UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Thiago Adriano da Silva

iRec: Um framework para modelos interativos em Sistemas de Recomendação

São João del-Rei

2022

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Thiago Adriano da Silva

iRec: Um framework para modelos interativos em Sistemas de Recomendação

Dissertação apresentada como requisito para obtenção do título de mestre em Ciências no Curso de Mestrado do Programa de Pós Graduação em Ciência da Computação da UFSJ.

Orientador: Leonardo Chaves Dutra da Rocha

Coorientador: Adriano César Machado Pereira

Universidade Federal de São João del-Rei – UFSJ

Mestrado em Ciência da Computação

São João del-Rei

2022

Ficha catalográfica elaborada pela Divisão de Biblioteca (DIBIB)
e Núcleo de Tecnologia da Informação (NTINF) da UFSJ,
com os dados fornecidos pelo(a) autor(a)

S586i

Silva, Thiago.
iRec: Um framework para modelos interativos em Sistemas de Recomendação / Thiago Silva ; orientador Leonardo Rocha; coorientador Adriano Pereira. -- São João del-Rei, 2022.
76 p.

Dissertação (Mestrado - Ciência da Computação) -- Universidade Federal de São João del-Rei, 2022.

1. Sistemas de Recomendação. 2. Aprendizado por Reforço. 3. Aprendizado de Máquina. 4. Multi-Armed Bandit. 5. Comércio Eletrônico. I. Rocha, Leonardo , orient. II. Pereira, Adriano, co-orient. III. Título.

Thiago Adriano da Silva

iRec: Um framework para modelos interativos em Sistemas de Recomendação

Dissertação apresentada como requisito para obtenção do título de mestre em Ciências no Curso de Mestrado do Programa de Pós Graduação em Ciência da Computação da UFSJ.

Trabalho aprovado. São João del-Rei, 1 de setembro de 2022:

Leonardo Chaves Dutra da Rocha
Orientador

Adriano César Machado Pereira
Coorientador

Diego Roberto Colombo Dias
Convidado 1

Leandro Balby Marinho
Convidado 2

São João del-Rei
2022

Agradecimentos

Agradeço, inicialmente, a Deus que me deu forças durante toda minha graduação e agora em meu mestrado, me ajudando a superar todos os obstáculos e medos, não só durante todo esse periodo de estudos e pesquisa, mas em toda minha vida. De forma particular, sou eternamente grato a minha família, pois desde o começo sempre me apoiou sem esitar em momento algum. Especialmente, a minha mãe Elaine, uma mulher batalhadora, que sempre me mostrou o caminho certo a seguir, sempre me incentivando nos estudos, nas horas difíceis, de desânimo e cansaço, e que também me ensinou os principais valores e princípios da vida. Não há palavras que possa descrever o quão grato eu sou por tudo que fez e ainda faz por mim. Sou eternamente grato ao meu pai Adriano, que da mesma forma que minha mãe, sempre me apoiou, nunca me deixou faltar nada, sempre me fazendo sonhar cada vez mais alto. Sinto um imenso orgulho por fazer parte de seu legado. Agradeço à minha irmã Laryssa, que sempre esteve disposta a me ajudar em qualquer situação, obrigado por ser essa pessoa que em diversos momentos, eu sempre soube que poderia contar com sua ajuda. Também de maneira muito especial, agradeço a minha namorada Maria Grazielle, por toda ajuda, incentivo, dedicação, amor e paciência. E a toda restante da minha família, que sempre me deu forças e torceu pelo meu sucesso.

