*, 2008). A tecnologia de recomendações é, portanto, uma parte importante do problema de busca de informações que surgiu junto com a *Web*.

1.2 Objetivos

Este trabalho tem por objetivo implementar um sistema de recomendação para o *marketplace* de empregos domésticos Casa e Café a fim de proporcionar aos usuários maiores chances de contratação e uma melhor experiência ao utilizar a plataforma. Para isso, são coletados e analisados dados de profissionais e de contratantes cadastrados no *website*, o que foi autorizado pela empresa. Esta monografia também tem como finalidade inserir a aluna no tema de Sistemas de Recomendação, servindo como base para futuros projetos na área.

1.3 Organização

No capítulo 2, apresenta-se a teoria envolvendo a construção de um sistema de recomendação, como os conceitos e as técnicas de cada modelo e os algoritmos utilizados. Também são realizadas comparações entre esses modelos, apresentando suas vantagens e suas desvantagens. No capítulo 3, são descritas a metodologia e as ferramentas utilizadas no desenvolvimento do sistema de recomendação do Casa e Café, abordando, em detalhes, etapas como a análise de dados e a escolha do modelo de SR utilizado. Além disso, conforme são relatadas as fases de implementação deste trabalho, são apresentados os resultados obtidos. Por fim, no capítulo 4, são apresentadas informações sobre o curso de graduação, uma revisão do trabalho descrito nesta monografia e futuros projetos. Adicionalmente, o código desenvolvido para a construção do sistema de recomendação do Casa e Café é mostrado apêndice 5.

Capítulo 2

REVISÃO BIBLIOGRÁFICA

2.1 Considerações iniciais

Nesta seção, é apresentado o referencial teórico para o desenvolvimento de um SR, sendo discutido os modelos de SR existentes, os algoritmos de AM envolvidos e as técnicas para avaliar a qualidade das recomendações previstas.

2.2 Sistemas de Recomendação

Sistemas de recomendação foram propostos para que fosse solucionado o problema do grande volume de dados disponível ao usuário quando ele utiliza algum serviço da *web*. Dessa forma, a finalidade de um SR é melhorar e personalizar a experiência dessas pessoas. Para isso, ele deve realizar sugestões significativas para um conjunto de usuários dado uma coleção de produtos que pode interessá-los (MELVILLE; SINDHWANI, 2010).

Conforme dito em (BOBADILLA *et al.*, 2013), os SRs são inspirados pelo comportamento social humano, no qual leva-se em conta as preferências, as opiniões e as experiências de outros para se tomar uma decisão. O embasamento teórico para a implementação de um SR são apresentadas em 2.2.1, 2.2.2 e 2.2.3.

2.2.1 Estruturação dos Dados

Para o desenvolvimento de um sistema de recomendação, existem duas classes de entidades que são essenciais: usuários e itens (LESKOVEC; RAJARAMAN; ULLMAN, 2014). Os usuários representam os indivíduos a quem serão recomendados os produtos oferecidos por alguma aplicação *web*, os quais, por sua vez, são representados pela classe itens.

Cada uma dessas entidades é modelada como um perfil composto por atributos. Os atributos nada mais são do que características que descrevem a classe analisada. Por exemplo, se deseja-se criar um sistema de recomendação para um *e-commerce* de livros, é necessário estudar quais são as propriedades que melhor definem os usuários e os livros do *site*.

Uma provável escolha de atributos para a modelagem do perfil de um usuário seria: ID registrado no sistema, sexo, idade, ocupação e CEP. Já para a construção do perfil de um livro

poderiam ser escolhidos os seguintes atributos: ID registrado no sistema, categoria literária, título e autor.

A partir das avaliações que os usuários fazem sobre os itens disponibilizados, é possível construir uma matriz de utilidade. Por meio de uma matriz de utilidade, pode-se relacionar um usuário a um item, definindo o que é conhecido como o grau de preferência que um usuário possui por um item.

A avaliação pode ser definida de formas diferentes, dependendo da aplicação para a qual se está construindo o sistema de recomendação (AGGARWAL, 2016). As várias categorias de avaliação são descritas a seguir:

1. Avaliações contínuas: é utilizada uma escala contínua para especificar o grau de interesse que um item desperta em um usuário. Por exemplo, é possível proporcionar ao usuário uma escala de avaliação que varia entre -10 e 10. A desvantagem dessa abordagem é o incômodo causado no usuário ao forçá-lo a escolher um valor dentre inúmeras possibilidades;

2. Avaliações baseadas em intervalo: as avaliações são definidas em uma escala, em sua maioria, de 5 pontos ou de 7 pontos. Exemplos para esta categoria seriam valores de 1 a 5, de -2 a 2 ou de 1 a 7. Uma característica importante da avaliação baseada em intervalo é que a distância entre os valores são equidistantes, diferentemente da avaliação contínua;

3. Avaliações ordinais: utiliza-se valores nominais para definir os valores de avaliação, o que torna esta categoria muito parecida com a baseada em intervalo. Exemplos para os valores utilizados seriam "Gosto intensamente", "Não gosto intensamente", "Gosto", "Não gosto"e "Neutro". A principal diferença entre a avaliação ordinal e a baseada em intervalo é que, na primeira, não existe uma diferença definida entre os valores adjacentes. Porém, na prática, os valores nominais são interpretados como valores numéricos, o que deixa estas duas categorias extremamente similares;

4. Avaliações binárias: nesta categoria, existem apenas dois valores possíveis, os quais representam se o usuário gostou ou não gostou de determinado item. Quando o usuário é neutro em relação a algum item, apenas não se preenche com nenhum valor;

5. Avaliações unárias: apenas são especificadas avaliações de preferências positivas. Este tipo de avaliação é comumente originado de ações realizadas por um usuário. Por exemplo, o ato de um usuário comprar um item é uma ação de interesse positivo. Entretanto, se o usuário não realiza uma compra, pode-se entender que ele não gostou ou que é neutro ao item. A avaliação unária é interessante, pois permite a coleta de avaliações sem que o usuário tenha que dar uma nota aos itens, o que simplifica o desenvolvimento de sistemas de recomendação em aplicações que as avaliações explícitas são impossibilitadas.

Tabela 1 – Matriz de avaliação representando notas de livros em uma escala 1-5

Usuário	Livro 1	Livro 2	Livro 3	Livro 4	Livro 5
A	4			5	
B	5	4			2
C	1		3	4	
D		4		4	5

Na Tabela 1, está exemplificada uma matriz de utilidade para o caso da loja de livros virtual. Nela, relaciona-se um conjunto de usuários (A, B e C) a um conjunto de livros (1, 2, 3, 4 e 5) em uma escala de notas de 1 a 5 (avaliação ordinal).

Observando a Tabela 1, nota-se que a maioria dos elementos da matriz não está preenchida, o que significa que o usuário não avaliou boa parte dos itens disponíveis. Esta é uma característica comum de dados utilizados em sistemas de recomendação, pois o objetivo é, justamente, prever os espaços não preenchidos em uma matriz da utilidade.

Várias são as estratégias que podem ser utilizadas por um SR para prever quais itens devem sugeridos a um usuário, como: (1) recomendar itens que usuários similares achem relevante, (2) recomendar itens similares aos quais um usuário já se interessou no passado, (3) recomendar itens com base no contexto do usuário, (4) recomendar itens baseados em relações sociais e (5) recomendar itens com base no comportamento do usuário (CUNHA, 2016). Cada uma dessas técnicas será abordada na próxima seção.

2.2.2 Modelos e Técnicas

Diversos são os modelos que podem ser utilizados na implementação de um sistema de recomendação. As diferentes características que cada um apresenta permitem que inúmeras limitações impostas pelos dados de entrada sejam superadas, sendo possível elaborar um sistema de recomendação de alta qualidade. As técnicas existentes são descritas nas seções 2.2.2.1, 2.2.2.2 e 2.2.2.3.

2.2.2.1 Sistema de Recomendação de Filtragem Colaborativa

Neste modelo, as recomendações são realizadas por meio do poder colaborativo das avaliações que inúmeros usuários fazem sobre itens. O conceito básico dos métodos de FC é conseguir prever qual seria a nota que um usuário daria para um item que ele ainda não avaliou, para isso, utiliza-se de avaliações existentes.

A estrutura de dados utilizada no modelo de FC é conhecida como matriz de avaliação. Ela pode ser descrita como $R \approx U \times I$, sendo U o conjunto de usuários, em que $u \in 1...N$, e I o conjunto de itens, em que $i \in 1...M$. Cada elemento nesta matriz é uma representação numérica das avaliações que um usuário u realiza sobre um item i, designado por R_{ui} (CUNHA, 2016).

Diante destas características, os métodos de FC podem ser interpretados como generalizações de modelos de classificação e de regressão. Nestes, existe uma variável dependente (classe) que pode ser desconhecida, enquanto nos modelos FC, podem existir diversas colunas (atributos) com valores desconhecidos (AGGARWAL, 2016).

Em um problema de recomendação, não existe uma distinção clara entre variáveis classe e variáveis atributos, pois ambas desempenham os dois papéis. Dessa forma, as linhas de uma matriz de avaliação não podem ser categorizadas de teste ou de treinamento. Entretanto, quando se fala em classificação e em regressão, esta diferenciação existe. Na Figura 2, são comparados o problema de classificação/regressão com o de filtragem colaborativa.

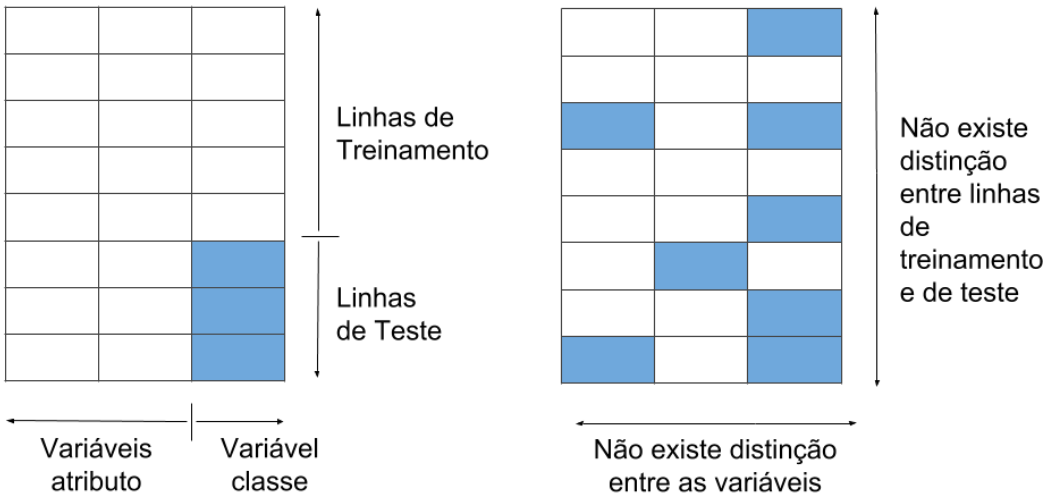

Figura 2 – Comparação entre a classificação tradicional e o problema de filtro colaborativo

Um dos maiores problemas enfrentados pelos sistemas de recomendação é escassez inicial de avaliações, comumente chamada de *cold start*. Novos itens e novos usuários representam um grande desafio por não terem gerado dados de avaliação, os quais são a essência do modelo de FC. Nesses casos, estas técnicas não são a melhor alternativa, sendo necessário utilizar, por exemplo, o modelo de Filtragem Baseada em Conteúdo (detalhado na seção 2.2.2.2).

