GUILHERME APARECIDO GREGORIO

Orientador: Anderson Almeida Ferreira

UM FRAMEWORK PARA VALIDAÇÃO DE SISTEMAS DE RECOMENDAÇÃO DE JORNAIS ONLINE.

Ouro Preto

Julho de 2018

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas
Bacharelado em Ciência da Computação

UM FRAMEWORK PARA VALIDAÇÃO DE SISTEMAS DE RECOMENDAÇÃO DE JORNAIS ONLINE.

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

GUILHERME APARECIDO GREGORIO

Ouro Preto
Julho de 2018

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

Um framework para validação de sistemas de recomendação de jornais online.

GUILHERME APARECIDO GREGORIO

Monografia defendida e aprovada pela banca examinadora constituída por:

Dr. Anderson Almeida Ferreira – Orientador
Universidade Federal de Ouro Preto

Msc. Bráulio Miranda Veloso – Co-orientador
Universidade Federal de Ouro Preto

Dr. José Maria Ribeiro Neves
Universidade Federal de Ouro Preto

Dr. Álvaro Guarda
Universidade Federal de Ouro Preto

Ouro Preto, Julho de 2018

Resumo

Sistemas de recomendação têm ganhado grande importância nas últimas décadas. Isso se deve ao fato de a tarefa de recomendar itens, tais como produtos, informações ou serviços à um usuário na Internet ser um dos maiores desafios no mundo virtual, devido à grande quantidade de conteúdo disponível. O usuário logo é incapaz de ver todos os itens disponíveis. Tendo isso em vista, um sistema de recomendação pode filtrar itens e fazer sugestões de novos itens aos usuários a partir das características dos itens, dos conteúdos dos mesmos e da experiência geral de todos os usuários. No contexto de jornais online, está à disposição do usuário informações de diversos assuntos. Existe uma grande diversidade nos perfis dos usuários, muitos com interesses bem focados, outros acessam artigos bem variados. Normalmente, um usuário inicia uma sessão de leitura em busca de um artigo de determinado assunto, é comum que as leituras sejam realizadas a partir das recomendações ou links disponíveis, gerando assim um histórico de leituras. Em jornais online, são utilizados sistemas de recomendação para sugerir artigos relevantes para os usuários, almejando que estes permaneçam mais tempo em seus domínios. A recomendação de notícias online tem aspectos específicos, em comparação a outros domínios, incluindo o conteúdo altamente dinâmico do banco de dados, demanda por itens sempre novos e mudanças na preferência do usuário. Novos itens são constantemente criados, contudo os usuários interagem com uma pequena fração das notícias disponíveis, pois essa se torna rapidamente desatualizada e desinteressante. Na literatura, há diversas propostas de sistemas de recomendação para jornais online. Elas se diferem pelas abordagens adotadas, mas todas precisam se adaptar ao alto dinamismo dos dados de jornais online. Cada proposta é normalmente testada em bases de dados reais e validadas com experimentos que simulam a dinâmica de leitura dos usuários dos jornais. Nessa monografia foi criado um *framework* para validar propostas de algoritmos já existentes, avaliando os diferentes algoritmos para demonstrar quais são mais eficazes. Foram coletados dados de um jornais online brasileiro para executar a simulação. Foram implementados alguns algoritmos de recomendação e algumas métricas de avaliação. Os resultados dos algoritmos foram avaliados nessas métricas e comparados com os resultados da recomendação do próprio jornal. O código do *framework* depois de testado foi disponibilizado publicamente, bem como sua documentação.

PALAVRAS-CHAVE: Recomendação de Notícias; Sistemas de Recomendação; Simulação de Avaliação On-line; Recuperação de Informação.

Agradecimentos

Gostaria de agradecer os meus familiares pela ajuda e incentivo para continuar os estudos em especial minha mãe e meu pai, Luiza e Nelson. Que sempre me ajudaram de todas as formas possíveis.

Obrigado também a todos os amigos que dedicaram um tempo para ler meu trabalho. Em especial e com carinho. Obrigado Ana Paula e me desculpa pela falta de tempo.

Obrigado meus orientadores Anderson Almeida Ferreira e Bráulio Miranda Veloso. Obrigado pelo conhecimento que vocês me passaram, pela paciência e incentivo.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Os sistemas de recomendação têm ganhado grande destaque nos últimos anos. Isso em consequência da quantidade crescente de usuários que utilizam serviços na Internet. Serviços tais como vendas de produtos online; disponibilização de informações multimídias; entre outros (Wei et al., 2007; Billsus e Pazzani, 2007). Com uma vasta gama de informação, o usuário pode não conseguir localizar itens que são de seu interesse. Desta forma, os sistemas de recomendação têm sido um grande diferencial para empresas que provem informações (Adomavicius e Tuzhilin, 2005). Um contexto que é bastante aplicado é o sistema de recomendação para jornais online (Resnick et al., 1994), onde os itens a serem recomendados para os usuários são os artigos de notícias. Contudo, isso exige a necessidade de teste e análise dos diversos algoritmos de sistemas de recomendação, para a escolha de um algoritmo que melhor se adeque ao contexto em que será aplicado. Então, o tema e objetivo deste trabalho é criar um *framework* de teste e análise de diversos algoritmos de recomendação aplicado ao contexto de jornais online.

Segundo Veloso (2016) uma descrição formal de um sistema de recomendação clássico é dado por: "Dados n usuários e m itens, a matriz $Y \leftarrow U \times I$ é a matriz de *ratings*. Para cada usuário Uj com $j \in \{1, ..., n\}$ e cada item I_k , com $k \in \{1, ..., m\}$, a entrada $Y_{j,k}$ é preenchida com um valor de *rating*, considerando que o usuário U_j tenha avaliado o item I_k. Caso contrário, é marcado com um símbolo de desconhecido (por exemplo: $NA, ?, -ou, \emptyset$). O problema usual de recomendação é predizer automaticamente com qual valor de *rating* um usuário U_j preencheria os itens ainda não rotulados por ele (completar a coluna j da matriz Y com valores do conjunto de *ratings*) e recomendar ao usuário somente os itens com alto valor predito de *rating*."

A abordagem clássica de sistemas de recomendação é chamada de filtragem colaborativa. Outras abordagens são: baseada em conteúdo e baseada em conhecimento. Se o sistema de recomendação utilizar mais de uma abordagem ele é caracterizado como híbrido (Ricci et al., 2015).

Considerando as leituras de artigos sucessivos feitas por usuários em jornais online, os itens

do sistema de recomendação são formados pelos artigos de notícias. No entanto, o problema de recomendação de notícias é diferente da recomendação clássica. Normalmente, não há *ratings* disponíveis, ou seja, não há uma classificação indicando se o artigo foi interessante para o usuário. Nesse caso, existe apenas a informação de que um usuário leu um determinado artigo, que normalmente é dada por um simples *click* e é considerada como uma classificação positiva, indicando o interesse do usuário por aquele conteúdo. A falta de *ratings* dificulta a aplicação de métodos de filtragem colaborativa tradicionais (Li et al., 2010). Logo a recomendação de novos artigos é normalmente feita baseada no conteúdo de itens similares ao conteúdo dos itens vistos pelo usuário. Essa abordagem é chamada de recomendação baseada em conteúdo (Ricci et al., 2015).

Na literatura sobre recomendação de notícias, há trabalhos que utilizam métodos baseados em conteúdo como o de (Chesnais et al., 1995). Há ainda alguns que, mesmo enfrentando dificuldades que são inerentes a esse processo, utilizam técnicas de filtragem colaborativa (Das et al., 2007) e outros que utilizam uma abordagem híbrida, que misturam características de métodos baseados em conteúdo a métodos de filtragem colaborativa (Agrawal et al., 2009; Miranda et al., 1999). Além dessas três abordagens, há propostas na literatura que utilizam abordagem de conteúdo juntamente com conhecimento (Guimarães et al., 2013; Hsieh et al., 2016; Li et al., 2010; Zhao et al., 2015). Nessas últimas, as recomendações utilizam conhecimento extraído tanto dos dados dos próprios jornais online quanto de fontes externas. Essas quatro abordagens existentes apresentam suas particularidades. Por exemplo, as abordagens de filtragem colaborativa e híbrida precisam de avaliações explícitas, enquanto as outras não. Já as abordagens baseadas em conteúdo, ou seja, aquelas que são compreendidas como híbridas e de conhecimento precisam das informações de conteúdo dos artigos ou dos usuários. Cada particularidade carrega consigo vantagens e desvantagens. Dependendo do contexto onde o algoritmo de recomendação será aplicado, certas particularidades são melhores que outras, fazendo com que os algoritmos de determinada abordagem se destaquem quando aplicado ao contexto correto. Escolher entre diferentes algoritmos de recomendação, que utilizam abordagens diversas, é um problema típico da área de sistemas de recomendação, pois não há um mecanismo para decisão de qual o melhor algoritmo.