Não menos importantes, agradeço a todos os meus professores que tive ao longo do curso, mas em especial àqueles que fizeram parte deste trabalho. Como meu orientador Professor Leonardo Rocha por ter me recebido, desde minha graduação, como aluno de iniciação científica, onde me tornei muito mais maduro e capaz de resolver diversos problemas graças ao conhecimento adquirido durantes todos esses meses. E por ter me incentivado a cursar o mestrado, e me mostrado que seriamos capazes de criar não só este trabalho, mas todos os outros artigos que desenvolvemos durante esse periodo, dos quais me orgulho muito. Agradeço por sempre estar me direcionando sobre o melhor caminho a ser seguido, de forma clara e única, além também da busca pelo perfeccionismo durante todo o desenvolvimento. Da mesma forma, sou grato ao meu coorientador Adriano Pereira, pela paciência durante diversas conversas e chamadas que tivemos durante todo o desenvolvimento deste trabalho, sendo sempre muito claro e objetivo em todas as situações, sempre disposto à ajudar em diversos problemas que tivemos durante o desenvolvimento deste trabalho. Ao Nícollas Silva, por toda ajuda nas diversas conversas que fizemos com objetivo de realizar alguns experimentos, além das diversas horas gastas em busca da solução para alguns problemas e refatoração de alguns módulos do *iRec*, com todas suas brincadeiras e formas descontraídas de lidar com tudo isso. Tambem preciso agradecer aos amigos que fiz durante esse tempo na faculdade, em particular ao Heitor Werneck, que considero fundamental no desenvolvimento do *iRec*, com todo seu conhecimento da

àrea, foi quem me ajudou, a compreender melhor as melhores práticas de programação, além da sua participação no desenvolvimento do *iRec*, sendo essencial para a conclusão deste trabalho. Ao Carlos Mito, também por toda ajuda recebida durante o desenvolvimento do nosso *framework*. Abro espaço também para agradecer CAPES, que financiou este trabalho desde o início com uma bolsa de mestrado. Enfim, sou eternamente grato a todos vocês, que de alguma forma fizeram parte da minha formação, serei eternamente grato.

"Confie no Senhor de todo o seu coração
e não se apoie em seu próprio entendimento.
Reconheça o Senhor em todos os seus caminhos,
e ele endireitará as suas veredas."
(Bíblia Sagrada, Provérbios 3:5-6)

Resumo

Atualmente, Sistemas de Recomendação (SsR) assumem um papel de destaque na academia e seus avanços refletem diretamente na qualidade de serviços oferecidos pela indústria, como em plataformas de *e-commerce*, de entretenimento e, até mesmo, em redes sociais. Um dos principais avanços recentes é que os SsR são responsáveis por orientar a experiência dos usuários a cada *feedback* fornecido por meio de recomendações interativas, em que a cada interação do usuário o sistema deve recomendar um ou mais itens, receber o *feedback* e atualizar seu conhecimento para a próxima recomendação. A literatura atual tem proposto diversos trabalhos que representam o cenário de recomendação interativa como um modelo de *Multi-Armed Bandits* (MAB). Apesar dos significativos avanços recentes, ainda falta consenso sobre as melhores práticas a serem adotadas para avaliar essas soluções MAB. Diversas variáveis podem afetar o processo de avaliação, mas a maioria dos trabalhos se preocupa apenas com a métrica de precisão de cada método. Assim, este trabalho propõe um *framework* para modelos interativos em SsR, denominado *iRec*. Nossa proposta abrange todo o processo de experimentação, seguindo as principais diretrizes de avaliação de SsR. O *iRec* fornece três módulos para preparar os conjuntos de dados, criar novos agentes de recomendação e simular diferentes cenários interativos. Além disso, também contém diversos algoritmos de última geração, um módulo de ajuste de hiperparâmetros, um vasto conjunto de métricas de avaliação, diferentes formas de visualização dos resultados e um processo de validação estatística.

Palavras-chaves: Sistemas de Recomendação, Aprendizado por Reforço, Multi-Armed Bandits, Aprendizado de Máquina, Comércio Eletrônico.