Dentro do FC, existem dois tipos de métodos comumente utilizados: o Baseado em Memória (BME) e o Baseado em Modelo (BMO). No BME, são aplicadas heurísticas na matriz de avaliação para que sejam selecionados os itens que serão recomendados. No BMO, é induzido um modelo a partir da matriz de avaliação que gerará as recomendações como resultado. Cada um desses métodos é detalhado a seguir.

2.2.2.1.1 Filtragem Colaborativa Baseada em Memória

Neste método, também conhecido como Filtragem Colaborativa Baseada em Vizinhança, as recomendações são realizadas com base nos itens ou nos usuários mais similares (chamados de vizinhos mais próximos). Se for utilizada a similaridade entre os itens, categoriza-se o modelo como Baseado em Item (BI); se for utilizada a similaridade entre os usuários, categoriza-se o modelo como Baseado em Usuário (BU).

Uma outra diferença entre estes dois modelos é que, no modelo BU, utiliza-se as avaliações de vários usuários para se recomendar itens a um usuário alvo. Já no modelo BI, tem-se como base apenas as avaliações do usuário alvo para que se possa calcular os itens similares e, assim, serem realizadas as recomendações.

Independentemente do modelo utilizado, BU ou BI, existem três passos que generalizam o algoritmo que define o método BME (MELVILLE; SINDHWANI, 2010):

1. Computar a similaridade entre usuários ou itens
2. Selecionar os k usuários ou itens com a maior similaridade (vizinhança)
3. Gerar uma recomendação usando uma fórmula de predição

Para se calcular a similaridade entre usuários ou entre itens (passo 1), utiliza-se, tipicamente, umas dessas funções: Similaridade Cosseno (2.1), Correlação de *Pearson* (2.2) ou Distância Euclidiana (2.3). A seguir, estas funções são definidas matematicamente para dois usuários ou itens i e j com vetores v e w extraídos da matriz de avaliação, sendo $\overline{v}$ e $\overline{w}$ a média dos valores em cada vetor:

$$sim(i,j) = \frac{v.w}{|v||w|} \tag{2.1}$$

$$sim(i,j) = \frac{\sum_{i=1}^{k}(v_k - \overline{v})(w_k - \overline{w})}{\sqrt{\sum_{i=1}^{k}(v_k - \overline{v})^2 \sum_{i=1}^{k}(w_k - \overline{w})^2}} \tag{2.2}$$

$$sim(i,j) = \sqrt{\sum_{i=0}^{k}(v_k - w_k)^2} \tag{2.3}$$

Após o cálculo da similaridade, seleciona-se o grupo de k usuários ou de k itens que apresentaram os maiores valores (passo 2). Definido o grupo, é necessário recomendar os itens mais relevantes (passo 3). Para isso, precisa-se ponderar os valores de similaridade do grupo associando-os às avaliações existentes.

Este procedimento é essencial para que seja coberto o problema de usuários com diferentes percepções de avaliação, isto é, indivíduos dando notas diferentes para itens que despertam um interesse igual. Por exemplo, dada uma escala de 1 a 5, um usuário A pode avaliar um item de alto interesse com a nota 5, enquanto um usuário B pode avaliar o mesmo item com o mesmo interesse com uma nota 4.

A solução para esta particularidade é calcular a média das avaliações iniciais antes de determinar o valor médio da avaliação do grupo (AGGARWAL, 2016). A média centrada de um usuário u para um item i é definida pela subtração de sua avaliação média da sua avaliação inicial, assim como mostrado na Equação (2.4):

$$s_{uj} = r_{uj} - u_u, \forall u \in 1...m \tag{2.4}$$

Finalmente, é possível obter as fórmulas de predição para o caso dos modelos Baseado em Usuário (Equação 2.5) e Baseado em Item (Equação 2.6), a partir das quais é possível obter a avaliação de um usuário u para um item i:

$$pred(u,i) = \overline{R_u} + \frac{\sum_{n \subset vizinhos(u)} (sim(u,n))(R_{ni} - \overline{R_n})}{\sum_{n \subset vizinhos(u)} (sim(u,n))} \tag{2.5}$$

$$pred(u,i) = \frac{\sum_{j \in avaliados(u))} (sim(u,n))R_{ui})}{\sum_{n \subset avaliados(u)} (sim(u,n))} \tag{2.6}$$

A partir das características de cada um dos métodos, é possível extrair as principais vantagens e desvantagens que eles apresentam. Se analisado o modelo Baseado em Item, tem-se que sua principal vantagem é o fato das recomendações serem realizadas com base nas avaliações do próprio usuário alvo. Esta peculiaridade proporciona que as recomendações sejam, na maioria das vezes, extremamente relevantes (AGGARWAL, 2016). Em contrapartida, apesar de o método Baseado em Usuário apresentar recomendações menos relevantes mais frequentemente, ele oferece maior diversidade ao usuário. A diversidade é importante em um sistema de recomendação para que não ocorra a situação em que o usuário não goste de todos os itens recomendados por eles serem muito parecidos.

Uma outra vantagem do modelo BI é a estabilidade em relação a mudanças nas avaliações. A razão para isso está relacionada ao fato de que existem muito mais usuários do que itens, sendo mais fácil encontrar dois itens avaliados por muitos usuários do que dois usuários que avaliaram os mesmos itens. No caso do modelo BU, a simples adição de algumas avaliações pode causar uma drástica mudança nos valores de similaridade, o que não ocorre no método BI (AGGARWAL, 2016).

Um problema comum aos dois métodos é a complexidade computacional da fase *offline*, a qual se resume aos cálculos de similaridade (DS, 2017). Por exemplo, caso o número m de

Tabela 2 – Vantagens e Desvantagens das Técnicas de Filtragem Colaborativa Baseada em Memória

Técnica	Vantagens	Desvantagens
BI	Produz recomendações majoritariamente relevantes, Estabilidade a mudanças nas avaliações	Reduz a diversidade das recomendações, Alto custo computacional durante a fase *offline*, Dificuldade em lidar com a esparsidade da matriz de avaliação
BU	Proporciona alta diversidade de recomendações	Produz recomendações não relevantes com maior frequência, Vulnerável a mudanças nas avaliações, Alto custo computacional durante a fase *offline*, Dificuldade em lidar com a esparsidade da matriz de avaliação

usuários seja na ordem de milhões, a complexidade de um modelo BU seria no melhor caso representado por $O(m)$, o qual gera um tempo de execução tão grande que o SR se torna inviável de ser implementado.

Uma possível solução para este problema é a substituição da fase *offline* tradicional por uma de agrupamento, comumente conhecida por seu termo no idioma inglês: *clustering*. Nesta estratégia, realiza-se a criação de grupos, assim como ocorre no modo tradicional de vizinhos mais próximos, porém eles não são construídos em torno de um usuário alvo, o que reduz a complexidade computacional (AGGARWAL, 2016).

Outra desvantagem inerente aos métodos de Filtragem Colaborativa Baseada em Vizinhança é a dificuldade em lidar com a esparsidade da matrizes de avaliação, já que apenas uma pequena porção dos itens é avaliada por cada usuário. Uma alternativa para contornar este problema é o uso de métodos de redução de dimensionalidade, nos quais as matrizes de avaliação são representadas por meio de outra matriz de menor dimensão em termos de fatores latentes. Os modelos de fatores latentes melhoram os métodos de vizinhança tanto em termos de qualidade, quanto em termos de eficiência (AGGARWAL, 2016). Estes métodos serão abordados na seção seguinte.

As vantagens e as desvantagens mencionadas anteriormente são resumidas na Tabela 2.

2.2.2.1.2 Filtragem Colaborativa Baseada em Modelo

Neste método, as recomendações são realizadas por meio da estimativa de parâmetros a partir modelos estatísticos que utilizam as avaliações proporcionadas por usuários (MELVILLE; SINDHWANI, 2010). Para isso, utiliza-se de técnicas de Aprendizado de Máquina (AM) e de Mineração de Dados (MD) para se construir um modelo preditivo.

Os métodos de Filtragem Colaborativa Baseada em Modelo (FCBM) abrangem técnicas como árvores de decisão, modelos de regressão, métodos Bayesianos e modelos de fatores latentes. As três primeiras são tipicamente utilizadas em prolemas de classificação, porém são adaptadas para atender às necessidades dos problemas de recomendação.

Como um exemplo de adaptação, pode-se citar as árvores de decisão. O maior desafio em generalizar este algoritmo para um FC é distinguir as variáveis classe das variáveis atributos. A grande dificuldade reside no particionamento dos dados de treinamento durante a fase de

construção da árvore (AGGARWAL, 2016). Além disso, é difícil identificar qual das variáveis deve ser prevista, pois, como já foi mencionado, elas atuam concomitantemente como variáveis do tipo classe e do tipo atributo.

Para solucionar estes problemas, deve-se construir diferentes árvores de decisão para prever a avaliação de cada item e criar uma representação de menor dimensão para os dados utilizando um método de redução de dimensionalidade. Assim, se houverem m usuários e n itens, primeiramente, reduz-se a dimensionalidade da matriz de avaliação. Posteriormente, para cada usuário, interpreta-se cada item como sendo uma variável classe, enquanto os demais são variáveis atributos Modelos de Fatores Latentes (MFL) (KOREN; BELL; VOLINSKY, 2009).

Embora a generalização dos algoritmos de classificação seja uma boa alternativa para melhorar a eficiência e a qualidade dos métodos FC, o Modelo de Fatores Latentes (MFL) é a técnica mais robusta para o desenvolvimento de sistemas de recomendação atualmente. Nele, tenta-se relacionar itens e usuários por meio de fatores inferidos a partir de padrões identificados nas avaliações. A grande vantagem dessa técnica é não precisar que a matriz esteja inteiramente completa, ou seja, o problema da esparsidade da matriz de avaliação não afeta seu desempenho.

Uma das mais bem sucedidas maneiras de se implementar o MFL é por meio da fatorização de matrizes. Nesta técnica, relaciona-se usuários e itens a partir de padrões nas avaliações e, quando encontrada uma alta correlação entre eles, é realizada uma recomendação. Este método se tornou popular nos últimos anos por combinar uma boa escalabilidade com acurácia nas recomendações (KOREN; BELL; VOLINSKY, 2009).

Os métodos de Fatorização de Matrizes (FM) possibilitam que as correlações entre usuários e itens sejam obtidas de maneira elegante e de uma só vez, tornando-os o estado da arte em termos de FC (AGGARWAL, 2016). No modelo básico de FM, a matriz de avaliação R $m \times n$ é aproximada pela fatorização em uma matriz U $m \times n$ e em uma matriz V $m \times n$, como é mostrado na Equação 2.7.

$$R \approx UV^T \tag{2.7}$$

Cada coluna da matriz U (ou V) é designada como vetor ou componente latente e cada linha, como fator latente. A iésima linha $\overline{u_i}$ de U é referenciada como fator de usuário e contém k entradas correspondentes à afinidade do usuário i em relação aos k atributos da matriz de avaliação. Analogamente, linha $\overline{v_i}$ de V é chamada de fator de item e representa a afinidade do iésimo item em relação aos k atributos.

Como exemplo (adaptado de (AGGARWAL, 2016)), tem-se o caso explicitado na Figura 3, em que $\overline{u_i}$ é um vetor de dimensão dois que contém a afinidade do usuário i em relação aos gêneros literários de suspense e de romance na matriz de avaliação.