1.1 Justificativa

Os sistemas de recomendação têm ganhado muita atenção nas duas últimas décadas. A busca por melhorias nos algoritmos de recomendação é constante, mesmo que essas tragam um ganho pequeno. Segundo Wei et al. (2007), pequenas melhorias em algoritmos de recomendação podem significar grandes aumentos nos lucros das empresas. Sistemas de recomendação de jornais online tem se tornado um foco ainda maior. Existe diversas pesquisas propondo melhorias em sistemas de recomendação de jornais online, devido ao grande problema de re-

comendar corretamente artigos que sejam relevantes aos usuários. Uma boa recomendação é importante, pois motiva o usuário a consumir mais artigos do jornal online e pode definir o ganho ou perda do usuário. Existe então a necessidade de comparar diversos algoritmos de uma só vez para identificar o melhor algoritmo para ser usado no sistema final. Existem diversos trabalhos na literatura propondo novos algoritmos de recomendação, entretanto há poucos trabalhos que avaliam o funcionamento destes algoritmos *online*. Os testes para algoritmos de recomendação normalmente utilizam uma base de dados estática com todos os dados de usuários, de itens e das iterações, tanto para o processo de treino quanto para a avaliação. Esses testes são chamados de testes *offline*, são facilmente reprodutíveis, mas acabam não identificando o melhor modelo para se utilizar na aplicação final. Em um ambiente real de jornal online, as notícias vão surgindo ao longo do tempo, alterando a quantidade de artigos disponíveis aos usuários. Essa dinâmica normalmente não é levada em consideração nos testes *offline*.

Testes *online* são aqueles feitos em tempo real, enquanto os usuários estão acessando o sistema. Entretanto, os algoritmos acabam sendo testados em usuários diferentes. Os usuários tem suas próprias preferências e leem os artigos segundo estas. Eles normalmente não procuram reler as notícias. Desta forma os testes *online* acabam avaliando os algoritmos de forma diferenciada, cada algoritmo é avaliado em acessos de usuários diferentes. Logo, é necessário utilizar uma forma de avaliação que seja mais justa com os algoritmos e que forneça indícios do melhor algoritmo para a aplicação final.

Visto tudo isso, foi planejado um esquema de teste/treino que observa a ordem temporal dos dados, simulando uma aplicação *online*. Esse esquema foi implementado como um *framework* e disponibilizado para que diversas pesquisas que envolvam sistemas de recomendação de jornais online possam utilizá-lo.

1.2 Objetivo

O objetivo geral deste trabalho foi propor e desenvolver um *framework* que simula um ambiente de jornal online para avaliar e testar sistemas de recomendação. Como objetivos específicos foram definidos os seguintes:

- Coletar e analisar uma base de dados de jornais online;
- Analisar métricas de avaliação de sistemas de recomendação;
- Analisar modelos de sistemas de recomendação;
- Modelar um *framework* de validação de sistemas de recomendação de jornais online;
- Avaliar os resultados de algoritmos de recomendação usando o *framework*.

Capítulo 2

Revisão da Literatura

Este capítulo apresenta a fundamentação teórica das áreas que compõem este trabalho. Encontra-se organizado da seguinte forma: A secção 2.1 trata da definição conceitual, da contextualização histórica e da exemplificação de sistemas de recomendação. A secção 2.2 trata de sistemas de recomendação de jornais online. Por fim, na secção 2.3 são apresentados alguns trabalhos relacionados à *framework* de validação de sistemas de recomendação.

2.1 Sistemas de Recomendação

O sistema de recomendação de (Goldberg et al., 1992) é uma das primeiras implementações, utilizando a técnica de filtragem colaborativa. Esse sistema depende das opiniões explícitas dos usuários, o que não é interessante para sistemas de recomendação em grande escala devido ao fato de que cada usuário tem que conhecer todos os outros e estes têm que interagir entre si. Mais tarde, na literatura, muitos sistemas baseados em filtragem colaborativa automatizada surgiram. Ainda com a mesma abordagem, o trabalho de (Resnick et al., 1994) propõem uma solução parcialmente colaborativa para notícias e filmes da base *Usenet*.

(Shardanand e Maes, 1995) e (Hill et al., 1995) criaram sistemas web que geram recomendações para filmes (*Video Recommender*) e músicas (*Ringo*), aplicando outras técnicas aos sistemas de recomendação, incluindo Redes Bayesianas, Clusterização e *Horting*. A técnica Redes Bayesianas mostra-se prática em ambientes onde as preferências do usuário mudam lentamente com o tempo de uso do sistema e são menos eficientes onde as preferências do usuário tendem a mudar frequentemente. A Clusterização é usada para agrupar usuários que têm preferências similares. Uma vez que um usuário faz parte de um grupo, as predições para ele podem ser aferidas com a média do grupo. *Horting* é uma técnica que utiliza um grafo, no qual os nodos são os usuários e as arestas indicam o grau de similaridade entre dois usuários (Aggarwal et al., 1999). Predições são produzidas com o caminhamento no grafo, combinando as opiniões dos usuários mais próximos.

Segundo (Sarwar et al., 2001), o gargalo em sistemas de recomendação baseados em filtra-

gem colaborativa convencional, está na busca por vizinhos em potencial que estão relacionados com o usuário corrente. Algoritmos baseados na filtragem por conteúdo evitam esse gargalo explorando primeiro os relacionamentos entre os itens ao invés dos usuários. As recomendações são computadas com base nos itens que são similares a outros que o usuário já avaliou. Os dados usados nos experimentos desse trabalho foram obtidos da *MovieLens*, que é um sistema web de recomendações de filmes baseado em filtragem colaborativa.

O trabalho de (Koren et al., 2009) relata sobre o prêmio da *Netflix*, uma competição aberta para o melhor algoritmo de filtragem colaborativa com o objetivo de prever classificações dos usuários para os filmes, com base em avaliações de ações anteriores do usuário com o sistema, sem qualquer outra informação sobre os outros usuários ou dos filmes. Um ponto interessante que o autor ressalta é que o algoritmo que realmente foi aplicado para o funcionamento não foi aquele que ganhou o prêmio, mas sim um algoritmo que foi classificado abaixo do ganhador. O algoritmo campeão não foi o melhor quando experimentado no ambiente de produção. Aqui é visto mais um ponto forte para o objetivo deste trabalho, que no caso desse problema os algoritmos deveriam ter sido colocados a prova em uma simulação mais próxima do real.

De acordo com (Cremonesi et al., 2010), uma prática comum de avaliação de sistemas de recomendação está relacionada à sua performance por meio de métricas de erro, como o RMSE (root mean squared error) e o MAE. Porém, em muitos sistemas de recomendação, somente as melhores recomendações são mostradas. Ou seja, o sistema sugere alguns itens específicos para o usuário que provavelmente serão muito atraentes para ele. Os critérios de erro clássico não medem o desempenho do top-N. Metodologias baseadas em métricas de erro não são boas para a acurácia de top-N. Para chegar a essa conclusão (Cremonesi et al., 2010) utilizou o conjunto de dados da *Netflix*, sistema web provedor de filmes e séries, com um conjunto treinado de 100M *rating*(avaliações).

Atualmente, diversas empresas usam sistemas de recomendação, duas destas são: a Pandora, empresa que oferece um serviço de rádio online, e a *Netflix* empresa que oferece um serviço de filmes online. Ambas obtêm vantagem em relação aos seus concorrentes devido ao sistema de recomendação.

A apresentação de definições e classificação de vários autores sobre sistemas de recomendação contribui para esclarecer e justificar o problema em estudo e orientar o desenvolvimento deste trabalho. Nas seções 2.1.1 e 2.1.2, são apresentadas descrições sobre Pandora e*Netflix* e seus respectivos sistemas de recomendação. Estas subseções têm como objetivo exemplificar o sistema de recomendação e por se tratar de empresas bem sucedidas que têm como principal ferramenta de marketing e venda, seus respectivos sistemas de recomendação e também as técnicas usadas por cada modelo.

2.1.1 Pandora

É um serviço de *streaming* de áudio online e um exemplo de sistema baseado em conteúdo. A rádio Pandora originou do projeto *Music Genome Project* (Felfernig e Burke, 2008). Ela expõe música com uma estação de rádio online onde um usuário pode criar estações baseadas em seus interesses musicais. O usuário indica em cada estação uma ou mais músicas ou artistas que ele gosta. Com base nessas preferências, Pandora interpreta músicas semelhantes que o usuário também pode gostar. À medida que o sistema de recomendação é executado, o usuário pode aperfeiçoar a estação, dando uma *thumbs up* ou um *thumbs down* para uma música em particular. Um *thumbs up* significa que o usuário gosta do que ele ouve e quer ouvir mais músicas semelhantes. Um *thumbs down* significa que o usuário nunca quer ouvir esta música em particular novamente e não está interessado em tipos similares. Com cada uma dessas informações sobre o interesse do usuário, a Pandora espera melhorar o serviço de recomendação de novas músicas para os usuários.

2.1.2 Netflix

É uma empresa de que oferece o serviço de *streaming* de vídeos online. A Netflix tem reinventado a televisão na Internet. Com um produto que é um serviço de assinatura e permite aos usuários assistir a qualquer vídeo em sua coleção de filmes e programas de TV em qualquer momento em uma ampla gama de dispositivos conectados à Internet. Até o momento, segundo (Gomez-Uribe e Hunt, 2016), mais de 65 milhões de usuários transmitem mais de 100 milhões de horas de filmes e programas de TV por dia.

Para (Gomez-Uribe e Hunt, 2016) o espaço de televisão na Internet é jovem e a concorrência está madura, de modo que a inovação é crucial. Um grande diferencial do produto é o sistema de recomendação que ajuda os usuários a encontrar vídeos para assistir em cada sessão. O sistema de recomendação não é um algoritmo, mas sim uma coleção de algoritmos diferentes que servem diferentes casos de uso que vêm juntos para criar a experiência completa da *Netflix*. Os seres humanos são péssimos na tarefa de escolher entre muitas opções, rapidamente ficando sobrecarregados e escolhendo nenhum dos itens ou fazendo poucas escolhas. Ao mesmo tempo, um benefício da Internet a TV é que pode levar vídeos de um catálogo mais amplo, atraente para uma ampla gama de interesses e incluindo preferencias de interesse do usuário.