Abstract

Currently, Recommendation Systems (RSs) have assumed a prominent role in the academy and their advances have directly reflected in the quality of services offered by the industry, such as in e-commerce platforms, entertainment, and even social networks. One of the major recent advances is that the RSs are responsible for guiding the users' experience to each feedback provided through interactive recommendations. At each user interaction, the system must recommend one or more items, receive feedback and update their knowledge for the next recommendation. Current literature has proposed several works representing the interactive recommendation scenario as a Multi-Armed Bandits (MAB) model. However, despite the recent advances, there is still a lack of consensus on the best practices to evaluate such bandit solutions. Several variables might affect the evaluation process, but most of the works have only been concerned with the accuracy of each method. Thus, this work proposes an interactive RS framework named *iRec*. It covers the whole experimentation process by following the main RS guidelines. The *iRec* provides three modules to prepare the dataset, create new recommendation agents, and simulate the interactive scenario. Moreover, it also contains several state-of-the-art algorithms, a hyperparameter tuning module, distinct evaluation metrics, different ways of visualizing the results, and statistical validation.

Keywords: Recommender Systems, Reinforcement Learning, Multi-Armed Bandits, Machine Learning, Electronic Commerce.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Neste capítulo, primeiramente, é feita uma breve introdução sobre Sistemas de Recomendação (SsR) e os diversos métodos existentes. Junto a isso, contextualizamos o cenário onde nosso trabalho está inserido, apresentando os SsR interativos, baseados em aprendizado por reforço, mais especificamente sobre *Multi-Armed Bandits* (MAB), destacando que, apesar dos avanços recentes na área, ainda existe uma falta de consenso sobre as melhores práticas para avaliar essas soluções de MAB. Em seguida, apresentamos o objetivo deste trabalho, que consiste em propor um *framework*, denominado *iRec*, capaz de cobrir essa lacuna na literatura, abrangendo todo o processo experimental seguindo as principais diretrizes dos SsR. De maneira complementar, consolidamos quatro questões de pesquisa relevantes para o desenvolvimento do *iRec*. Por fim, listamos as principais contribuições providas pelo framework, destacando sua relevância.

1.1 Contextualização e Motivação

A grande maioria das aplicações Web atuais possui uma enorme variedade de itens em seus catálogos. Quanto maior o número de itens disponíveis, torna-se mais difícil o processo de reter a atenção do usuário e conectá-lo diretamente ao que ele se interessa (SCHWARTZ; SCHWARTZ, 2004; RICCI, 2015). Ao disponibilizarem centenas de milhares de produtos distintos, involuntariamente, diversos serviços e-commerce dificultam seus usuários a encontrarem itens potencialmente relevantes. Neste contexto, ferramentas capazes de filtrar as informações disponíveis, mostrando apenas o que é de interesse do usuário, tornam-se cada vez mais importantes (BOBADILLA, 2013; MOURAO, 2014). Essas aplicações são denominadas Sistemas de Recomendação e visam estimar itens potencialmente relevantes para os usuários, com base em um conhecimento prévio sobre os seus comportamentos e/ou características relevantes dos itens (ADOMAVICIUS; TUZHILIN, 2005).

Em sua maioria, os SsR analisam os padrões de consumo de cada usuário para fornecerem uma recomendação personalizada que se adapte às preferências destes. Procura-se modelar o perfil de um usuário baseando-se em avaliações feitas anteriormente pelo próprio usuário para outros itens ou por usuários com um perfil similar ao dele. De acordo com a abordagem utilizada, os SsR são classificados em três categorias: (1) filtragem baseada em conteúdo (Content-Based - CB); (2) filtragem colaborativa (Collaborative Filtering - CF); e (3) técnicas híbridas. CB baseia-se na premissa de que usuários possuem interesses em itens com características específicas, correlacionando os usuários com as características dos itens. Por outro lado, CF assume que usuários compartilham interesses em comum,

correlacionado os usuários para recomendar itens em comum (MOURAO, 2014). Por sua vez, os métodos híbridos combinam os métodos de CB e CF, na tentativa de explorar as qualidades de ambas as abordagens. Existem também outros métodos híbridos que visam combinar as abordagens de CB e CF com informações externas aos sistemas, como as informações relacionadas aos usuários, que são provenientes de redes sociais, localização demográfica e outras (ADOMAVICIUS; TUZHILIN, 2005).