A partir da Equação 2.7, a avaliação r_{ij} de R pode ser aproximada como o produto

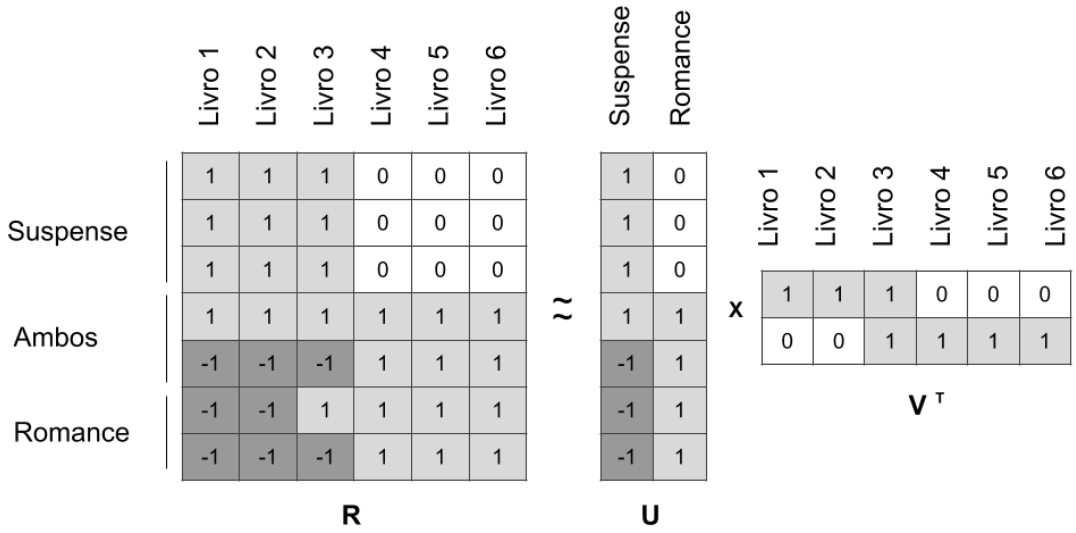

Figura 3 – Comparação entre a classificação tradicional e o problema de filtro colaborativo

escalar do iésimo fator de usuário com o jésimo fator de item, como apresentado na Equação 2.8:

$$r_{ij} \approx \overline{u_i} \cdot \overline{v_j} \tag{2.8}$$

Para que uma matriz de avaliação R completamente especificada corresponda às matrizes U e V fatorizadas, é necessário solucionar a Equação 2.9 como um problema de otimização (SAID; BELLOGíN, 2014):

$$Minimizar : J = \frac{1}{2} \parallel R - UV^T \parallel^2 \tag{2.9}$$

Na Equação 2.9, $\parallel . \parallel^2$ representa a norma Frobenius quadrática da matriz, a qual é igual à soma dos quadrados das entradas da mesma. Dessa forma, a função objetivo corresponde à soma dos quadrados das entradas da matriz residual ($R - UT^T$), podendo ser vista como uma função de perda quadrática, o que quantifica a perda de acurácia ao estimar a matriz R por meio da fatorização (AGGARWAL, 2016).

Como a Equação 2.9 foi elaborada dada uma matriz de avaliação R completamente preenchida, é necessário ajustá-la para que sejam consideradas entradas faltantes, assim como na maioria dos dados utilizados por um sistema de recomendação. Portanto, a função objetivo deve ser reescrita em termos das entradas observadas para sejam construídas as matrizes U e V.

Dado que o conjunto de todos os pares usuário-item observados em R é designado pela matriz diagonal S, as avaliações desconhecidas podem ser descritas pela Equação 2.10 (CUNHA, 2016):

$$r_{ij(p_{red})} = \sum_{s=1}^{k} u_{is} \cdot v_{js} \tag{2.10}$$

A diferença entre o valor de avaliação observado e o predito para uma entrada (i, j) é dado por $e_{ij} = r_{ij} - r_{ij(p_{red})} = r_{ij} - \sum_{s=1}^{k} u_{is}\dot{v}_{js}$. Assim, a função objetivo, que abrange matrizes de avaliação incompletas, é calculada apenas sobre as entradas observadas em S, como mostrado na Equação 2.11 (AGGARWAL, 2016):

$$Minimizar : J = \frac{1}{2} \sum_{(i,j) \in S} (r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js})^2 \tag{2.11}$$

A grande diferença entre os vários métodos de FM são as restrições impostas às matrizes U e V, como ortogonalidade e não-negatividade dos fatores latentes, e a natureza da função objetivo (BOKDE; GIRASE; MUKHOPADHYAY, 2015). Existem três principais técnicas utilizadas no desenvolvimento de modelos de FM que solucionam a função objetivo de uma FM: *Singular Value Decomposition* (SVD), *Stochastic Gradient Descent* (SGD) e *Alternating Least Squares* (ALS).

A técnica de SVD é utilizada apenas em casos em que a matriz de avaliação está completa, solucionando a Equação 2.9. Nela, uma matriz de avaliação R é decomposta em três outras matrizes, U, V^T e S, conforme é apresentado na Equação 2.12. As matrizes U e V são ortogonais com dimensões $m \times n$ e $n \times n$ respectivamente. Esta característica faz com que esta técnica tenha uma restrição de ortogonalidade.

A matriz S, chamada de matriz singular, tem dimensão $m \times n$ e é diagonal, sendo que suas entradas são números não negativos reais. Por meio de S, é possível obter os fatores mais importantes selecionando os maiores valores singulares. Assim, as matrizes U e V são reduzidas dimensionalmente mantendo as linhas e as colunas que se correspondem aos k valores singulares em S (CUNHA, 2016).

Após a aproximação utilizando os k fatores, a matriz de avaliação R_k é representa conforme mostrado na Equação 2.13

$$R \approx U \times S \times V^T \tag{2.12}$$

$$R_k \approx U_k \times S_k \times V_k^T \tag{2.13}$$

Tabela 3 – Características dos Métodos de Fatorização de Matrizes

Método	**Restrições**	**Vantagens**	**Desvantagens**
Sem restrições (SGD, ALS)	Não é imposta nenhuma restrição	Solução de alta qualidade, Bom desempenho para a maioria das matrizes	Complexidade
SVD	Bases ortogonais	Bom desempenho para matrizes densas	Solução de baixa qualidade para matrizes incompletas

A técnica SGD é caracterizada por não possuir restrições impostas às matrizes U e V, assim como a ALS, podendo ser executada para uma matriz incompleta. Naquela, minimiza-se a função 2.11 atualizando os parâmetros utilizados para cada entrada observada. No método ALS, aborda-se o problema de minimização da função objetivo como uma convergência para o mínimo valor de erro, o que o torna mais estável que o SGD (AGGARWAL, 2016).

Na técnica ALS, existem dois passos que levam à convergência do valor mínimo. No primeiro, mantém-se a matriz U fixa para cada linha da matriz V, tratando o problema como uma regressão de mínimos quadrados. No segundo, realiza-se o mesmo procedimento, porém, fixa-se a matriz V (CUNHA, 2016).

Comparando-se os diferentes métodos de se implementar a FM, é possível resumir as características observadas na Tabela 3 (adaptada de (AGGARWAL, 2016)):

2.2.2.2 *Sistema de Recomendação de Filtragem Baseada em Conteúdo*

Este modelo realiza as recomendações com base em itens que são similares àqueles que o usuário já se interessou no passado. A similaridade entre itens é calculada em relação aos atributos de seus perfis, quanto mais similar um item é de outro já avaliado positivamente pelo usuário, maiores são as chances dele ser recomendado (AGGARWAL, 2016).

Na estratégia de Filtragem Baseada em Conteúdo (FBC), os dados de usuários e de itens são encapsulados em um modelo de vetor de espaço, uma estrutura semelhante à matriz de utilidade. A similaridade entre os itens é dada pela similaridade dos vetores de espaço a quais estão associados (CUNHA, 2016), sendo calculada, principalmente, por meio das funções Distância Euclidiana (2.3) e Similaridade Cosseno (2.1).

Após serem calculados os valores de similaridade entre os vetores, estes são ordenados de maneira que os itens de maior interesse do usuário são recomendados, ou seja, aqueles que apresentam maior valor de similaridade. Diante desta técnica, pode-se amenizar o problema do *cold start* para novos itens, pois, diferentemente do FC, as informações de outros usuários não influenciam no processo de recomendação, apenas as características de cada item.

Ainda no modelo FBC, é possível recomendar itens a um usuário comparando-se os atributos destas duas entidades. Para isso, basta realizar os mesmos procedimentos apresentados anteriormente para os itens, porém, calcula-se a similaridade entre os vetores de espaço de um

Tabela 4 – Vantagens e Desvantagens do Modelo de Filtragem Baseada em Conteúdo

Vantagens	**Desvantagens**
Soluciona o problema de *cold start* para novos itens	Retorna recomendações "óbvias"
Apresenta simplicidade de implementação	Reduz diversidade das recomendações

usuário e de um item. Esta abordagem é realizada em situações *cold start* para novos usuários, pois, é necessário que os perfis de ambas entidades, usuário e item, sejam compostos pelos mesmos atributos. Por exemplo, baseando-se no caso do *e-commmerce* de livros, um usuário e um livro devem ser descritos pelos seguintes atributos: ID registrado no sistema, autor e gênero literário.

Diante desta estruturação dos dados, pode-se notar que existe pouca flexibilidade no modelo, o que não só acarreta em uma recomendação "óbvia", mas pode produzir um fenômeno chamado de *underfitting*, o qual é abordado na seção 2.2.3. Dadas estas características do modelo de FBC, podem ser listadas suas vantagens e desvantagens, as quais fomram resumidas na Tabela 4. O principal atrativo desta técnica é poder realizar recomendações para novos itens quando não há dados suficientes, solucionando o problema de *cold start* parcialmente. Outra vantagem deste modelo é a simplicidade de implementação.

Uma desvantagem do FBC é redução da diversidade das recomendações por causa do uso estrito do conteúdo para cálculo da similaridade entre itens. Por exemplo, se um usuário nunca consumiu um item com determinada característica, a chance de serem recomendados itens desta determinada característica é nula (AGGARWAL, 2016). Outra desvantagem é que as recomendações para novos usuários não são eficientes devido à pouca flexibilidade da estruturação dos dados.

2.2.2.3 Sistemas de Recomendação Híbridos

A ideia principal de um Sistema de Recomendação Híbrido (SRH) é solucionar os problemas que cada método apresenta por meio do uso das melhores estratégias que cada um deles pode oferecer (CUNHA, 2016). A combinação mais clássica para a implementação de um SRH é relacionar os modelos de FC e de FBC (BOBADILLA *et al.*, 2013).

Diversas são estratégias que podem ser escolhidas para relacionar aqueles dois modelos, mas a estrutura dos dados, quase sempre, é realizada com uma matriz de avaliação (obtida por meio do FC) e um modelo de espaço de vetor (obtido por meio do FBC) (BOKDE; GIRASE; MUKHOPADHYAY, 2015).

As diferentes combinações podem ser realizadas de alguma dessas formas por exemplo: implementação separada dos algoritmos relacionando apenas resultados, utilização do FBC durante alguma fase do FC e criação de um sistema de recomendação unificado que entrelaça ambos métodos.