2.2 Sistemas de Recomendação de Jornais online

Os sistemas de recomendação são classificados como: filtragem colaborativa; filtragem baseada em conteúdo e abordagem híbrida. A filtragem colaborativa, reconhecendo semelhanças entre os usuários com base em seus históricos de consumo, fornece uma boa solução de recomendação para os cenários onde os históricos entre usuários se sobrepõem e o conteúdo é quase estático. A filtragem baseada em conteúdo ajuda a identificar novos itens que combi-

nam com um perfil de consumo do usuário existente, mas os itens recomendados são sempre semelhantes aos itens previamente aceitos pelo usuário. As abordagens híbridas foram desenvolvidas combinando duas ou mais técnicas de recomendação. Por exemplo, a incapacidade de recomendar novos itens da filtragem colaborativa pode ser contornada combinando com a filtragem baseada em conteúdo.

No entanto, em muitos cenários na web, o conteúdo sofre mudanças frequentes, como o caso de notícias populares que mudam ao longo do tempo. Além disso, o número de visitantes de um jornal online muitas vezes é inteiramente novo, não há dados históricos desses usuários. Essas questões tornam as abordagens de sistemas de recomendação tradicionais difíceis de serem aplicadas [Li et al. (2010)]. Assim, torna-se indispensável aprender novas técnicas para aplicar onde os interesses e conteúdo do usuário são dinâmicos.

Um usuário de jornal online leva um determinado tempo para ler cada artigo e em seguida selecionar outro artigo ou simplesmente não ler mais nenhum. Esse tempo que o usuário leva para ler artigos é extremamente variável.

O usuário faz acesso a diversos artigos em sequência. A quantidade de artigos acessados define o tamanho da sessão do usuário. O tamanho das sessões pode variar, sendo que um usuário frequente que tem o hábito de ler artigos em um determinado jornal pode ter diversos tamanhos de sessões. Assim como existem usuários que leem de forma bem esporádica.

Além do dinamismo do conteúdo disponível e dos acessos dos usuários, o cenário de recomendação de notícias sofre na maioria dos casos, com a falta de *feedback*, ou seja, não existe o *rating*, que é a avaliação explícita do usuário.

Esse ambiente dinâmico é simulado dentro do *framework*, com as iterações dos usuários do jornal e também o surgimento de novos itens para a recomendação. Ou seja, é feita uma simulação considerando a ordem temporal dos usuários e itens, com o intuito de aproximar o mais próximo possível do cenário real.

2.3 Frameworks de Validação

Este capítulo apresenta trabalhos sobre *frameworks* de validação. Na Secção 2.3.1 é apresentado um trabalho que descreve um *framework* para avaliação de filtragem colaborativa. Na secção 2.3.2 é apresentado outro trabalho mais próximo a este, um *framework* para validação de sistemas de recomendação online. Na secção 2.3.3 é apresentado um *framework* de validação de sistema de recomendação de notícias.

2.3.1 Um Framework para avaliação de Filtragem Colaborativa

O objetivo de (Herlocker et al., 1999) neste trabalho é apresentar um *framework* para a performance de sistemas baseados em filtragem colaborativa e examinar de forma empírica vários algoritmos de recomendação existentes.

Para (Herlocker et al., 1999) sistemas que usam filtragem colaborativa têm uma grande aceitação e têm sido bem sucedidos no mercado. Inicialmente, em site como a *Amazon.com*, uma grande loja de livros online, *CDNow.com*, uma grande loja de CDs online e *MovieFinder.com*, um site de filmes muito visitado na web.

O autor faz a seguinte distinção entre filtragem baseada em conteúdo e filtragem colaborativa, que segundo ele, são diferentes abordagens. A Filtragem colaborativa recomenda a informação com base nas interações dos usuários com o sistema. Essas interações geram dados rotulados pelos próprios usuários, os chamados *ratings*. Os *ratings* são avaliações dos usuários e podem se apresentar de forma explícita ou implícita. De forma explícita os usuários são orientados a avaliar o item, que geralmente pode ser com um único número para cada item ou baseado em uma escala,onde se o número é alto, aquele item é classificado com um forte interesse, caso seja baixo é classificado com pouco interesse. Avaliações implícitas geralmente são derivadas de uma coleta de dados baseada em alguma estratégia, por exemplo muito usado em sistemas de compras, onde as avaliações podem ser baseadas no histórico de compras do usuário. Desta forma, um sistema baseado em uma filtragem colaborativa compara pessoas com os mesmos interesses para fazer a recomendação, ou seja, é feito um cálculo de similaridade entre os usuários.

Sistemas que têm como base filtragem baseada em conteúdo, recomenda a informação certa para a pessoa certa comparando representações que contém informações sobre o interesse da pessoa, essas representações são baseadas nos dados extraídos dos itens, por exemplo um usuário que assiste muitos filmes no contexto de drama, o sistema com uma filtragem baseada em conteúdo, tem acesso a essa informação e logo pode predizer que o usuário queira ver mais filmes deste gênero. A Filtragem baseada em conteúdo seleciona informações que podem ter relevância usando técnicas como *vector-space queries*, *inteligent agents* e visualização de informação.

O trabalho (Herlocker et al., 1999) foca em filtragem colaborativa com avaliações explicita do usuário, usando *ratings* em um intervalo de 1 a 5.

O funcionamento é basicamente o seguinte, coleta-se todos os *ratings* de um usuário, para um sistema baseado em filtragem colaborativa executar as predições. Dado os dados de um usuário avaliados de acordo com sua preferência, dando uma avaliação, por exemplo num dado intervalo 1 a 5, onde o nível de interesse do usuário é crescente de acordo com tal intervalo. Logo, tem-se uma lista de itens, com isso o sistema retorna uma lista de predições com os top-n itens para aquele usuário.

O problema pode ser formulado como uma matriz de usuários por itens, onde cada célula representa um *rating* referente a um item. Dentro desse conceito, o problema é predizer os campos ainda não especificados pelo usuário. Em sistemas de filtragem colaborativa geralmente essa matriz é muito esparsa, uma vez que os usuários avaliam uma pequena parte dos itens em relação ao número total de itens em uma base de dados.

Um modelo de algoritmo de filtragem colaborativa escolhido por (Herlocker et al., 1999) é o método baseado em vizinhos. Nessa abordagem um subconjunto de usuários é selecionado, baseando nas similaridades de seus *ratings*, ou seja, nos dados avaliados pelos usuários. O método baseado nos vizinhos pode ser separado nos seguintes passos:

1. Avaliar o peso de todos os usuários, esse peso relacionado com a similaridades dos *ratings* com o usuário em questão;

2. Selecionar um subconjunto de usuários para usar como um conjunto de predições possíveis para um determinado item;

3. Normalizar os *ratings* e computar a predição combinando os pesos de vizinhos selecionados de acordo com o *rating*.

No *framework* proposto pelo autor para avaliar a qualidade das predições de um algoritmo denota-se a Cobertura e Acurácia.

Na figura 3.1 a seguir é representado uma demonstração da ideia de como funciona um algoritmo baseado na abordagem dos vizinhos.Onde uma exemplificação de 3 usuários diferentes, cada usuário tem seu próprio conjunto de itens. O item A é relevante para os 3 usuários, o item B e D são relevantes para os usuários 2 e 3. Os usuários 2 e 3 tem mais itens em comum, então considerando que o usuário 2 tem interesse pelo item C, é relevante recomendar C para o usuário 3.

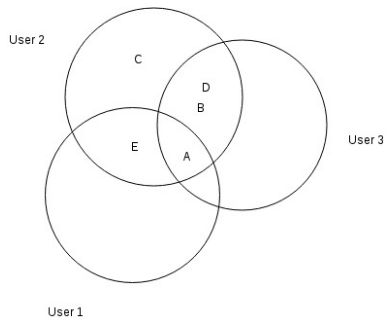

Figura 2.1: Similaridade vizinhos

Nos experimentos (Herlocker et al., 1999) analisa o desempenho comparativo de duas métricas de similaridade diferentes, correlação de *Pearson* e correlação de *Spearman*. Os 50 vizinhos mais similares são selecionados usando cada algoritmo e uma predição é calculada a partir desses vizinhos.

Nos testes é levada em consideração a similaridade dos vizinhos avaliada em pesos. É o primeiro passo nos algoritmos de recomendação baseados em vizinhos, que é o peso de todos os usuários em relação à similaridade com o usuário ativo. Várias medidas de ponderação

de similaridade podem ser utilizadas. Neste trabalho é usada a correlação de *Pearson*, a correlação de *Spearman*, vetor de similaridade baseado no cosseno (Salton e Buckley, 1988), entropia (Press, 2007) e a diferença do quadrado. A base de dados usada foi da *MovieLens*, uma base de dados de filmes, na época com 122176 *ratings* e 1173 usuários, estes últimos tendo em média 20 *ratings*. E alguns usuários foram selecionados de forma aleatória para os testes.

2.3.2 Um framework para validação de sistemas de recomendação online

Em (Hayes e Cunningham, 2002), é apresentado um *framework* de avaliação para sistemas de recomendação com base na ideia da utilidade do sistema, que é uma medida comparativa de como a estrategia de recomendação é executada em relação aos interesses do usuário. O autor reforça a atenção para o de fato que essa avaliação deve medir se as pessoas reais estão dispostas a agir com base nas predições do sistema. Portanto, é necessário avaliar as estratégias de recomendação como parte de aplicações online, simuladas por usuários.