Dessa forma, os SsR são utilizados numa ampla variedade de aplicações, tanto em sugestões de filmes, músicas, compras *on-line*, quanto em notícias de interesses, anúncios, dentre vários outros. Empresas como *Amazon*, *Netflix* e *Google* são reconhecidas pelo uso intensivo de SsR com os quais obtêm grande vantagem competitiva. A maior parte desses serviços se destacam por utilizar SsR interativos que permitem que os usuários finais interajam com sistemas de recomendação a todo instante e criem um ciclo de *feedback*, os quais atraem muita atenção devido à sua estratégia de recomendação flexível e à consideração de experiências ideais, ou seja, são capazes de satisfazer o gosto dos usuários, recomendando itens relavantes a curto e longo prazo. Para lidar com a preferência dinâmica do usuário e otimizar as experiências acumulativas, os pesquisadores vêm considerando o conceito de aprendizado por reforço (*Reinforcement Learning* - RL) nestes sistemas.

Nesse sentido, várias soluções baseadas em RL surgiram recentemente na literatura. Dentre elas, grande atenção é dada aos modelos de *Multi-Armed Bandits* (MAB) (ZHAO, 2013; WU, 2016; WANG; HEBERT, 2016; WANG, 2017). Tradicionalmente, o MAB é definido como um modelo de decisão sequencial em que um *agent* deve escolher continuamente uma ação a entre um conjunto de ações $\mathcal{A}$ (também conhecido como *arms*). Sua seleção é guiada pela *action selection policy* de cada modelo e pela sua *value function* relacionada à importância de cada *arm*. Selecionar a ação $a \in mathcal A$ em uma tentativa t resulta em um certo reward $R_t(a_t) \in mathbb R$. O objetivo principal é maximizar os *rewards* retornados após todas as tentativas: $\sum_{t=1}^{T} R_t(a_t)$. Os itens são representados como *arms* a serem selecionados no domínio de recomendação. Enquanto selecionar um *arm* é equivalente a recomendar um item, o *reward* é a resposta do usuário (cliques, aceitação, satisfação etc.), assim, maximizando o *reward* do modelo no longo prazo, o sistema maximiza diretamente o número de vezes que um usuário avaliou um item recomendado pelo modelo.

Semelhante aos cenários tradicionais, esses sistemas de decisão sequencial têm duas opções possíveis: (1) *exploiting* o *arm* que parece ser a melhor opção até agora; ou (2) *exploring* um *arm* ainda não testado (ou não tentado o suficiente) (ZHAO, 2013; SANZ-CRUZADO, 2019). A qualidade de qualquer solução MAB está relacionada à forma como ela trata esse dilema *exploration* e *exploitation* em sua política de seleção de ação (*action selection policy*). As três soluções MAB clássicas são ϵ-*Greedy* (AUER, 2002), *Upper Confidence Bounds* (UCB) (AUER, 2002; AUER, 2002) e *Thompson Sampling*

(TS) (CHAPELLE; LI, 2011). Em geral, ϵ-*Greedy* realiza escolhas aleatórias de acordo com um parâmetro ϵ. Por sua vez, o UCB cria um intervalo de confiança para cada *arm* e, primeiro, seleciona aqueles com maiores incertezas. Por outro lado, TS desenha uma distribuição desconhecida para cada *arm* e a usa para prever o *reward* esperado associado. Esses algoritmos são a base das estratégias mais recentes (ABEILLE; LAZARIC, 2017; WANG, 2018; SHAMS, 2021).