Tabela 5 – Comparação dos Modelos de Sistema de Recomendação

Modelo	**Algoritmos**	**Vantagens**	**Desvantagens**
Baseada em Memória (Filtragem Colaborativa)	Baseada em Usuário, Baseada em Item	Fácil implementação, Fácil combinação com outros modelos de otimização	Problema de *cold start*, Dificuldade em lidar com matrizes de avaliação , Alto custo na fase offline
Baseada em Modelo (Filtragem Colaborativa)	SVD, SGD, ALS	Alta eficiência e acurácia, Boa solução para lidar com esparsidade e escalabilidade	Difícil implementação, Perda de informações na redução da dimensionalidade (SVD)
Baseada em Conteúdo	kNN	Boa solução para lidar com o problema de *cold start*, Fácil implementação	Baixa diversidade
Híbrido	Combinações entre Filtragem Colaborativa e Filtragem Baseada em Conteúdo	Boa solução para superar as limitações mais prejudiciais de cada modelo, Alta acurácia	Alta complexidade, Difícil implementação

Diante das características de cada modelo, é possível analisar suas vantagens e as desvantagens, como é apresentado na Tabela 5:

2.2.3 Métricas e Fatores de Avaliação

A avaliação de um sistema de recomendação é importante para que sejam conhecidos seu desempenho e sua efetividade (MELVILLE; SINDHWANI, 2010). Os principais tipos de avaliação são categorizados como *offline* e *online*. No paradigma *offline*, são utilizados históricos de dados para que sejam testados o desempenho e a qualidade de predição dos algoritmos desenvolvidos, sendo uma forma simples de avaliar os resultados gerados por um sistema de recomendação.

Entretanto, a estratégia *offline* tem a desvantagem de ser focada em analisar os resultados da recomendação, o que é prejudicial para entender os fatores humanos envolvidos no processo (CUNHA, 2016). Este problema pode ser facilmente resolvido se utilizada a estratégia *online*, a qual é adequada para coletar informações sobre as percepções dos usuários.

Dentro do paradigma *offline*, existem métricas que guiam a avaliação de um sistema de recomendação. A primeira delas é acurácia. A acurácia se refere à comparação dos resultados preditos com as avaliações reais, sendo a principal medida no processo de avaliação. (AGGARWAL, 2016). A maneira mais difundida de se calcular a acurácia é por meio do Erro Médio Absoluto (EMA) (definido na Equação 2.14) e da Raiz do Erro Quadrático Médio (REQM) (definida na Equação 2.15):

$$EMA = \frac{\sum_{i_j} | r_{pred} - r_{ij} |}{N} \tag{2.14}$$

$$REQM = \frac{\sqrt{\sum_{ij} \mid (r_{pred} - r_{ij})^2}}{N} \tag{2.15}$$

Outra métrica que pode ser utilizada na estratégia *offline* é a cobertura, a qual avalia as porções de itens que são recomendadas. A ideia é que essa métrica detecte se apenas uma pequena parte dos itens é recomendada, o que é algo prejudicial para o desempenho de um sistema de recomendação. Pode-se analisar tanto a cobertura em relação a itens, quanto em relação a usuários. O cálculo de cobertura é realizado comparando os dados de entrada e de saída (SHANI; GUNAWARDANA, 2013).

Além dessas duas principais métricas de avaliação, é importante mencionar que existem outros fatores que devem ser observados ao se implementar um sistema de recomendação, como a ocorrência de *overfitting* e de *underfitting*. Embora estes fenômenos não estejam diretamente relacionados a estratégias de avaliação, eles influenciam nos resultados de recomendação de maneira prejudicial e, portanto, devem ser identificados na fase de avaliação.

O *overfitting* se refere ao caso em que um modelo treina muito bem um conjunto de treinamento, isto é, quando um modelo aprende os detalhes e os ruídos do conjunto de testes, fazendo com que novos dados não se apliquem ao modelo construído. Este fenômeno ocorre, principalmente, em modelos não parametrizados e não lineares, nos quais existe uma grande flexibilidade de modelagem, como acontece em modelos de alta complexidade. Um exemplo de modelo que é bastante sujeito a *overfitting* são as Árvores de Decisão (BROWNLEE, 2016).

O *underfitting* ocorre quando um modelo não consegue extrair de maneira adequada informações do conjunto de treinamento, não sendo possível generalizar o modelo para novos dados (BRONSHTEIN, 2017). Este fenômeno geralmente acontece quando se utiliza um modelo muito simples, no qual não há variáveis atributos suficientes (BROWNLEE, 2016). O *underfitting* é fácil de detectar utilizando métricas de avaliação, como as citadas anteriormente, e deve ser solucionado testando diferentes algoritmos.

Para que seja construído um modelo que minimize a ocorrência desses fenômenos, é necessário balancear os valores de variação e de distorção e sua complexidade, a qual é medida, por exemplo, em termos de quantidade de variáveis de atributos e o tipo de algoritmo implementado. A variação mede quantas vezes o resultado da predição foi diferente do valor real e a distorção mede a distância entre eles (FORTMANN-ROE, 2017). Esta relação é evidenciada na Figura 4:

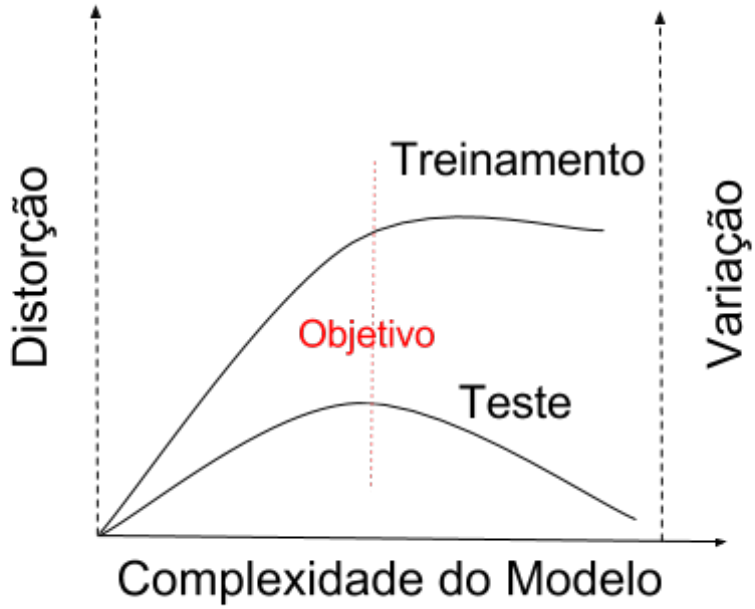

Figura 4 – Relação entre complexidade do modelo, variação e distorção

Capítulo 3

DESENVOLVIMENTO

3.1 Considerações Iniciais

Neste capítulo, são descritos os passos para a implementação do sistema de recomendação para o Casa e Café, sendo especificadas as técnicas e as decisões de projeto tomadas ao longo do desenvolvimento. Também são apresentados os resultados obtidos com a ferramenta desenvolvida, assim como as dificuldades enfrentadas em sua construção e possíveis melhorias para trabalhos futuros.

3.2 Atividades realizadas

Durante o desenvolvimento do projeto, foram realizados três passos principais: a manipulação dos dados, a escolha do modelo e a avaliação dos resultados. As duas últimas etapas ocorreram concomitantemente com a implementação do sistema de recomendação, pois, como é de se esperar, é necessário comparar as técnicas e os algoritmos para que se possa escolher o modelo ideal. Cada um dos passos é detalhado nas seções seguintes.

3.2.1 Coleta, Análise e Escolha dos dados

A primeira etapa na implementação de um sistema de recomendação é a escolha dos dados que irão compor os perfis das entidades de usuário e de item, assim como o tipo de avaliação, que pode ser, por exemplo, ordinal. Este passo envolve não só habilidades de análise de dados, mas exige que as regras de negócio da aplicação sejam muito bem conhecidas.

O Casa e Café é uma agência de empregos *online*, portanto existem dois tipos de *personas* utilizando a plataforma: contratantes e profissionais. Cada um deles pode interagir na plataforma em busca de profissionais e de vagas de trabalho respectivamente. Tanto os profissionais, quanto as vagas de trabalho são apresentadas na forma de perfis com informações relevantes ao interessado, como localização e pretensão salarial.

Diante destas características, estudou-se as *personas* que utilizam os serviços do Casa e Café e o fluxo de dados desde o cadastro até o momento da possível contratação. A partir disso, foram extraídos dados de profissionais e de vagas do Casa e Café de uma base de dados *MySQL* no formato de arquivo *CSV*. A extração foi realizada através de um *software* de código

aberto chamado *Pentaho*, que é utilizado para operações de integração de *big data* e soluções de *business intelligence*.

Posteriormente à extração de dados, foi desenvolvido um código na linguagem *Python* para facilitar sua visualização e sua manipulação, o qual pode ser encontrado na seção 5. Analisando as informações coletadas, foi possível analisar quais seriam os atributos que iriam compor os perfis de usuário e de item. Como mencionado anteriormente, a aplicação do Casa e Café permite que profissionais e contratantes realizem buscas, o que os caracterizariam ambos como usuários e como itens. Entretanto, por simplicidade, escolheu-se representar os profissionais como sendo itens, já que o número de castrados nessa categoria é maior.

Os dados coletados também foram analisados em forma de histogramas para que a distribuição dos valores de cada atributo fosse conhecida. Esta informação fez com que o segmento "Empregada Doméstica"(representada pelo valor um) fosse escolhido como o segmento de estudo para o desenvolvimento do sistema de recomendação (tipo de serviço mais procurado). Apenas um segmento foi escolhido, pois, no fluxo de busca da aplicação, o usuário especifica o tipo de serviço que deseja, não fazendo sentido ser avaliado mais de um segmento. Alguns dos histogramas referentes aos profissionais e aos contratantes são apresentados nas Figuras 5, 6, 7 e 8.

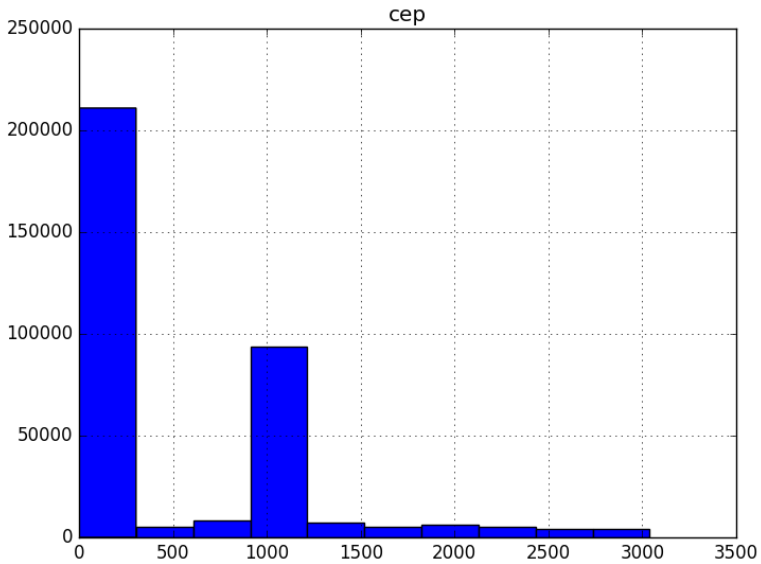

Figura 5 – Histograma de atributo "CEP"para o conjunto de dados do contratante

Por meio da análise dos dados, observou-se a existência de vários atributos nulos e a existência de informações similares em colunas diferentes, como CEP, bairro e cidade. Diante desta situação, foram excluídas as linhas e as colunas que não agregariam valor para a construção da ferramenta inicial.