Konstan e Riedl (1999) sugerem que as abordagens existentes para avaliar sistemas de recomendação podem ser divididas em duas categorias:

- Avaliação *off-line*: onde o desempenho de um mecanismo de recomendação é avaliado baseado em conjuntos de dados existentes;
- Avaliação *on-line*: onde o desempenho é avaliado de acordo com os usuários de uma execução do sistema de recomendação.

A avaliação *on-line* é problemática por causa da necessidade de montar um sistema totalmente desenvolvido e criar uma comunidade de usuários. Consequentemente, a avaliação *off-line* é mais favorecida por ser mais fácil de se aplicar.

A arquitetura do *framework* é muito semelhante à de um sistema de recomendação padrão. A grande diferença é que, em vez de um motor de recomendação, existe um componente para os dois motores de recomendação que recebe pedidos de recomendação e chama os métodos relevantes dos respectivos motores. Os dois conjuntos de resultados são então apresentados de acordo com uma política do componente que monta as apresentações ao usuário.

Em relação à execução do *framework*, para gerenciar uma avaliação específica entre duas soluções alternativas, os seguintes elementos devem ser claramente especificados:

1. Recursos disponíveis: uma API que define quais recursos os algoritmos podem acessar para fornecer recomendações;
2. Métodos de recomendação: uma API que define os métodos que um algoritmo deve implementar;

3. Política de apresentação: define como as recomendações fornecidas pelos dois algoritmos serão apresentadas ao usuário;

4. *Feedback* avaliativo: define como as ações dos usuários serão consideradas evidências de preferência de um algoritmo sobre o outro;

5. Validação com métricas: define como analisar o *feedback* avaliativo em ordem para determinar qual algoritmo é o melhor.

2.3.3 PEN recsys: Um Framework Personalizado para Sistemas de Recomendação de Notícia

Esse trabalho descreve a arquitetura de um *framework* de avaliação de sistemas de recomendação com foque em notícias. Propõe fazer a avaliação dos algoritmos de maneira online para *websites* que estão em produção no mercado, ou seja, uma avaliação em produção.

Guimarães et al. (2013) relatam a dificuldade de se recomendar no domínio de notícias, devido a adaptação dos algoritmos de recomendação. E também ao conteúdo dinâmico dos jornais. Ou seja, notícias surgem e desaparecem o tempo todo, notícias antigas podem não ser mais interessantes para o usuário. E ainda existe o problema de que o número de algoritmos especializados para recomendar notícias é pequeno.

Nesse trabalho é relatado que o *framework* segue o paradigma software como serviço e é implementado usando a linguagem Java EE. Segundo o autor ele escala muito bem, pois cada componente é localizado fisicamente em sites diferentes. Ele possui uma interface web de controle, onde pode se executar tarefas simples como ativar ou desabilitar um algoritmo de recomendação.

Esse *framework* possuem seis principais módulos sendo que dois deles mais importantes são: *dispatcher* e *statistics*. *dispatcher* é um módulo responsável por distribuir um sistema de recomendação para um usuário de maneira aleatória, fazendo assim um teste de performance A/B, chamado também de teste multivariante. Isso pode não ser uma boa abordagem e pode enviesar os resultados. *statistics* é um módulo responsável pela avaliação das recomendações e mostra apenas o cálculo do CTR.

O *Framework Pen* contém quatro versões de algoritmos de recomendação padrão, arvores de contexto, filtragem colaborativa simples, uma abordagem baseada em conteúdo, os mais populares e artigos randômicos. Para adicionar um novo sistema de recomendação o usuário deve implementar um método denominado de *getRecommendations*. Uma observação e ponto negativo deste trabalho é que a ferramenta não está disponível para fins de pesquisa. O autor (Guimarães et al., 2013) relata o uso para meios comerciais. Comparando com a metodologia usada para testar e validar os algoritmos de recomendação também pode-se fazer uma crítica em relação ao uso do teste A/B, que aplica para cada usuário um algoritmo diferente.

Capítulo 3

Desenvolvimento

Para concepção de um *framework* de validação de sistemas de recomendação de jornais online, que simule as interações dos usuários com o jornal, foi proposta uma arquitetura geral de funcionamento do *framework*, ilustrada pela Figura 3.1. Basicamente, essa arquitetura representa os módulos e etapas a serem seguidas pelo *framework*, desde o tratamento dos dados até a recomendação feita por um algoritmo de recomendação e posterior avaliação das recomendações. O detalhamento da estrutura e dos dados é descrito nas seções deste capítulo de acordo com cada módulo proposto.

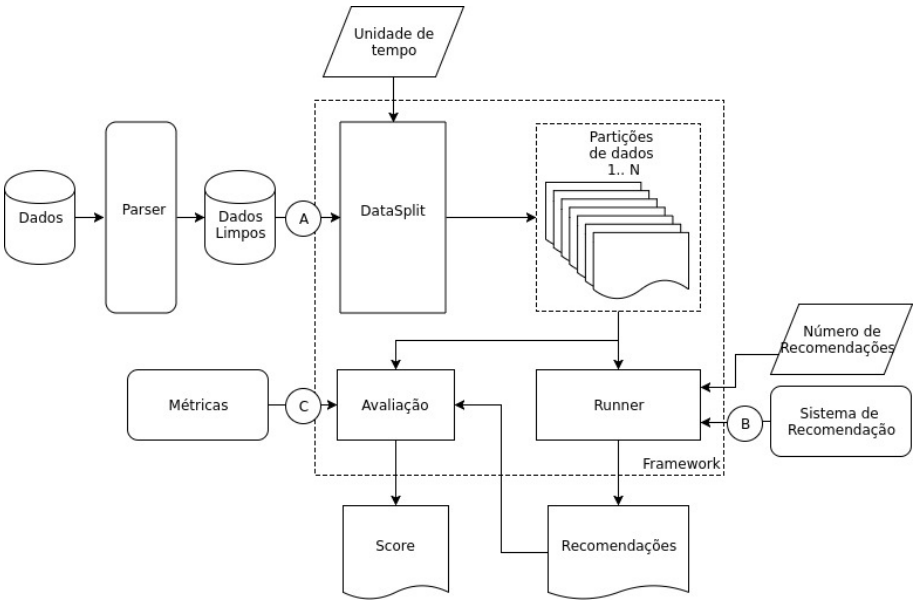

Figura 3.1: Arquitetura do Framework

Na figura 3.1, é mostrada a arquitetura geral do *framework* com os principais módulos: *DataSplit*; *Runner* e *Evaluation*. Também é mostrado o fluxo e entrada de dados entre cada

módulo. *A*, *B* e *C* são protocolos ou padrões definidos: *A* padrão para receber os dados, *B* padrão para receber os algoritmos de recomendação, que é feito via uma interface que o usuário deve implementar e *C* padrão para receber as métricas que também é feito via interface.

3.1 Framework

A criação deste *framework* focou em deixar o mesmo o mais fácil possível de ser usado por outros pesquisadores que desejam testar e avaliar sistemas de recomendação para jornais. Com isso segue uma exemplificação de caso de uso, mostrando como ficou relativamente fácil o uso dessa ferramenta. Seque uma exemplificação do funcionamento do *framework* 1.

Está seção tem como objetivo apresentar e esclarecer o funcionamento do *framework* proposto. Está organizada em subseções, onde cada uma delas descreve o funcionamento e objetivo de um módulo:3.1.1 Módulo Tratamento dos Dados, 3.1.2 Módulo de Recomendação e 3.1.3 Módulo de Avaliação.

Algoritmo 1: Exemplo de funcionamento

```
class Main {
    public static void main(String[] args) {
        // Iniciando o framework e rodando o modulo de dataSplit
        Framework.insertData( arqOfLlogs, arqOfData);
        Framework.runDataSplit(unitTime, iniTime);
        // Adicionando os sistemas de recomendacao criados pelos usuario do
            framework, e rodando-os
        Framework.insertRecSys( aRecSysMakeByUser() );
        Framework.insertRecSys( otherRecSysMakeByUser() );
        ...
        Framework.runRunner(numberRec);
        // Adicionando as metricas implementadas pelos usuario do framework, e
            rodando-os
        Framework.insertMetrics( aMetricMakeByUser() );
        Framework.insertMetrics( anotherMetricMakeByUser() );
        ...
        Framework.runEvaluator();
    }
}
```

O algoritmo 1 é um exemplo de implementação da classe *Main* que o usuário do *framework* deve implementar. É mostrado a inicialização dos dados, a inserção de sistemas de recomendação e métricas.

3.1.1 Módulo Tratamento dos Dados (DataSplit)

Observando a Figura 3.1, repare no literal *A* que está entre dados e o módulo *DataSplit*, ele é o protocolo de entrada de dados para o módulo *DataSplit* que será descrito a seguir.

Inicialmente a base é composta por 2 arquivos no formato csv. Um contendo informações sobre os artigos e outro com informações sobre os usuários, tais como o tempo de acesso e artigos lidos. A estrutura desses arquivos é descrita na Seção 3.2.1. O módulo tratamento dos dados é responsável por criar as partições e diretórios onde os dados irão ser manipulados. Vale ressaltar como descrito na Seção 3.1 que os dados já devem estar limpos e normalizados de acordo com as exigências deste *framework*.