Apesar dos avanços recentes, esse novo cenário online apresenta um grande problema na avaliação dos SsR atuais. Como mencionado, esses novos algoritmos são baseados em RL e geralmente exigem um ajuste de vários parâmetros que podem afetar diretamente sua eficácia. Em um cenário interativo, devemos definir pelo menos: (1) o número de tentativas a serem realizadas; (2) o número de itens a serem recomendados em cada tentativa; (3) se o usuário pode receber as mesmas recomendações nas próximas tentativas; (4) a forma como cada usuário será escolhido para receber alguma recomendação; (5) os critérios para atualizar o conhecimento do sistema; (6) a métrica de *reward/regret* a ser otimizada; e muitos outros. No entanto, infelizmente, há uma completa falta de consenso sobre as melhores práticas de avaliação de um SsR interativo. Consequentemente, surgiram duas grandes preocupações: avaliação não reproduzível e comparações injustas. Muitos trabalhos utilizam configurações experimentais que beneficiam alguns métodos, seja através de uma base de dados ou uma metodologia de avaliação enviesada, além da comparação de modelos de recomendação sem tunagem de parâmetros. (SUN, 2020). Além disso, conforme apontamos em uma recente revisão sistemática da literatura sobre MAB em recomendação (SILVA, 2022), a maioria dos trabalhos simplesmente abstraem as noções de *reward* ou *regret* da teoria RL para avaliar seus algoritmos. Nesse sentido, eles estão preocupados apenas com a eficácia da previsão da recomendação - um conceito considerado insuficiente na literatura de recomendação atual.

1.2 Objetivos

A fim de atenuar essa lacuna presente no processo de avaliação das soluções de MAB, propomos neste trabalho um novo *framework* para avaliar SsR interativos, denominado *iRec*, o qual tem como objetivo fornecer uma comparação imparcial e justa entre distintos modelos de SsR com diversas metodologias de avaliação amplamente testadas e utilizadas na literatura. Para isso, nosso *framework* possui módulos específicos para o tratamento dos dados, para avaliação de um experimento completo, com tunagem de parâmetros e metodologias e métricas ideiais para o cenário de recomendação. O *iRec* consiste em uma estrutura completa composta por três principais componentes. O primeiro componente é responsável por configurar o ambiente (**environment**) e tem como objetivo realizar o carregamento e o pré-processamento dos conjuntos de dados de recomendação. O segundo é responsável por implementar os agentes (*agents*) requeridos por um cenário

interativo. No contexto deste trabalho, o *iRec* implementa **recommendation agents** (SsR) para treinarem, se necessário, e ,então , realizarem todas as previsões (recomendações) necessárias. O terceiro componente é responsável por criar a política de avaliação (**evaluation policy**) dos algoritmos usuais de aprendizado por reforço para definir como os agentes irão interagir com o ambiente da tarefa.

1.3 Desenvolvimento da Pesquisa

A pesquisa descrita foi conduzida a partir da investigação de quatro questões de pesquisa principais que foram respondidas no decorrer do trabalho. São elas:

QP1: Como os modelos MAB vêm sendo avaliados no cenário de sistemas de recomendação interativa?

QP2: Existe na literatura algum *framework* capaz de englobar todas etapas (desde a coleta dos dados até a avaliação dos resultados) para experimentação off-line para o cenário de sistemas de recomendação interativos?

QP3: É possível prover um ambiente de execução reprodutível no cenário de SsR interativos capaz de amenizar a lacuna presente no processo de avaliação das soluções de MAB?

QP4: O quão abrangente um ambiente reprodutível pode ser a fim de suprir as necessidades dos diversos cenários em SsR interativos?

De maneira geral, para responder à primeira pergunta, utilizamos dados recentes, coletados por uma revisão sistemática da literatura (RSL) (SILVA, 2022) sobre MAB no campo de recomendação. Durante toda essa revisão, foram levantadas diversas informações que nos permitem fazer inúmeras análises sobre como os trabalhos atuais avaliam seus modelos de MAB. Dessa forma, através dessa RSL, podemos observar que a grande maioria dos trabalhos optam por estratégias que dificultam o processo de reprodutibilidade. Percebe-se que não há um consenso quanto às etapas de pré-processamento que devem ser aplicadas. Além disso, as avaliações utilizadas para medir o quão bom é um recomendador ou estão focadas apenas no processo de aprendizagem, o que está muito relacionado à teoria de aprendizagem por reforço, ou estão focadas apenas no resultado final das recomendaçãoes (acurácia das recomendações aos usuários). Sendo que conceitos como novidade, diversidade e cobertura também são de extrema importância para medir o desempenho de um recomendador. A resposta completa a essa pergunta é apresentada no Capítulo 3.