Foi na fase de processamento dos dados também que, para cada contratante, foi elaborado

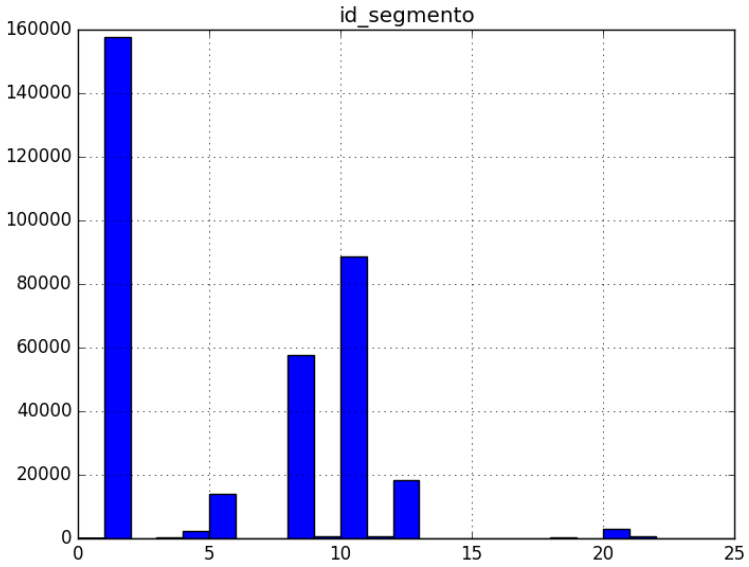

Figura 6 – Histograma de atributo "Segmento"para o conjunto de dados do contratante

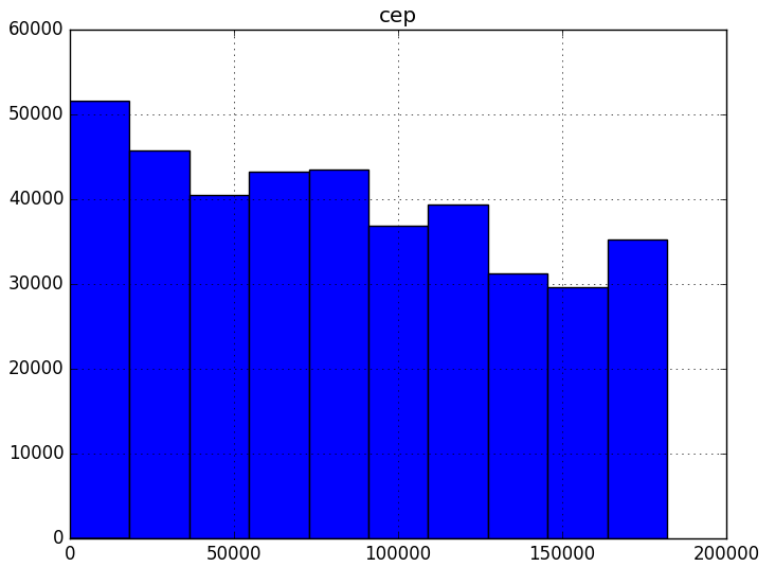

Figura 7 – Histograma de atributo "CEP"para o conjunto de dados do profissional

um *ranking* com os três primeiros profissionais com que eles entraram em contato. Esta informação é usada durante o processo de recomendação, porém, como ela é de alto custo computacional, este procedimento foi englobado nesta fase para que os resultados das recomendações fossem gerados no menor tempo possível.

O alto custo computacional se deve ao fato de que o conjunto de dados inteiro de contratantes é analisado para que sejam identificadas as ocorrências de "contato", isto é, é necessário analisar todas as linhas buscando pelas três primeiras que relacionam um mesmo

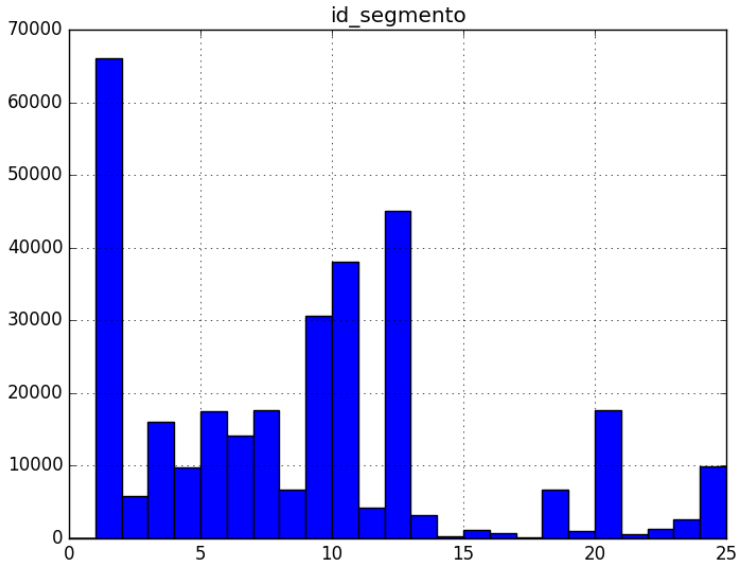

Figura 8 – Histograma de atributo "Segmento"para o conjunto de dados do profissional

contratante aos três primeiros profissionais com quem ele se relacionou.

Algumas das características do conjunto de dados de profissionais e de contratantes no decorrer do pré-processamento são apresentadas nas Tabelas 6 e 7.

Tabela 6 – Características Gerais dos Conjuntos de Dados de Contratante

Fase do Processamento	**Número de linhas no conjunto de contratantes**
Dados intactos	349845
Remoção de contratantes com pelo menos um atributo nulo	344929
Seleção de contratantes com atributo "segmento" igual a 1 (empregada doméstica)	157603
Remoção de relações duplicadas (várias linhas que se referiam a um mesmo par contratante-profissional)	35763
Remoção de profissionais que não possuiam profissionais ranqueados	14353
Remoção de pares contratante-ranking duplicados	1029

Tabela 7 – Características Gerais dos Conjuntos de Dados de Profissional

Fase do Processamento	**Número de linhas no conjunto de contratantes**
Dados intactos	396944
Remoção de profissionais com pelo menos um atributo nulo	222584
Seleção de profissionais com atributo "segmento" igual a 1 (empregada doméstica)	51604

Ainda no pré-processamento dos dados, alguns dos atributos, por exemplo "Frequência", são nominais, os quais não são interpretados por modelos de AM, nem por heurísticas de similaridade na linguagem *Python*, que foi utilizada para o desenvolvimento do sistema de

Tabela 8 – Atributos que compõem o perfil do item

Atributo	Descrição	Formato
Dormir	Designa se o profissional aceita dormir no trabalho	Boolean - 0 se não aceita dormir, 1 se aceita
Estado	Caracteriza o estado em que o profissional está disposto a trabalhar	String
Frequência	Descreve quantas vezes por semana ou por mês o profissional está disposto a trabalhar	String
ID Cidade	Representa o ID correspondente à cidade em que o profissional está disposto a trabalhar	Int
Pretensão Salarial	Designa a pretensão salarial do profissional	Float
Avaliação Geral	Representa a nota média recebida pelo profissional	Float
ID Segmento	Designa o ID do tipo de serviço do profissional	Float
ID Profissional	Representa o ID do profissional	Int

recomendação. Dessa forma, estes atributos foram transformados em valores numéricos. Os atributos finais que compõem os perfis do usuário e do item são descritos na Tabelas 9 e 8.

A última escolha a se fazer a respeito dos dados é referente às avaliações. Embora os profissionais terem sido modelados na base de dados com o campo "Avaliação Geral", apenas uma pequena porção foi realmente avaliada, além disso, o valor da nota é a média de todas as avaliações recebidas, o que torna inviável o uso de uma avaliação do tipo ordinal. Assim, escolheu-se a avaliação unária para representar os valores de avaliação, a qual capta interações dos usuários para descrever uma avaliação positiva. A ação dentro da plataforma que ponderou o interesse de um contratante foi o contato iniciado entre ele e um profissional, ou seja, se alguma conversa foi iniciada a partir de um contratante, uma avaliação positiva é caracterizada.

3.2.2 Escolha, Implementação e Avaliação do Modelo

Após a etapa de análise e de processamento dos dados, o segundo passo é a escolha do modelo de sistema de recomendação. Antes de qualquer implementação, particionou-se os dados em conjuntos de teste e de treinamento. Na maioria dos testes, a proporção de particionamento foi de 70% dos dados para treinamento e 30% para teste.

Para que o sistema de recomendação desenvolvido minimizasse o problema de *cold start*

Tabela 9 – Atributos que compõem o perfil do usuário

Atributo	**Descrição**	**Formato**
Dormir	Designa se o contratante deseja que o profissional durma no trabalho	Boolean - 0 se não aceita dormir, 1 se aceita
Estado	Caracteriza o estado do contratante	String
Frequência	Descreve quantas vezes por semana ou por mês o contratante deseja o serviço	String
ID Cidade	Representa o ID correspondente à cidade do contratante	Int
ID Contratante	Representa o ID do contratante	Int
ID Segmento	Designa o ID do tipo de serviço procurado pelo contratante	Float
Latitude	Valor da latitude do local que o contratante reside	Float
Longitude	Valor da longitude do local que o contratante reside	Float
Pretensão Salarial	Designa a pretensão salarial do profissional	Float
Nota Vaga	Corresponde à nota média dada pelos profissionais à vaga	Float

e, ao mesmo tempo, produzisse recomendações de qualidade, pensou-se em combinar as técnicas de Filtragem Colaborativa Baseada em Memória e de Filtragem Baseada em Conteúdo.

Assim, para cada usuário, selecionou-se outro usuário que fosse o mais similar possível com ele, sendo executado o algoritmo kNN com k igual a um. Para tanto, foram utilizadas as técnica de Filtragem Colaborativa BU e de Filtragem Baseada em Conteúdo, caracterizando o modelo como híbrido.

Posteriormente à seleção do usuário mais semelhante, foram recomendados os seus três profissionais ranqueados ao usuário alvo (quem elegeu aquele o mais similar). Este três profissionais recomendados constituíram um segundo *ranking*. Por fim, para avaliar a qualidade da predição, comparou-se a similaridade de cada profissional recomendado com os profissionais realmente escolhidos por meio da distância euclidiana.

Nessa comparação, utilizou-se dados experimentais para distinguir quais seriam as distâncias que representariam uma similaridade baixa ou alta. Para isso, calculou-se o valor da distância euclidiana para profissionais com atributos bastante ou pouco distintos. Por exemplo, comparou-se um profissional que não aceita dormir no trabalho e que mora na cidade do Rio de Janeiro com outro que aceita dormir e mora na cidade de São Paulo. Os usuários que serviram

Tabela 10 – Perfis dos profissionais que serviram como base para a obtenção dos valores de distância de referência para a avaliação das recomendações.

ID Profissional	77824	77825	153943	153946	380130
CEP	04836090	21042150	40252400	13059050	06410340
Cidade	São Paulo	Rio de Janeiro	Salvador	Campinas	Barueri
Dormir	0	0	0	1	1
Estado	SP	RJ	BA	SP	SP
Frequência	mensalista _2x	folguista	mensalista	mensalista	diarista _30
Bairro	Jardim Floresta	Bonsucesso	Cosme de Farias	Jardim Florence	Vila Porto
Latitude	-23.9	-22.9	-12.9	-23.5	-23.5
Longitude	-46.8	-43.3	-38.5	-47.1	-46.9
Pretensão Salarial	1500	900	2300	1200	1100

Tabela 11 – Matriz que contém os valores da distância euclidiana para cada par de profissionais

	1	**2**	**3**	**4**	**5**
1	0	4132930.00632	10256000.52114	3927999.03118	2326059.07127
2		0	13954000.42138	5132930.00631	3747956.05297
3			0	9454320.42316	10028967.8233
4				0	3324569.04118
5					0

de base para este procedimento são apresentados na Tabela 10 e os valores correspondentes a suas distâncias são apresentados na Tabela 11.