A primeira coisa que é realizada é a leitura desse dois arquivos, isso é feito através do método *Framework.insertData(logs,data)*, Algoritmo 2, onde é passado o arquivo de *logs*, que é referente à iteração dos usuários do jornal e *data* que é informações relevantes dos artigos. Esses métodos são responsáveis pela indexação da base de dados e podem serem omitidos se uma vez executados.

Algoritmo 2: Exemplo de funcionamento DataSplit

```
class Main {
    public static void main(String[] args) {
        long iniTime = 0, unitTime = 604800000; // tempo em milissegundos
        Framework.insertData("logs.csv","data.csv"); // insere dados
        Framework.runDataSplit(unitTime,iniTime);
    }
}
```

Nessa fase acontece a indexação dos artigos contidos em *data* de acordo com a data de publicação de cada. A organização é feita em diretórios */DataBase/articles/0..N/* onde cada pasta é equivalente ao dia de publicação. Como os dados já estão normalizados de acordo com uma data de início para a simulação, todos os artigos que antecedem a tal data serão indexados em */DataBase/articles/0/* e os demais em */DataBase/articles/1..N/*. Nesses diretórios estão indexados os artigos com as respectivas informações relevantes para a execução dos sistemas de recomendação de jornais online.

Em seguida é chamado o método *Framework.runDataSplit(unitTime,iniTime)*, Algoritmo 2, responsável por criar as partições de acordo com o parâmetro *unitTime*, que é quantidade de milissegundos que uma partição possui. Dessa forma é possível simular horas, dias, semanas ou meses em cada partição. Essa abordagem foi adotada para que os testes aconteçam de forma incremental, ou seja, a cada execução de uma partição é exportado um resultado. As partições podem sem encontradas em um diretório que é o mesmo informado pelo usuário do *framework* em *data* concatenado com */DataBase/partition0..N/tasks.txt*. Em cada um desses arquivos existem tarefas que serão passadas para o algoritmo de recomendação. Essas tarefas são compostas de um identificador para o usuário, identificador da sessão, identificador do artigo e identificador para a pasta em que o artigo se encontra.

Este modulo ainda é responsável pela criação dos gabaritos(*templates*) que serão utilizados no módulo de avaliação. Os arquivos referente aos gabaritos podem ser encontrados no diretório *DataBase/templates/template1..N.txt*, onde N é o número de partições da simulação.

Cada linha desses arquivos é estruturado da seguinte forma: id_session(identificador numérico referente a sessão); [id_article1,id_article2,..,id_articleK](lista com os K's identificadores de cada artigo lido pelo usuário, onde K é o tamanho da sessão)

Então existe 1 até N partições, onde N é o número que varia de acordo com o tamanho de dados que o usuário do *framework* deseja simular. Como padrão a primeira partição é usada como inicialização para os algoritmos de recomendação, isso para diminuir o problema de *Cold Start*. Esse problema acontece devido à falta de informação para no caso os algoritmos de recomendação, se não acontecer uma inicialização as predições tendem a ser muito ruins e logo é descartada a avaliação destas partições. Mesmo com essa estratégia não é possível estar livre do *Cold Start*, por exemplo em casos que o sistema recebe acesso de um usuário novo. Ele ainda não leu nenhum artigo e logo não possui nenhum histórico relacionado às suas preferências, então o sistema não pode predizer o que ele gostaria de ler.

3.1.2 Módulo de Recomendação (Runner)

Nesta subseção, é apresentado o módulo *Runner*, que é responsável por executar os algoritmos de recomendação. Este é também um objeto interno, que é construído usando dos padrões de projeto *Factory* e *Singleton*. Ele deve ser executado posterior ao módulo *DataSplit*, garantindo a existência do conjunto de dados e partições para iniciar a simulação. Para inserir um algoritmo de recomendação é usado o método *Framework.insertRecSys(Recommed Obj)*. Como argumento esse método recebe um objeto que implementa uma interface *Recommend* do *framework*, isso define o protocolo *B* como mostrado na Figura 3.1 que liga os sistemas de recomendação com o módulo.

Recommend, Algoritmo 3, possui os seguintes métodos que devem ser implementados pelo usuário:

1. Método *run(String task)* que é o principal método responsável por receber uma tarefa e repassar para o algoritmo de recomendação. O retorno desse método é uma lista com os identificadores de artigos recomendados;

2. Método *init(int numberOfRecommend, File partition, String path)*, este executa uma inicialização do sistema de recomendação para uma simulação mais próxima do real e uma forma de mitigar o problema do *Cold Start*. Como argumento é passado o número de recomendações a primeira partição e o diretório para o conjunto de dados. O retorno desse método é vazio;

3. Método *clean()*, este é pode ser usado para liberar memória interna do sistema de recomendação.

Algoritmo 3: Interface Recommend

```
public interface Recommend {
    /**
     * Receive news items
     * @param task it is a line of partition in the <id_user,id_session,
         timeStamp,id_article>.
     */
    List<String> run(String task);

    /**
     * initial the historic data
     * @param numberOfRecommend it  a number of partitions
     * @param partition it a file of partition
     */
    void init(int numberOfRecommend, File partition, String path);

    /**
     * clean everything
     */
    void clean();

}
```

Desta forma é mantido um protocolo entre as classes internas do *framework* e o usuário tem mais flexibilidade para implementar os seus algoritmos de recomendação.

Dado a existência de um objeto que estende e implementa a classe *Recommend*, para iniciar a simulação e a execução dos algoritmos de recomendação, é usado o método *Framework.runRunner()*, Algoritmo 1, que comunica com as classes internas do módulo *Runner* para executar de forma sequencial cada algoritmo de recomendação. Assim, cada recomendação dos algoritmos é salva em uma linha de um arquivo no seguinte diretório *DataBase/systens/systen1..N/recommends.txt*. Essas recomendações estão no formato: $id\_session; [id\_article1, id\_article2, .., id\_articleN]; time$. Onde o $id\_session$ é o identificador da sessão, $[id\_article1, id\_article2, .., id\_articleN]$ é a lista com os identificadores dos artigos recomendados pelo algoritmo e *time* é o tempo que o algoritmo levou para recomendar. Cada informação dessas serão usadas para a avaliação do algoritmo de recomendação.

3.1.3 Módulo de Avaliação (Evaluate)

É o módulo dedicado às métricas de avaliação. Onde, o módulo de recomendação faz uma predição usando algum algoritmo de recomendação e essa predição é avaliada de acordo com as métricas propostas.

O módulo recebe como parâmetro internamente os arquivos contendo as recomendações. Estes arquivos serão avaliados com base nos arquivos de gabarito(*templates*) descrito na seção

que fala sobre o módulo *DataSplit*, que representa todos os artigos de um usuário lido em uma determinada sessão de leitura, isso ao longo de uma partição, lembrando que a avaliação ocorre a cada partição.

Para a execução desse módulo é necessário a implementação da classe *Metrics*, que possui os seguintes métodos:

1. Método *run(session,recommends)*, o argumento *session* é uma matriz que contém os ids da sessão e ids de artigos lidos na correspondente sessão, *recommends* é uma matriz que contém os respectivos ids mas em relação a recomendação de um determinado algoritmo.

2. Método *getName(name)*, esse método é opcional e é usado para dar nome a métrica, esse nome é usado para nomear as pastas internas da aplicação.

Assim, o próprio usuário do *framework* pode implementar suas métricas de avaliação. Após implementar as métricas deve-se adicionar para a ferramenta fazendo uso do método *Framework.insertMetrics(Metrics obj)*. Onde deve ser passado um objeto *Metrics* como argumento.

Ao executar o método *Framework.runEvaluator()* as métricas são calculada sequencialmente. Cada métrica exporta um resultado que é salvo em um arquivo que se localiza em *DataBase/results/metric0..N/result.csv*. Cada linha desse arquivo é formatada da seguinte forma: id_partition(valor numérico que identifica cada partição); score (valor de cada métrica para avaliação daquela partição). Como saída é gerado um arquivo csv que contém o resultado das métricas e uma comparação estatística em relação aos dados reais.

3.2 Dados de Jornais Online

Os dados utilizados na experimentação deste trabalho foram cedidos por dois jornais online. Vale ressaltar que as informações recebidas dos dois jornais foram em relação aos *logs* do sistema, ou seja, os artigos tiveram que ser coletados do site dos dois jornais. Depois de coletado observa-se que esses dados possuem artigos com datas de publicação de janeiro de 2009 a maio de 2015.

A Figura 3.2 mostra a distribuição das datas de publicação dos artigos acessados no período 02 de fevereiro a 31 de março de 2015. Logo é importante observar que os usuários não leem apenas notícias novas, então um sistema de recomendação que foca em novidades pode não ser um boa abordagem para sistemas de recomendação de jornais online.