Intensionando responder a segunda pergunta deste trabalho, exploramos e analisamos, profundamente, diversos *frameworks* e bibliotecas de avaliação já existentes na literatura. Dessa forma, encontramos alguns trabalhos que tentam lidar com a avaliação

dos modelos MAB no cenário de recomendação interativa: *BEARS* (BARRAZA-URBINA, 2018) *MABWise* (STRONG, 2021; STRONG, 2019) e *OBP* (SAITO, 2020). Para cada uma dessas propostas, apresentamos detalhadamente a arquitetura de cada uma delas, bem como seus principais componentes. Em seguida, realizamos uma comparação dessas três propostas encontradas na literatura com o nosso *Framework iRec*. Através desse estudo, verificamos que nenhum dos trabalhos existentes até o momento, engloba integralmente o processo experimental de avaliação de um sistema de recomendação desde a entrada dos dados até a metodologia de avaliação. Apenas o *iRec* é capaz de cobrir todo esse processo. A resposta completa a essa pergunta é apresentada no Capítulo 3.

Conforme mencionamos na Seção 1.2, visando responder à terceira pergunta, propomos o *iRec* apresentando toda a sua estrutura, bem como bases de dados, algoritmos, métricas e políticas de avaliação de última geração integradas a ele, além de demonstrar como o *iRec* pode ser facilmente configurado para criação de diversos experimentos no cenário de recomendação interativa. Por fim, para responder à quarta pergunta, apresentamos a utilização de nosso *frameworks*, que foi instanciado em diversos cenários de recomendação interativa: músicas, filmes, livros e pontos de interesse (POI), demonstrando toda a amplitude que um ambiente como *iRec* pode trazer (ANONYMOUS, anonymous).

1.4 Principais Contribuições

A principal contribuição deste trabalho é ser uma referência no processo de avaliações reprodutíveis de SsR interativos por meio de nossa proposta denominada *iRec*. O *iRec* pode ajudar a comunidade científica de recomendação, orientando novos pesquisadores no campo da aprendizagem por reforço e também oferecendo a possibilidade de pesquisadores especialistas validarem e contrastarem seus algoritmos com várias linhas de base. De forma pontual, a seguir apresentamos as principais contribuições providas pelo *iRec*, destacando sua relevância:

- Compatibilidade com vários conjuntos de dados públicos de filmes, músicas, livros, produtos e roupas;
- Uma estrutura modular, reutilizável e extensível, onde qualquer pesquisador pode adicionar novos conjuntos de dados, modelos, políticas e métricas de avaliação de forma intuitiva;
- Todo um *pipeline* de execução paralelizado, registrando todas as ações (i.e. log), testes estatísticos e diversas formas de visualização dos resultados;
- Abordagens distintas de filtragem e pré-processamento para manipular o conjunto de dados inserido;

- Um total de 17 modelos de recomendação adaptados à tarefa de recomendações interativas;
- Um módulo de ajuste de hiperparâmetros para permitir a adaptação de modelos SsR interativos complexos e fornecer cenários de avaliação distintos;
- Diversas métricas de avaliação com objetivos distintos, como *accuracy*, *novelty*, *diversity* e *coverage*.

Ademais, esta dissertação de mestrado resultou em 5 artigos:

1. Um artigo no WebMedia 2020 (A3) (SILVA, 2020);
2. Um artigo no WebMedia 2021 (A3) (SILVA, 2021);
3. Um artigo no periódico Expert Systems With Applications 2021 (A1) (SILVA, 2022);
4. Um artigo no SIGIR 2022 (A1) (ANONYMOUS, anonymous);
5. Uma submissão de artigo no WebMedia 2022 (A3).