Diante desses dados, observou-se que os atributos de localização tem a maior influência nos valores de distância resultantes. Assim, estabeleceu-se que uma boa recomendação seria caracterizada se a distância entre os profissionais recomendados e escolhidos fosse menor que 2200000.0. Maiores detalhes do método de avaliação são descritos na seção 3.3.

Após a execução de um conjunto de testes ($\approx$ 50 execuções) , a acurácia obtida para este modelo foi alta. Porém, analisando os dados, foi possível identificar que, durante a etapa de recomendação, passaram a existir linhas referentes aos mesmos contratantes, ou seja, linhas com informações duplicadas. Este fato fez com que o cálculo da acurácia fosse afetado, já que um mesmo contratante era analisado várias vezes. Além disso, o grande número de atributos considerados no cálculo de similaridade estava adicionando uma flexibilidade muito alta ao modelo, deixando-o propenso a ser afetado por *overfitting*.

Outra falha detectada na primeira versão foi deixar de normalizar os atributos considerados nos cálculos de similaridade, o que foi corrigido na segunda versão normalizando os atributos antes de qualquer computação. Para que fossem solucionados os demais problemas, eliminou-se as linhas duplicadas e reduziu-se para cinco o número de atributos avaliados no cálculo de similaridade, os quais foram "Dormir", "Estado", "Frequência", "Cidade"e "Pretensão Salarial". Ainda, implementou-se a possibilidade de se escolher o valor de k utilizado no algoritmo kNN

para serem selecionados mais de um usuário que contribuirão com seus profissionais escolhidos para a formação do *ranking* de recomendação, tornando o sistema de recomendação mais robusto.

Diante destas modificações, o cálculo da similaridade entre os usuários também foi alterado. Nesta segunda versão, verifica-se, para os atributos "Dormir", "Estado", "Frequência"e "Cidade", se o usuários analisados possuem exatamente os mesmos valores. Se os atributos tiverem valores iguais, a distância é igual a zero, caso contrário, a distância é igual a um.

No caso do atributo "Pretensão Salarial", utilizou-se a distância euclidiana para obter o valor de similaridade relativo a este campo. Dada esta forma de calcular a similaridade, observa-se que a distância máxima é igual a cinco e, a mínima, igual a zero. Dessa maneira, nota-se que a similaridade é computada parcialmente, sendo sempre relativa à um campo, diferentemente do que foi realizado na primeira versão. Este artifício permite que seja mais fácil analisar as interferências individuais de cada campo, mesmo que eles tenham o mesmo peso no cálculo da similaridade.

Após a seleção dos k usuários mais próximos, tem-se a etapa de recomendação. Nela, pondera-se a distância e a colocação no *ranking* de um profissional. Mais especificamente, para cada um dos k usuários, calcula-se uma pontuação para cada um de seus profissionais baseando-se na distância daquele em relação ao usuário alvo (a quem se está fazendo a recomendação) e na colocação desses no *ranking* de escolha.

Formalmente, a ponderação se dá pela Equação 3.1 dado um contratante i a quem se deseja recomendar um profissional, um contratante j que foi selecionado como um dos k mais próximos em relação ao contratante i e um profissional l, o qual é um dos três profissionais com quem o contratante j entrou em contato:

$$Pontuacao(i,j,k) = \frac{1}{distancia(i,j) \cdot ranking(j,k)} \tag{3.1}$$

Após serem calculadas todas as pontuações, selecionou-se os três profissionais que obtiveram a maior pontuação para compor o *ranking* de recomendação.

A partir do resultado da primeira versão do modelo, também detectou-se que o procedimento para discernir se foi uma boa recomendação estava com pouco rigor. Diante desta situação, implementou-se um gerador de recomendações aleatórias para que elas pudessem servir como um *baseline* durante a avaliação. Nesse gerador, são atribuídos três quaisquer profissionais ao *ranking* de recomendação de cada contratante, desde que eles sejam diferentes daqueles já escolhidos.

Para avaliar a qualidade das recomendações, primeiramente, calculou-se a similaridade entre os profissionais do *ranking* escolhido e os do *ranking* de recomendação, tanto para o conjunto de dados real, quanto para o aleatório. O cálculo realizado para este procedimento foi o mesmo para se obter os k contratantes mais próximos, ou seja, leva em consideração os atributos

Tabela 12 – Parcela dos dados após cálculo de similaridade entre profissionais escolhidos e recomendados

Contratante	Soma dos Valores de cada Profissional Recomendado - CR	Soma dos Valores de cada Profissional Recomendado - CB
179	6	11
19	5	7
2653	8	13
1860	3	8

"Dormir", "Estado", "Cidade", "Frequência"e "Pretensão Salarial".

Após o cálculo de similaridade anterior, soma-se os valores de cada recomendado e armazena-se o resultado em um atributo da estrutura de contratante. Como são três profissionais recomendados e a distância máxima, como mencionada anteriormente, é igual a cinco, para cada contratante, a distância máxima total é igual a quinze. A mínima é igual a zero.

Diante desta situação, analisa-se, para cada contratante, se a distância total de seus profissionais é mais próxima do valor máximo (quinze) ou do valor mínimo (zero). Caso ela seja mais próxima do mínimo, considera-se um boa recomendação ("acerto"), caso contrário, o acerto não é contabilizado. Este procedimento é realizado para as recomendações reais e para as aleatórias. Após todos os contratantes passarem por essa análise, computa-se o total de acertos em relação ao total de recomendações realizadas. O valor obtido por meio desse cálculo é o valor da acurácia do modelo.

Além dessa avaliação, foram somados todos os valores de similaridade dos profissionais recomendados de cada contratante para se comparar a média dos mesmos para o conjunto de dados real e para o de *baseline*. Como exemplo, apresenta-se na Tabela 12 uma parcela dos dados para que esta métrica possa ser compreendida, em que "CR"designa conjunto de dados real e "CB"designa conjunto de dados de *baseline*:

Como pode ser visto, cada contratante possui um valor que representa a soma de todos os valores de similaridade dos profissionais recomendados. Assim, calculando-se a média da soma desses valores para cada conjunto, é possível avaliar qual das recomendações foi a mais eficiente. Para esta pequena amostra, o conjunto de dados real, no qual foram aplicados todos os métodos do sistema de recomendação, obteve uma média igual a 5. O conjunto de dados de *baseline*, no qual foram realizadas recomendações aleatórias, a média foi igual 9. Portanto, os resultados do sistema de recomendação foram superiores aos aleatórios.

3.3 Resultados Obtidos

Como mencionado na seção 3.2.2, foram desenvolvidas duas versões para o sistema de recomendação do Casa e Café. Na primeira, para se obter a acurácia do modelo, cada profissional

Tabela 13 – Resultados das avaliações para o modelo final

k	Acurácia (%) - CR	Acurácia (%) - CB	Distância - CR	Distância - CB
1	92.4	89.9	6.00200	22.00018
2	95.3	90.1	3.00004	21.00014
3	91.2	90.1	8.00006	21.00026
4	92.7	91.4	10.00002	42.00040
5	91.0	89.1	8.00002	20.00019
6	92.3	88.9	7.00008	13.00015
7	95.9	91.0	5.00003	11.00026
8	95.4	82.0	6.00002	56.00041
9	95.7	89.0	6.00004	34.00010
10	97.5	89.6	2.00003	24.80023
Média	94,0	89.1	6.10024	21.00037

recomendado foi comparado com cada profissional escolhido.

Na comparação, analisou-se se a distância euclidiana entre o par de profissionais era menor do que o valor 2200000.0. Caso ela fosse, contava-se uma recomendação ("acerto"); caso não fosse, a recomendação não era contada. Quando todas as recomendações de todos os contratantes foram analisadas, calculou-se a porcentagem de "acertos"em relação ao total de recomendações realizadas. Esta porcentagem é o valor da acurácia do modelo.

Após a execução de um conjunto de testes (≈ 50), a acurácia média obtida foi igual a 86%. Entretanto, foram observados vários problemas na forma como foi realizado o cálculo de avaliação, assim como foi explicitado na seção 3.2.2. Esta característica interferiu diretamente no valor da acurácia obtido. Para que este problema fosse solucionado, realizou-se as medidas mencionadas na seção 3.2.2, as quais consistiram em:

- Normalizar os dados antes de qualquer computação;
- Remover as linhas duplicadas após a recomendação;
- Diminuir o número de atributos analisados no cálculo de similaridade;
- Permitir que fosse possível escolher o valor de k para a execução do algoritmo kNN;
- Melhorar a forma de avaliar a qualidade das recomendações

Diante destas alterações, executou-se o modelo para diferentes valores de k, variando de 1 a 10. Para cada um desses valores, foram executados 30 testes para se obter a acurácia média do modelo e as distâncias médias referentes à similaridade dos profissionais recomendados para os dados reais e para os de *baseline*. Os resultados obtidos são resumidos na Tabela 13, sendo "CB"a designação de conjunto de dados de *baseline* e, "CR", a de conjunto de dados real.

Analisando os resultados da Tabela 13, verifica-se que ambos conjuntos de dados apresentaram uma alta acurácia média. Nota-se, também, que quanto menores os valores das distâncias

médias mais alta é a acurácia, o que é compatível com o esperado, pois maior será o número de acertos quanto mais similares forem os profissionais recomendados dos escolhidos. Ainda relacionando as duas métricas de avaliação, percebe-se que, quanto maior a diferença entre os valores das distâncias médias dos dois conjuntos de dados, maior será a diferença entre os valores de acurácia médios. Esta tendência também era esperada visto que as duas grandezas estão diretamente relacionadas.

Mesmo que os valores de acurácia e de distância médios tenham apresentado um comportamento como o previsto, corroborando a relação entre ambas métricas, verificou-se que a acurácia não conseguiu evidenciar de forma expressiva a diferença na qualidade das recomendações reais das de *baseline*. Como os valores de acurácia foram todos muito próximos (diferença máxima de 7,9% no se k igual a 10), é difícil visualizar que as recomendações feitas pelo modelo foram superiores às de *baseline*. Assim, nota-se que o cálculo da acurácia deve ser aprimorado para ser mais sensível à diferença entre os valores das distâncias médias entre os dois conjuntos de dados.

Também é possível notar que, de maneira geral, o aumento do valor de k resulta em melhores recomendações, o que era esperado, pois a diversidade de dados utilizados é maior, o que permite o aumento da cobertura nos experimentos. Observando a média dos valores das métricas para todos os valores de k (última linha da Tabela 13), nota-se que o sistema de recomendação gerou melhores recomendações se comparado ao de *baseline*, o era o objetivo deste projeto.

3.4 Dificuldades, limitações e trabalhos futuros

No decorrer do projeto, algumas dificuldades foram enfrentadas. A principal delas foi realizar a análise e o processamento dos dados, pois exigia um alto grau de conhecimento das técnicas de sistemas de recomendação para que a estrutura ideal pudesse ser escolhida, o que é algo difícil de se alcançar ainda no início do projeto. Além disso, grande parte informações extraídas da base de dados possuía campos nulos, o que inviabilizou o uso de certos atributos para a construção dos perfis de usuário e de item.

Diante desta situação, várias falhas ocorreram na implementação das técnicas, resultando em recomendações muito ruins inicialmente. Entretanto, com o passar do tempo os modelos de sistemas de recomendação foram compreendidos e suas implementações dominadas, o que proporcionou a superação desta dificuldade. Uma limitação ainda relacionada aos dados é em relação ao tipo de avaliação utilizada.