Os *logs* de acesso dos usuários são do período de dois meses: fevereiro e março de 2015. E analisando o gráfico mostrado na Figura 3.3 observa-se que a maior concentração de publicações se encontram no período de 02 de fevereiro a 31 de março de 2015, e ainda pode-se observar que a maior parte das publicações ocorre em um padrão de 5 em 5 dias, com isso pode-se dizer que a maior parte das noticias acontece durante a semana seguindo um padrão

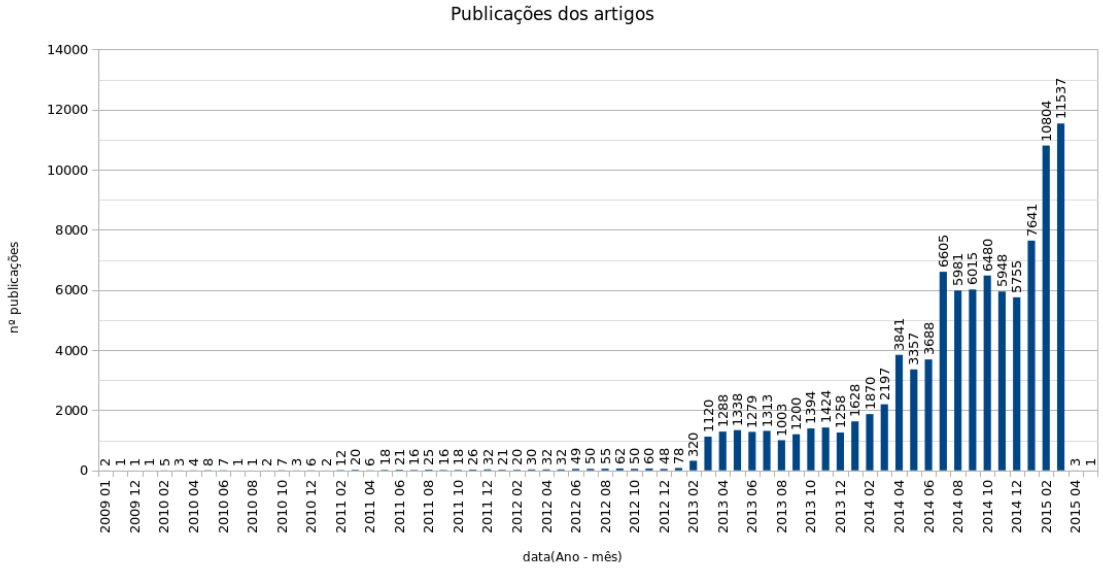

Figura 3.2: Distribuição de publicações

de segunda à sexta. O intuito de analisar esses dados é achar um período em que a simulação será mais parecida com a realidade do jornal.

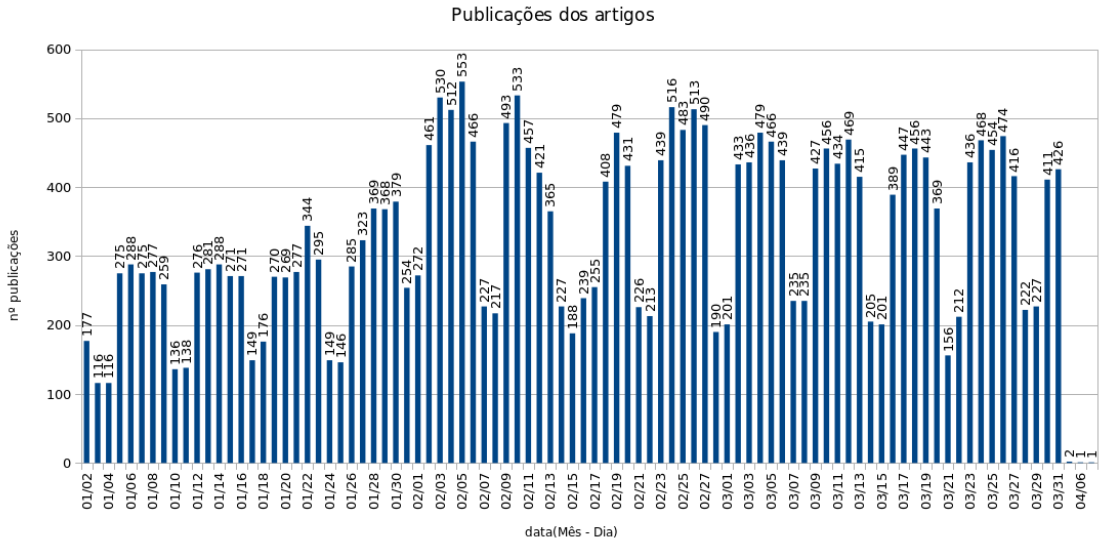

Figura 3.3: Distribuição ao longo de meses

As seguintes subseções apresentam toda a estrutura e informação relevante em relação aos dados, para utilização da construção do *framework*. As subseções são organizadas em 3.2.1

Coleta de dados; 3.2.2 Sessões de leituras e 3.2.3 Tamanho das Sessões.

3.2.1 Coleta de dados

Os dados foram coletados, tratados e salvos em arquivos no formato csv com as seguintes informações referentes aos artigos: artigo_id; URL; Tópico; Título; data de publicação; nome do autor; número de parágrafos; número de monetários; números de datas/hora; número de ordinais; número de porcentagem/fração; número de unidades de medidas; número de coletivos; número de multiplicativos; número de contagem; número de links; número de figuras; número de hashtags; número *stop-words*; número de palavras válidas; as três palavras mais frequentes no título e as trinta palavras mais frequentes no texto. Esses atributos foram selecionados de cada artigo para uso de entrada para os sistemas de recomendação, principalmente aqueles que são baseados em conteúdo.

O arquivo cedido pela empresa que é responsável pela recomendação, é o arquivo de *logs*, basicamente esse arquivo descreve o comportamento dos usuários em relação aos itens recomendados pelo sistema de recomendação. Esse arquivo possui os seguintes campos: *user_id* (um número inteiro que anonimiza o usuário), tamanho da sessão, artigo_id (um número inteiro único do artigo lido), *timestamp* (tempo em segundos do acesso ao artigo), textitrecs (lista dos item_ids recomendados presentes na página do artigo lido) e *click* (informação se a visualização veio de um *click* em um link recomendado).

Além do arquivo de *logs*, três tipos de arquivos foram cedidos pelos jornais, *IDs frequentes com URLs*, *IDs Não frequentes com URls* e *IDs frequentes sem URLs*. Esses arquivos possibilitaram informações para a coleta, eles possuem os identificadores e os links dos artigos. O primeiro arquivo possui os artigos mais frequentes do jornal, ou seja, artigos que são muito acessados, o segundo é referente a artigos que não são frequentes no jornal, artigos que não possuem acesso de usuários. E ainda existe um arquivo que possui os identificadores de artigos frequentes mas não existe o link previamente disponível, para esse caso foi implementado usando uma busca automática de cada artigo com o auxílio de um motor de busca, como o Google por exemplo.

É importante ressaltar que não foi possível coletar todos os artigos. Devido ao fato de os mesmos não existirem mais, isso mostra como o ambiente de jornal online é muito dinâmico e outro fator que ajudou nessa perda de informação é que o arquivo de *logs* é referente a acessos de usuários no período de fevereiro à março de 2015. Com relação aos artigos que não possuem urls e foi implementado uma busca automática, existe uma limitação de requisições diárias isso deixou a busca bem lenta e além de muitos itens não terem sido encontrados.

Para tratar os dados e extrair informações relevantes dos artigos foi necessário construir diversas expressões regulares, isso para tentar cobrir o maior número de tipos de documentos possíveis que apesar de serem do mesmo site não seguem um padrão bem definido.

3.2.2 Formato de entrada do Framework

Este *framework* não é dependente dos dados, porém é definido um formato de entrada dos dados assim como algumas considerações. Uma delas é com relação ao formato de entrada dos arquivos, que devem estar no formato csv. Além disso o arquivo de *logs*, deve conter todos os campos que foram descritos na Seção 3.2.1 e todos os dados devem ser normalizados de acordo com uma data de inicio da simulação. No caso deste trabalho, os dados foram normalizado de acordo com a data 01 de fevereiro de 2015, ou seja, as 0:00 de 01 de fevereiro de 2015 é considerado o tempo 0 milissegundos, toda data anterior a essa possui valor negativo e posterior positivo. Essa normalização deve ser realizada em um processo posterior ao tratamento interno do *framework* e a escolha do ponto inicial para a simulação é levado em critério a dispersão dos dados ao logo do tempo. Essa escolha do período em que a simulação ocorrerá, não impacta o funcionamento do *framework* mas pode influenciar de forma significativa nos resultados. Dessa forma é necessário um estudo dos dados para decidir em qual período é o melhor para trabalhar.

Existem ainda algumas considerações em relação ao arquivo de dados que contém informações referentes aos artigos o *data*, esse arquivo deve estar no formato csv e ordenado pelo tempo de publicação dos artigos. Com relação aos campos é necessário que tenha os seguintes na respectiva ordem: identificar do artigo; título do artigo e data de publicação em milissegundos e também normalizado de acordo com a data inicial para a simulação. Os demais campos são livres e serão usados apenas nos algoritmos de recomendação.

3.2.3 Sessões de Leituras

Cada usuário lê uma certa quantidade de artigos em um certo intervalo de tempo, isso é considerado como sendo as sessões de leituras e é formada pelas leituras consecutivas de artigos diferentes por um mesmo usuário. No caso de leituras repetidas do mesmo artigo em sequência é considerado apenas a primeira. Em cada leitura somente a informação do *item_id* e *timestamp* são mantidas. Uma sessão apresenta além de informações de leituras em ordem cronológica, o identificador do usuário que gerou aquela sessão.

Existem sessões de no mínimo 2 e no máximo 90 artigos lidos e com uma duração total inferior a 90 minutos. Essa duração é a diferença entre o tempo de inicio da sessão até a leitura do último artigo.

As definições de sessões relevantes são passadas ao *framework* através do arquivo *logs*.