Como não estão disponíveis avaliações do tipo ordinal na aplicação, foi necessário utilizar a informação de quais profissionais foram contatados por quais contratantes para caracterizar uma avaliação positiva. Entretanto, esta forma de captar as avaliações unárias é muito restritiva, pois um usuário pode ter se interessado por um profissional mesmo não entrando em contato

com ele. Frente às dificuldades enfrentadas durante o desenvolvimento do projeto, o tempo para aprimoramento do modelo final foi extremamente curto, o que inviabilizou a implementação de outras maneiras de calcular a similaridade e de avaliar a qualidade das recomendações. Estas modificações permitiriam testes mais completos e colaborariam para a obtenção de melhores resultados.

Para que o sistema de recomendação seja refinado, é necessário testar mais modelos e compará-los com os resultados atuais. Com isso, será possível promover mudanças nos perfis que, potencialmente, gerarão melhores recomendações. Um exemplo de implementação futura é a técnica de Filtragem Colaborativa Baseado em Modelo, o qual é mais complexo de desenvolver, mas é o estado da arte em sistemas de recomendações.

Capítulo 4

CONCLUSÃO

Neste capítulo, são apresentadas considerações relacionadas ao curso de Engenharia de Computação e contribuições proporcionadas por este projeto.

4.1 Considerações sobre o curso

O curso de Engenharia de Computação, oferecido pela Escola de Engenharia de São Carlos (EESC) e pelo Instituto de Ciências Matemáticas e de Computação (ICMC), proporciona ao estudante um conhecimento abrangente nas áreas de Ciência da Computação e de Engenharia Elétrica. Esta característica possibilita que o aluno seja versátil quando exposto a diversos problemas, podendo contar com um embasamento teórico extenso.

A desvantagem desta particularidade é a que a grade curricular dificulta que sejam desenvolvidos projetos diferentes dos exigidos pela graduação, o que empobrece a maneira como o conhecimento é absorvido durante o curso. Muitas vezes as disciplinas são focadas nos aspectos teóricos, deixando de lado o aprendizado prático.

4.2 Contribuições

Nesta monografia, desenvolveu-se um sistema de recomendação para a agência de empregos *online* Casa e Café na linguagem *Python* utilizando um modelo híbrido. Para isso, estudou-se a literatura para entender as técnicas envolvidas na implementação de um sistema de recomendação e formas de analisar os dados de entrada. Durante a elaboração do projeto, foram implementados vários modelos até que um respondesse com boas recomendações.

Devido às limitações encontradas na etapa de implementação deste trabalho, foi possível aprimorar os conceitos aprendidos sobre sistemas de recomendação e aprender a linguagem *Python* com ênfase para a área de Ciência de Dados e de Aprendizado de Máquina. Diante do trabalho desenvolvido, o sistema de recomendação será aprimorado conforme descrito na seção 3.4 para ele seja acoplado de fato à aplicação do Casa e Café.

REFERÊNCIAS

AGGARWAL, C. C. **Recommender Systems: The Textbook**. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2016. ISBN 3319296574, 9783319296579. Citado 10 vezes nas páginas 26, 28, 30, 31, 32, 33, 34, 35, 36 e 37.

BOBADILLA, J.; ORTEGA, F.; HERNANDO, A.; GUTIÉRREZ, A. Recommender systems survey. **Know.-Based Syst.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 46, p. 109–132, jul. 2013. ISSN 0950-7051. Disponível em: <http://dx.doi.org/10.1016/j.knosys.2013.03.012>. Citado 2 vezes nas páginas 25 e 36.

BOKDE, D.; GIRASE, S.; MUKHOPADHYAY, D. Matrix factorization model in collaborative filtering algorithms: A survey. **Procedia Computer Science**, v. 49, n. Supplement C, p. 136 – 146, 2015. ISSN 1877-0509. Proceedings of 4th International Conference on Advances in Computing, Communication and Control (ICAC3'15). Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050915007462>. Citado 2 vezes nas páginas 34 e 36.

BRONSHTEIN, A. **Train/Test Split and Cross Validation in Python**. 2017. <https://medium.com/towards-data-science/train-test-split-and-cross-validation-in-python-80b61beca4b6>. (Acessado em 28 de outubro de 2017). Citado na página 38.

BROWNLEE, J. **Overfitting and Underfitting With Machine Learning Algorithms**. 2016. <https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/>. (Acessado em 28 de outubro de 2017). Citado na página 38.

CHRIS, A. **The Long Tail: Why the Future of Business Is Selling Less of More**. [S.l.]: Hyperion, 2006. Citado na página 21.

CUNHA, T. D. S. **Model Management for Recommender Systems using Metalearning**. Tese (Doutorado) — Faculdade de Engenharia da Universidade do Porto, 2016. Citado 5 vezes nas páginas 27, 34, 35, 36 e 37.

DS thmsrey. **History of Ecommerce**. 2017. <http://thmsrey-ds.com/2017/07/21/recommender-systems-pros-and-cons-of-neighborhood-based-methods-2/>. (Acessado em 27 de outubro de 2017). Citado na página 30.

FORTMANN-ROE, S. **Understanding the Bias-Variance Tradeoff**. 2017. <http://scott.fortmann-roe.com/docs/BiasVariance.html>. (Acessado em 28 de outubro de 2017). Citado na página 38.

KOREN, Y.; BELL, R.; VOLINSKY, C. Matrix factorization techniques for recommender systems. 09 2009. Citado na página 32.

LAND, E. **History of Ecommerce**. 2008. <http://www.ecommerce-land.com/history_ecommerce.html>. (Acessado em 19 de outubro de 2017). Citado na página 21.

LESKOVEC, J.; RAJARAMAN, A.; ULLMAN, J. D. **Mining of Massive Datasets**. [S.l.]: Cambridge University Press, 2014. Citado 3 vezes nas páginas 21, 22 e 25.

MELVILLE, P.; SINDHWANI, V. Recommender systems. In: **Encyclopedia of Machine Learning**. [S.l.: s.n.], 2010. p. 829–838. Citado 4 vezes nas páginas 25, 29, 31 e 37.

SAID, A.; BELLOGíN, A. Comparative recommender system evaluation: Benchmarking recommendation frameworks. 09 2014. Citado na página 33.

SHANI, G.; GUNAWARDANA, A. Evaluating recommendation systems. 09 2013. Citado na página 38.

ZISOPOULOS, H.; KARAGIANNIDIS, S.; DEMIRTSOGLOU, G.; ANTARIS, S. Content-based recommendation systems. 11 2008. Citado na página 23.

Capítulo 5

CÓDIGOS IMPLEMENTADOS

```
# loading libraries
import pandas as pd
import numpy as np
import random
from sklearn.cross_validation import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
from sklearn.metrics.pairwise import euclidean_distances
from decimal import Decimal

pd.options.mode.chained_assignment = None  # default='warn'

def normalize(df):

  # Colunas que serao normalizadas
  featureList = ['pretensao'];

  result = df.copy();
  for feature_name in featureList:
      max_value = df[feature_name].max();
      min_value = df[feature_name].min();
      result[feature_name] = (df[feature_name] - min_value) / (max_value -
    min_value);
  return result;

def load_data():

  # Obtendo dados dos usuarios
  names = ['dormir', 'estado', 'frequencia', 'id_cidade', 'id_contratante',
     'id_segmento',
  'latitude', 'longitude', 'nota_vaga', 'pretensao', 'id_profissional', '
    ranking_1', 'ranking_2', 'ranking_3'];
  hirerDf = pd.read_csv('/home/elisa/TCC/test/cc_hirer-100.csv', header=
    None, names=names, delimiter=",");

  names = ['id_profissional', 'id_cidade', 'estado', 'nota_curriculo', '
    avaliacao_geral', 'id_segmento',
  'frequencia', 'pretensao', 'dormir'];
```

```
34   professionalDf = pd.read_csv('/home/elisa/TCC/test/cc_professional-100.
       csv', header=None, names=names, delimiter=",");
35
36   # Normalizando colunas
37   normalizedHirerDf = normalize(hirerDf);
38   normalizedProfessionalDf = normalize(professionalDf);
39
40   return normalizedHirerDf, normalizedProfessionalDf;
41
42 def check_equality(value1, value2):
43   if(value1 == value2):
44     return 0;
45   else:
46     return 1;
47
48 def knn(trainingSet, testSet, k):
49
50   columns = ['id_profissional', 'id_contratante', 'index', 'distancia', '
       ranking', 'score'];
51   distanceDf = pd.DataFrame(columns=columns);
52   neighborsDf = pd.DataFrame(columns=columns);
53
54   for row1 in testSet.itertuples():
55     for row2 in trainingSet.itertuples():
56       if (row2[5] != row1[5]):
57
58         # Campos em que semelhanca tem resultado discreto
59         sleepDistance = check_equality(row1[1], row2[1]);
60         stateDistance = check_equality(row1[2], row2[2]);
61         frequencyDistance = check_equality(row1[3], row2[3]);
62         cityDistance = check_equality(row1[4], row2[4]);
63
64         # Campos em que semelhanca tem resultado continuo
65         salaryDistance = euclidean_distances(row1[10], row2[10]);
66
67         totalDistance = sleepDistance + stateDistance + frequencyDistance +
       cityDistance + salaryDistance;
68
69         distanceDf.loc[len(distanceDf.index)] = [row2[12], row1[5], row1.
       Index, totalDistance, 1, 0];
70         distanceDf.loc[len(distanceDf.index)] = [row2[13], row1[5], row1.
       Index, totalDistance, 2, 0];
71         distanceDf.loc[len(distanceDf.index)] = [row2[14], row1[5], row1.
       Index, totalDistance, 3, 0];
72
73     res = distanceDf.sort_values(['distancia'],ascending=True).head(k*3);
74     neighborsDf = neighborsDf.append(res, ignore_index=True);
```

```
75      distanceDf.drop(distanceDf.index, inplace=True);
76
77    return neighborsDf;
78
79  def recommend(neighbors, testSet, k):
80
81    columns = ['id_profissional', 'id_contratante', 'index', 'score'];
82    scoreDf = pd.DataFrame(columns=columns);
83    index = 0;
84
85    for row1 in neighbors.itertuples():
86
87      if(row1[4] == 0):
88        score = (1/0.000000000000000001)/(row1[5]);
89      else:
90        score = (1/row1[4])/(row1[5]);
91
92      neighbors.set_value(row1.Index,'score', score);
93      id_contratante = row1[2];
94
95    for row1 in neighbors.itertuples():
96      totalScore = neighbors.loc[neighbors['id_profissional'] == row1[1], '
        score'].sum();
97      scoreDf.loc[len(scoreDf.index)] = [row1[1], row1[2], row1[3],
        totalScore];
98
99    res = scoreDf.sort_values(['score'],ascending=False).head(3);
100
101   testSet['ranking_1P'] = 0;
102   testSet['ranking_2P'] = 0;
103   testSet['ranking_3P'] = 0;
104   testSet['score'] = 0.00000000000000000000;
105
106   meanScore = res['score'].mean();
107
108   for row1 in testSet.itertuples():
109     if(row1[5] == id_contratante):
110       testSet.set_value(row1.Index,'score', meanScore);
111       testSet.set_value(row1.Index,'ranking_1P', res.iloc[0, 0]);
112       testSet.set_value(row1.Index,'ranking_2P', res.iloc[1, 0]);
113       testSet.set_value(row1.Index,'ranking_3P', res.iloc[2, 0]);
114       break;
115
116   return testSet.loc[(testSet['ranking_1P'] != 0) & testSet['id_contratante
      '].isin([row1[5]])];
117
118 def random_prediction(recommendedDf, professionalDf):
```

```
119
120   # Criando estrutura que contem profissionais recomendados aleatoriamente
121   columns = recommendedDf.dtypes.index;
122   randomPredictionDf = pd.DataFrame(columns=columns);
123   randomPredictionDf = randomPredictionDf.append(recommendedDf,
        ignore_index=True);
124
125   idList = [];
126
127   for row1 in randomPredictionDf.itertuples():
128     for i in range(3):
129       randomRowNumber = random.randrange(professionalDf.shape[0]);
130       randomProfessionalId = professionalDf.iloc[randomRowNumber,0];
131
132       idList.append(randomProfessionalId);
133     randomPredictionDf.set_value(row1.Index,'ranking_1P', idList[len(idList
        )-3]);
134     randomPredictionDf.set_value(row1.Index,'ranking_2P', idList[len(idList
        )-2]);
135     randomPredictionDf.set_value(row1.Index,'ranking_3P', idList[len(idList
        )-1]);
136
137   return randomPredictionDf;
138
139 def professional_similarity(recommended, chosen):
140
141   # Campos em que semelhanca tem resultado discreto
142   sleepDistance = check_equality(recommended['dormir'].values[0], chosen['
        dormir'].values[0]);
143   stateDistance = check_equality(recommended['estado'].values[0], chosen['
        estado'].values[0]);
144   frequencyDistance = check_equality(recommended['frequencia'].values[0],
        chosen['frequencia'].values[0]);
145   cityDistance = check_equality(recommended['id_cidade'].values[0], chosen
        ['id_cidade'].values[0]);
146
147   # Campos em que semelhanca tem resultado continuo
148   salaryDistance = euclidean_distances(recommended['pretensao'].values[0],
        chosen['pretensao'].values[0]);
149
150   totalDistance = sleepDistance + stateDistance + frequencyDistance +
        cityDistance + salaryDistance;
151
152   return totalDistance;
153
154 def calculate_accuracy_score(professionalDf, analisedDf):
155
```

```
156   columns = ['id_profissional', 'distance'];
157   similarityDf = pd.DataFrame(columns=columns);
158   distanceDf = pd.DataFrame(columns=columns);
159
160   for row1 in analisedDf.itertuples():
161     # Profissionais escolhidos pelo contratante
162     chosen1 = professionalDf[professionalDf.id_profissional.isin([row1
      [12]])];
163     chosen2 = professionalDf[professionalDf.id_profissional.isin([row1
      [13]])];
164     chosen3 = professionalDf[professionalDf.id_profissional.isin([row1
      [14]])];
165
166     for x in xrange(15,18):
167       predicted = professionalDf[professionalDf.id_profissional.isin([row1[
      x]])];
168
169       distance1 = professional_similarity(predicted, chosen1);
170       distance2 = professional_similarity(predicted, chosen2);
171       distance3 = professional_similarity(predicted, chosen3);
172       similarityDf.loc[len(similarityDf.index)] = [row1[x], distance1];
173       similarityDf.loc[len(similarityDf.index)] = [row1[x], distance2];
174       similarityDf.loc[len(similarityDf.index)] = [row1[x], distance3];
175
176     res = similarityDf.sort_values(['distance'],ascending=True).head(3);
177     totalDistance = res['distance'].sum();
178     distanceDf.loc[len(similarityDf.index)] = [row1[5], totalDistance];
179
180   return distanceDf;
181
182 def check_recommendation(recommendedDf, professionalDf):
183
184   # Criando recomendacoes aleatorias
185   randomDf = random_prediction(recommendedDf, professionalDf);
186
187   # Calculando pontuacao para verificar acuracia da recomendacao
188   randomDistanceDf = calculate_accuracy_score(professionalDf, randomDf);
189   recommendedDistanceDf = calculate_accuracy_score(professionalDf,
      recommendedDf);
190
191   recommendedCount = 0.00000000000000000000;
192   randomCount = 0.00000000000000000000;
193
194   for x in xrange(0, len(recommendedDistanceDf)-1):
195     # Contabilizando acertos da recomendacao
196     distanceToMax = 15 - recommendedDistanceDf.iloc[x,1];
197     distanceToMin = recommendedDistanceDf.iloc[x,1] - 0;
```

```
198     if(distanceToMin <= distanceToMax):
199       recommendedCount = recommendedCount + 1;
200
201     # Contabilizando acertos dos aleatorios
202     distanceToMax = 15 - randomDistanceDf.iloc[x,1];
203     distanceToMin = randomDistanceDf.iloc[x,1] - 0;
204     if(distanceToMin <= distanceToMax):
205       randomCount = randomCount + 1;
206
207   # Soma da coluna de distancia
208   totalRecommendedDistance = recommendedDistanceDf['distance'].sum();
209   totalRandomDistance = randomDistanceDf['distance'].sum();
210
211   total = len(recommendedDistanceDf);
212
213   recommendedScore = 0.00000000000000000000;
214   recommendedScore = recommendedCount/total;
215
216   randomScore = 0.00000000000000000000;
217   randomScore = randomCount/total;
218
219   print("Recomendacao: ", recommendedScore);
220   print("Baseline: ", randomScore);
221
222 # Obtendo dados dos usuarios
223 hirerDf, professionalDf = load_data();
224
225 # Separando em conjuntos de treinamento e de teste
226 trainingSet, testSet = train_test_split(hirerDf, test_size=0.33);
227
228 # Obtendo os k contratantes mais parecidos
229 k = 10;
230 neighborsDf = knn(trainingSet, testSet, k);
231
232 # Recomendando profissionais
233 columns = testSet.dtypes.index;
234 recommendedDf = pd.DataFrame(columns=columns);
235 recommendedDf['ranking_1P'] = 0;
236 recommendedDf['ranking_2P'] = 0;
237 recommendedDf['ranking_3P'] = 0;
238 recommendedDf['score'] = 0;
239
240 for row1 in neighborsDf.itertuples():
241   candidates = neighborsDf.loc[neighborsDf['id_contratante'].isin([row1
      [2]])];
242   if(candidates.empty == False):
243     result = recommend(candidates, testSet, k);
```

```
244     recommendedDf = recommendedDf.append(result, ignore_index=True);
245
246 # Removendo linhas duplicadas apos recomendacao
247 recommendedDf = recommendedDf.drop_duplicates(subset=['id_contratante'],
        keep='last');
248
249 # Calculando acuracia do modelo
250 check_recommendation(recommendedDf, professionalDf);
```

Código-fonte 1: recommender-system.py

```
1  # loading libraries
2  import pandas as pd
3  import numpy as np
4  from sklearn import preprocessing
5  from sklearn.preprocessing import LabelEncoder
6  from sklearn.preprocessing import StandardScaler
7
8  pd.options.mode.chained_assignment = None  # default='warn'
9
10 def loading_data():
11   # Obtendo dados dos usuarios
12   names = ['cep', 'cidade', 'dormir', 'endereco', 'estado', 'experiencia',
       'frequencia', 'id_cidade', 'id_contratante', 'id_segmento',
13     'latitude', 'longitude', 'nota_vaga', 'pretensao', 'id_profissional'];
14   hirerDf = pd.read_csv('/home/elisa/TCC/user_hirer2.csv', header=None,
       names=names, delimiter=";");
15
16   # Obtendo dados dos profissionais
17   names = ['id_profissional', 'cep', 'bairro', 'id_cidade', 'cidade', '
       id_estado', 'estado', 'latitude',
18   'longitude', 'endereco', 'nota_curriculo', 'avaliacao_geral', '
       id_segmento', 'frequencia', 'pretensao', 'pretensao_dormir',
19   'dormir'];
20   professionalDf = pd.read_csv('/home/elisa/TCC/professional_profile2.csv',
       header=None, names=names, delimiter=";");
21
22   return hirerDf, professionalDf;
23
24 def data_preprocessing(hirerDf, professionalDf):
25
26   # Removendo colunas que adicionam muitos atributos nulos e/ou que nao sao
       utilizadas
27   hirerDf.drop('cep', axis=1, inplace=True);
28   hirerDf.drop('cidade', axis=1, inplace=True);
29   hirerDf.drop('endereco', axis=1, inplace=True);
30   hirerDf.drop('experiencia', axis=1, inplace=True);
31
```

```
32 professionalDf.drop('cep', axis=1, inplace=True);
33 professionalDf.drop('bairro', axis=1, inplace=True);
34 professionalDf.drop('cidade', axis=1, inplace=True);
35 professionalDf.drop('id_estado', axis=1, inplace=True);
36 professionalDf.drop('latitude', axis=1, inplace=True);
37 professionalDf.drop('longitude', axis=1, inplace=True);
38 professionalDf.drop('endereco', axis=1, inplace=True);
39 professionalDf.drop('pretensao_dormir', axis=1, inplace=True);
40
41 # Removendo linhas que contem pelo menos um dos atributos nulo
42 hirerDf = hirerDf.dropna();
43 professionalDf = professionalDf.dropna();
44
45 # Removendo linhas que nao sao do segmento de 'Empregada Domestica'
46 hirerDf = hirerDf[hirerDf.id_segmento == 1];
47 professionalDf = professionalDf[professionalDf.id_segmento == 1];
48
49 # Removendo linhas que relacionam um contratante mais de uma vez a um
    mesmo profissional
50 hirerDf = hirerDf.drop_duplicates(subset=['id_contratante', '
    id_profissional', 'id_segmento'], keep='first');
51
52 # Restringindo numero de amostras analisadas
53 hirerDf.drop(hirerDf.index[50002:], inplace=True)
54
55
56 for row1 in hirerDf.itertuples():
57   if(professionalDf[professionalDf.id_profissional.isin([row1[11]])].
    empty == True):
58     hirerDf = hirerDf[hirerDf.id_profissional != row1[11]];
59
60 # Adicionando colunas para serem armazenados os profissionais rankeados
61 hirerDf['ranking_1'] = 'default';
62 hirerDf['ranking_2'] = 'default';
63 hirerDf['ranking_3'] = 'default';
64
65 for row1 in hirerDf.itertuples():
66   value_list = [row1[5]]
67   data = hirerDf[hirerDf.id_contratante.isin(value_list)];
68   if (len(data) >= 3):
69     if (row1[5] == data.iloc[0, 4]):
70       hirerDf.set_value(row1.Index,'ranking_1',(data.iloc[0,10]))
71       hirerDf.set_value(row1.Index,'ranking_2',(data.iloc[1,10]))
72       hirerDf.set_value(row1.Index,'ranking_3',(data.iloc[2,10]))
73
74 # Removendo linhas que nao possuem profissionais rankeados
75 hirerDf = hirerDf[hirerDf.ranking_1 != 'default'];
```

```
76
77    # Tranformando campos nominais em numericos
78    le = LabelEncoder()
79    hirerDf['dormir'] = le.fit_transform(hirerDf['dormir'])
80    hirerDf['estado'] = le.fit_transform(hirerDf['estado'])
81    hirerDf['frequencia'] = le.fit_transform(hirerDf['frequencia'])
82    professionalDf['estado'] = le.fit_transform(professionalDf['estado'])
83    professionalDf['frequencia'] = le.fit_transform(professionalDf['
        frequencia'])
84
85    # Removendo linhas duplicadas apos rankeamento
86    hirerDf = hirerDf.drop_duplicates(subset=['id_contratante', 'ranking_1',
        'ranking_2', 'ranking_3',
87     'id_segmento', 'id_cidade'], keep='last');
88
89    return hirerDf, professionalDf;
90
91 # Inicializando dados do contratante
92 hirerDf, professionalDf = loading_data();
93
94 hirerDf, professionalDf = data_preprocessing(hirerDf, professionalDf);
95
96 hirerDf.to_csv('cc_hirer-1000t.csv');
97 professionalDf.to_csv('cc_professional-1000t.csv');
```

Código-fonte 2: data-processing.py