3.2.4 Tamanho das Sessões

Ao longo da leitura de um artigo ou ao término, os usuários podem se interessar por outros artigos disponíveis, e assim continuar a sessão, no mesmo assunto ou transitando para um artigo de outro assunto.

Nos dados coletados, existe muitas sessões compostas por mais de um artigo. O tamanho de uma sessão é definido como a quantidade de artigos lidos, e estes tamanhos são bem variados. Pode-se ver no gráfico 3.4 abaixo o tamanho das sessões em relação aos usuários.

Figura 3.4: Tamanho das sessões

Capítulo 4

Experimentos

Para experimentar o *framework* foi executado testes com os algoritmos de recomendação implementados para toda a base. Ao final, cada algoritmo retorna uma lista de recomendações e tal lista foi avaliada com as métricas implementadas. Como comparativo foi usado a recomendação do próprio Jornal. Esses testes não tem como objetivo mostrar uma melhora nas recomendações mais sim um caso de uso e validação do *framework* de jornais online.

4.1 Parâmetros

Para essa simulação foi usado toda a base de dados no período disponibilizado pelo jornal que vai de 1 de fevereiro à 30 de março de 2015. Considerando como dada de início 1 de fevereiro, fazendo assim a partição dos dados, totalizando 8 partições que representam 8 semanas de simulação. Para isso o parâmetro *timeUnit* foi atribuído o valor de 604800000 que é a representação em milissegundos de uma semana e o parâmetro *initTime* atribuído com valor 0.

Os dados de entrada referente aos artigos são ordenados pela data de publicação de cada artigo e essa se encontra em milissegundos e normalizada com a data, 0:00 horas de 1 de fevereiro. Desta forma os valores a baixo dessa data são negativos e valores acima fiquem positivos. O arquivo referente aos artigos possui 117472 artigos e o arquivo referente as iterações dos usuários do jornal possui 7705164 de usuários e 18124490 de sessões.

Os testes apresentados são em relação ao funcionamento e validação. Não foi feito nenhum teste de desempenho. Isso pode ser justificado pelo fato de que os algoritmos de recomendação estão mais ligados a escalabilidade do que o próprio *framework* que irá recebe-los.

4.2 Algoritmos de Recomendação

Para testar e validar o funcionamento e fluxo de dados do *framework*, foram implementados e testados quatro algoritmos de recomendação simples. A ideia aqui é implementar casos de uso

do *framework*. Logo não é a preocupação deste trabalho a implementação de um algoritmo de recomendação robusto e sim garantir o funcionamento do *framework*. Esta seção então descreve o funcionamento de cada algoritmo implementado. A seguir a descrição de Recomendação Randômica (Seção 4.2.1), Recomendação Popular (Seção 4.2.2), Recomendação com TF-IDF (Seção 4.2.3) e Recomendação Usuários Similares (Seção 4.2.4).

Todos esses algoritmos estendem e implementam a classe *Recommed*, para teste e posterior comparação com o sistema de recomendação dos jornais é usado uma lista de recomendação de tamanho 4 e todos os algoritmos utilizam a primeira partição de dados para inicialização.

4.2.1 Recomendação Randômica

Esta é uma implementação ingênua feita apenas para comparação com os outros algoritmos. Este algoritmo estende e implementa a classe *Recommend*.

O primeiro passo é a inicialização das estruturas do algoritmo com o método *init(int numberOfRecommend, File partition, String path)* que foi descrito anteriormente. Implementa também o método *run(String task)* que recebe uma tarefa do *framework*, esta tarefa é então processada e o algoritmo recomenda uma lista de recomendação. Para montar a lista de recomendação é analisado se o artigo já foi lido anteriormente, isso para evitar recomendações de artigos que já foram lidos.

4.2.2 Recomendação dos Populares

Esta implementação recomenda para os usuários os artigos mais acessados. Assim como o algoritmo anterior é usado a primeira partição para iniciar as estruturas. A lista dos artigos mais populares é dada pelo número de acessos em cada artigo.

Na fase de recomendação o algoritmo recebe uma tarefa e processa esta tarefa de acordo com uma lista dos mais populares. Os artigos mais populares que ainda não foram lidos pelo usuário são recomendados.

4.2.3 Recomendação com TF-IDF

Essa implementação é uma abordagem clássica de sistemas de recomendação baseados em conteúdo, que utiliza o cálculo de TF-IDF em cima dos termos que foram extraídos dos artigos. Lembrando brevemente que os dados que o *framework* já são tratados e esses termos são passados como características de cada artigo.

Na fase de inicialização desse algoritmo é usado a primeira partição para o cálculo prévio do TF-IDF, o cálculo é dado pela seguinte fórmula:

$$w_{i,j} = TF_{i,j} * IDF_i$$

onde, $TF_{i,j}$ é a frequência do termo i no documento j, que é dado por:

$$TF_{i,j} = \frac{f_{i,j}}{\sum_j f_{k,j}}$$

IDF_i é dado como a frequência i do termo em relação a N que é o número de documentos, que é dado por:

$$IDF_i = \log(\frac{N}{ni})$$

Uma vez calculado só é necessário calcular novamente quando acontece a postagem de um novo artigo, essa postagem de um novo artigo também é passada como uma tarefa pelo *framework*.

Após o cálculo do TF-IDF é aplicado o algoritmo cosseno para cálculo de similaridade nos pesos $w_{i,j}$, o objetivo é achar o artigo mais semelhante ao artigo que o usuário está lendo naquele momento.

O cálculo da similaridade do cosseno é dado por:

$$cos(d_i, d_j) = \frac{d_i.d_j}{|d_i| * |d_j|}$$

onde, d_i e d_j são vetores que contém os pesos TF-IDF dos termos dos documentos i e j.

Após o cálculo da similaridade é observado se o usuário já leu aquele artigo, senão então ele entra para lista de recomendação. É importante ressaltar que esse cálculo também só é realizado quando o algoritmo de recomendação recebe um novo artigo. É usado uma estrutura de dados para armazenar e evitar o cálculo a todo instante.

É um método simples, porém sua aplicação pode ser um pouco cara. Se mostra lento na fase de inicialização e a estrutura de dados cresce rapidamente demandando mais recurso.

4.2.4 Recomendação Usuários Similares

Esse algoritmo é um exemplo de filtragem colaborativa baseada no usuário, onde é usado o cálculo de similaridade entre os usuários de acordo com suas preferências.

Da mesma forma que nos outros algoritmos é inicializado com a primeira partição e calcula-se a similaridade dos usuários com base nos artigos que eles acessam.

Para cálculo da similaridade é usado a similaridade do cosseno, descrita anteriormente e o vetor usado para o cálculo é a coleção de artigos que o usuário já leu até o momento. Com base nessa similaridade é recomendado para o usuário corrente de maneira aleatória um artigo dos quatro vizinhos mais similares ao usuário corrente.

4.3 Métricas

Esta secção tem como objetivo descrever as métricas implementadas como caso de uso do *framework* e também como uma avaliação dos sistemas de recomendação. Estas métricas estendem e implementam a classe *Metrics*. Está organizado da seguinte forma: Mediada F14.3.1 e CTR4.3.2.

4.3.1 Medida F1

Precisão e Revocação são métricas bem populares de avaliação de informação e podem ser aplicadas a sistemas de recomendação. Proposto por (Cleverdon et al., 1966) elas são muito utilizadas até os dias atuais.

Segundo o autor (Gunawardana e Shani, 2009), essa métrica resulta um bom comportamento para aplicação de jornais online.

A seguir a tabela 4.3.1 de contingência mostrando as relações dos usuários com os artigos. Onde temos casos onde o usuário ativo pode selecionar artigos ainda não selecionados estes podem ser relevantes ou irrelevantes. Ainda os artigos não selecionados podem ser relevantes ou irrelevantes.

Tabela 4.1: Tabela de contingência

	Selecionado	**Não Selecionado**
Relevante	**verdadeiros positivos (VP)**	**falsos positivos (FP)**
irrelevante	**falsos negativos (FN)**	**verdadeiros negativos (VN)**

A precisão é a proporção do número de elementos relevantes recuperados pelo número total de elementos recuperados, como indica a equação:

$$p = \frac{VP}{VP + FP}$$

A revocação é o número de elementos relevantes recuperados pelo número total de elementos relevantes, como mostra a equação:

$$r = \frac{VP}{VP + FN}$$

Para a experimentação desse trabalho foi focado a observação da medida de $F1$, que é a média harmônica entre as duas métricas, apresentada pela equação:

$$F1 = 2.\frac{p.r}{p + r}$$

4.3.2 CTR

Essa métrica é muito usada em recuperação de informação. Ela representa o número de vezes que um item foi clicado. Em jornais online como não existe a classificação do usuário tal métrica se comporta muito bem.

O calculo do CTR é dado por:

$$CTR = \frac{h}{n} * 100$$

onde, h é o número de vez que o usuário clicou em uma recomendação e n é o número de recomendações possíveis.

O CTR é retornado como uma porcentagem e o CTR final é a média para cada partição de dados.

4.4 Resultados

Os resultado apresentados foram obtidos como saída da execução do *framework*, e todos os resultados são comparados com o gabarito que é o comportamento real do usuário, ou seja, informações referente aos artigos lidos pelo usuário. Esta comparação é feita para todos os algoritmos de recomendação incluindo as recomendações feita pelo próprio jornal.

4.4.1 Avaliação CTR

A tabela 4.2 mostra resultados de CTR para os algoritmos de recomendação incluindo as recomendações do Jornal. Onde cada coluna é referente a um sistema de recomendação e cada linha refere-se a uma partição. Essa métrica pode ser medida a proporção de *clicks* em consideração as recomendações.

Tabela 4.2: CTR dos Sistemas de Recomendação

Partição	**Jornal**	**Randômico**	**Populares**	**TF-IDF**	**Usuários Similares**
1	21,2933	0.2933	0,250	1,8630	2,2724
2	15,4689	0,289	0,419	2,5188	2,5702
3	16,3323	0,441	0,330	1,3217	2,0979
4	16,7609	0,456	0,332	1,5843	0,049
5	14,5932	0,461	0,331	1,6400	1,7447
6	16,8778	0,458	0,362	5,7613	0,570
7	17,4918	0,453	0,385	5,7983	4,2016
8	17,4923	0,446	0,330	1,0217	0,579

Na tabela 4.2 é mostrado o valor da métrica CTR para os algoritmos, com um intervalo variando de 0% a 100%, incluindo as recomendações feita pelo algoritmo do jornal, os valores de CTR do sistema de recomendação do jornal é o melhor resultado tendo em média 16,7532% por partição. O algoritmo de usuários similares tem 1,7606%, TF-IDF têm 2,6886%, os algoritmos populares têm e o randômico possui média abaixo de 0,0%. Esse resultado mostra o funcionamento dos algoritmos implementado e passados como parâmetro para o *framework*, onde pode-se avaliar o melhor e pior resultado de acordo com a métrica em questão. Nesse caso pode-se avaliar as recomendações do algoritmo do jornal como as que tiveram maior aproveitamento.

4.4.2 Métrica F1

A Tabela 4.3 mostra os resultados em termos de F1 para cada sistema de recomendação. O intervalo de variação desta métrica é dado de 0 a 1, sendo que quanto maiores, melhores são os resultados. Cada coluna é referente a um sistema de recomendação e cada linha refere-se a uma partição. A F1 é usada como uma medida de acurácia de cada algoritmo que considera tanto a precisão quanto a revocação das recomendações. No caso da precisão mostra o quão preciso é o algoritmo, ou seja, se os artigos recomendados são de fato acessado. A revocação medi a variabilidade das recomendações, por exemplo, se o algoritmo recomenda artigos variados.

Tabela 4.3: F1 dos Sistemas de Recomendação

Partição	**Jornal**	**Randômico**	**Populares**	**TF-IDF**	**Usuários Similares**
1	0.2517	0,0002	0,0002	0,0579	0,0620
2	0,5892	0,0007	0,0041	0,0584	0,0713
3	0,5413	0,0007	0,0033	0,0584	0,0726
4	0,4561	0,0008	0,0033	0,0579	0,0092
5	0,4617	0,0003	0,0033	0,0574	0,0573
6	0,6585	0,0002	0,0003	0,0577	0,0071
7	0,4533	0,0001	0,0038	0,0580	0,0569
8	0,4462	0,0005	0,0033	0,0586	0,0579

Para a tabela 4.3 temos os maiores valores de F1 para o sistema de recomendação do jornal com média de 0,4823 por partição. Como os outros algoritmos tiveram uma taxa de acerto bem pequena e logo a precisão deles ficou bem baixa, em consequência tiveram baixos valores de F1.

Então como esperado o algoritmo do jornal é melhor que os algoritmos aqui testados. Mas isso era esperado, visto que os algoritmos implementados são relativamente simples e abordagens clássicas. Isso mostra também o tamanho do desafio de recomendar notícias principalmente na questão de escalabilidade quando envolve grande quantidade de dados.

Por meio dessas implementações dos algoritmos de recomendação e métricas, foi experimentado um caso de uso do *framework* para validação de sistemas de recomendação de jornais online, simulando assim um ambiente real de jornal online. Todas essas implementações estenderam e implementarão das classes do *framework*. Os algoritmos foram experimentados com dados reais de jornais online e por fim avaliados com resultados que o *framework* retorna. Com essa metodologia é validado o funcionamento da ferramenta tanto para com os dados, algoritmos e métricas, tudo voltado para o problema de recomendação de notícias.

Capítulo 5

Considerações Finais

Sistemas de recomendação são ferramentas úteis para os usuários devido a quantidade massiva de dados disponíveis e muita das vezes é um grande diferencial para empresas que oferecem algum serviço, mas existe uma grande quantidade de dados que o usuário pode escolher.

Neste trabalho, foi proposto um *framework* de avaliação de sistemas de jornais online, a arquitetura do sistema foi projetada e implementada na linguagem java, foram estudadas algumas métricas de avaliação e alguns algoritmos de recomendação com adaptações direcionadas a recomendação de notícias. A primeira versão estável do *framework* encontra documentada e disponibilizada no site *github(`https://github.com/gui666/Framework_Recommender_System`)*.

Assim, o objetivo principal do trabalho que foi criar e validar o uso da ferramenta com casos de uso foi concluído. Como trabalhos futuros. Criar mais casos de uso em busca de falhas no *framework*, melhorar o desempenho de alguns módulos, principalmente o módulo *Evaluate* que se encontra muito lento para executar as tarefas de avaliação. Um trabalho pensado mais que não houve tempo suficiente para fazer, é criar mais um módulo para classificar os algoritmos em um *rank* de acordo com as métricas de avaliação, onde as métricas receberiam pesos a rigor do usuário.

Este trabalho pode ser útil para pesquisadores que desejam simular um ambiente de jornal online e fazer um estudo comparativo de diversos algoritmos de recomendação de jornais online, de forma que é dado como entrada os sistema de recomendação e métricas para avalia-los. E ao final é possível ter dados comparativos entre os sistemas. Com isso o usuário do *framework* pode listar os algoritmos que melhor se comportaram para o ambiente de jornais online devido a simulação que é imposta pela ferramenta.

Referências Bibliográficas

Adomavicius, G. e Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749.

Aggarwal, C. C.; Wolf, J. L.; Wu, K.-L. e Yu, P. S. (1999). Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 201–212. ACM.

Agrawal, M.; Karimzadehgan, M. e Zhai, C. (2009). An online news recommender system for social networks. *Urbana*, 51:61801.

Billsus, D. e Pazzani, M. J. (2007). Adaptive news access. In *The adaptive web*, pp. 550–570. Springer.

Chesnais, P. R.; Mucklo, M. J. e Sheena, J. A. (1995). The fishwrap personalized news system. In *Community Networking, 1995. Integrated Multimedia Services to the Home., Proceedings of the Second International Workshop on*, pp. 275–282. IEEE.

Cleverdon, C. W.; Mills, J. e Keen, M. (1966). Factors determining the performance of indexing systems.

Cremonesi, P.; Koren, Y. e Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pp. 39–46. ACM.

Das, A. S.; Datar, M.; Garg, A. e Rajaram, S. (2007). Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pp. 271–280. ACM.

Felfernig, A. e Burke, R. (2008). Constraint-based recommender systems: technologies and research issues. In *Proceedings of the 10th international conference on Electronic commerce*, p. 3. ACM.

Goldberg, D.; Nichols, D.; Oki, B. M. e Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.

Gomez-Uribe, C. A. e Hunt, N. (2016). The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):13.

Guimarães, S.; Ribeiro, M. T.; Assunção, R. e Meira Jr, W. (2013). A holistic hybrid algorithm for user recommendation on twitter. *Journal of Information and Data Management*, 4(3):341.

Gunawardana, A. e Shani, G. (2009). A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, 10(Dec):2935–2962.

Hayes, C. e Cunningham, P. (2002). An on-line evaluation framework for recommender systems. Technical report, Trinity College Dublin, Department of Computer Science.

Herlocker, J. L.; Konstan, J. A.; Borchers, A. e Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 230–237. ACM.

Hill, W.; Stead, L.; Rosenstein, M. e Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 194–201. ACM Press/Addison-Wesley Publishing Co.

Hsieh, C.-K.; Yang, L.; Wei, H.; Naaman, M. e Estrin, D. (2016). Immersive recommendation: News and event recommendations using personal digital traces. In *Proceedings of the 25th International Conference on World Wide Web*, pp. 51–62. International World Wide Web Conferences Steering Committee.

Konstan, J. A. e Riedl, J. (1999). Research resources for recommender systems. In *CHI'99 Workshop Interacting with Recommender Systems*.

Koren, Y.; Bell, R. e Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8).

Li, L.; Chu, W.; Langford, J. e Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670. ACM.

Miranda, T.; Claypool, M.; Gokhale, A.; Mir, T.; Murnikov, P.; Netes, D. e Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *In Proceedings of ACM SIGIR Workshop on Recommender Systems*. Citeseer.

Press, W. H. (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.

Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P. e Riedl, J. (1994). Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pp. 175–186. ACM.

Ricci, F.; Rokach, L.; Shapira, B. e Kantor, P. B. (2015). *Recommender systems handbook*. Springer.

Salton, G. e Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Sarwar, B.; Karypis, G.; Konstan, J. e Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295. ACM.

Shardanand, U. e Maes, P. (1995). Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 210–217. ACM Press/Addison-Wesley Publishing Co.

Veloso, B. M. (2016). Modelos estocásticos para leitores de jornais online. Master's thesis.

Wei, K.; Huang, J. e Fu, S. (2007). A survey of e-commerce recommender systems. In *Service systems and service management, 2007 international conference on*, pp. 1–5. IEEE.

Zhao, Z.; Cheng, Z.; Hong, L. e Chi, E. H. (2015). Improving user topic interest profiles by behavior factorization. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 1406–1416. International World Wide Web Conferences Steering Committee.