1.5 Organização do Documento

Este trabalho é organizado da seguinte forma. O Capítulo 2 apresenta um referencial teórico sobre o problema de recomendação, no qual o definimos formalmente. Ainda neste, endereçamos o problema de *Multi-Armed Bandit* e descrevemos os tradicionais algoritmos MAB: ϵ-*Greedy* (AUER, 2002), *Upper Confidence Bounds* (UCB) (AUER, 2002) e *Thompson Sampling* (TS) (CHAPELLE; LI, 2011). Para cada um dos algoritmos, sumarizamos suas principais características e apresentamos, em detalhes, como cada abordagem lida com o dilema de *exploration/exploitation*. No Capítulo 3, respondemos a primeira pergunta desta pesquisa, na qual mostramos os problemas relacionados à avaliação dos modelos MAB. Para isso, descrevemos todo o processo de uma RSL, utilizada como base para avaliarmos os critérios de avaliação empírica de modelos de MAB. Após todo o processo de análise e estudo, descrevemos todas limitações encontradas nos trabalhos que fizeram uso de modelos MAB no cenário de recomendação interativa. Levantamos diversos problemas relacionados à reproducitilidade dos resultados, à falta de padronização quanto às estratégias de pré-processamento, além de outros problemas relacionados às métricas e políticas de avaliação utilizadas para medir a qualidade das recomendações.

O Capítulo 4 tem como objetivo responder à segunda pergunta deste trabalho. Dessa forma, apresentamos alguns *frameworks*/bibliotecas que encontramos que tentam lidar com a avaliação de algoritmos MAB. Para cada um dos trabalhos encontrados, detalhamos toda sua estrutura, descrevendo seus principais componentes e características.

Ao fim deste capítulo, fizemos uma comparação com o nosso *framework iRec*, a partir da qual podemos ver que apenas *iRec* engloba integralmente o processo experimental de avaliação de um sistema de recomendação desde a entrada dos dados até a metodologia de avaliação. No Capítulo 5, apresentamos, finalmente, toda a estrutura e utilização do *framework iRec*, como resposta à pergunda de número três. Nele, especificamos todas estratégias presentes no módulo de preparação de dados, módulo para tunagem de parâmetros, modelos de recomendação disponíveis, além de diversas políticas e métricas de avaliação voltadas diretamente para o cenário de recomendação. Também mostramos como utilizar o *iRec*, seja por linhas de comando, ou importanto e utilizando os módulos disponíveis em nossa biblioteca. No Capítulo 6, respondemos a pergunta número quatro, demonstrando a utilização de nosso
framework, apresentando todo o processo de configuração em cenários distintos de recomendação: filmes, músicas, livros, POI (points-of-interest). Por fim, o Capítulo 7 apresenta as conclusões deste trabalho, sumarizando todas as análises e resultados apresentados em cada capítulo anterior. Neste, também retratamos alguns possíveis trabalhos futuros a serem realizados.

2 Referencial Teórico

Neste capítulo, apresentamos uma definição formal do problema de recomendação, enfatizando a tarefa de predizer os itens de interesse do usuário. Em seguida, discutimos sobre o problema de *Multi-Armed Bandits* no cenário de recomendação e apresentamos as três soluções clássicas de modelos MAB: $\epsilon - Greedy$, *Thompson Sampling* e *Upper Confidence Bounds*. Por fim, descrevemos como os pesquisadores avaliam o desempenho dos modelos MAB no cenário de recomendação interativa.

2.1 Problema de recomendação

Atualmente, SsR têm assumido um papel de destaque na academia e seus avanços têm refletido diretamente na qualidade de serviços oferecidos pela indústria, como em plataformas de *e-commerce*, de entretenimento, e até mesmo em redes sociais (AMATRIAIN; BASILICO, 2015; AMATRIAIN; BASILICO, 2016). Em geral, o principal objetivo da recomendação é definido como encontrar/predizer entre um número potencialmente grande de itens aqueles que melhor se adéquam aos interesses individuais de cada usuário (MOURAO, 2014). Mais formalmente, o problema da recomendação pode ser definido da seguinte forma: consideremos $U = \{u_1, u_2, ..., u_m\}$ o conjunto de todos os usuários e $I = \{i_1, i_2, ..., i_n\}$ o conjunto de todos os possíveis itens a serem recomendados pelo sistema. O objetivo da tarefa de recomendação consiste em encontrar um subconjunto de itens $Ru \subset I$, de tamanho k *(i.e.,* $|Ru| = k$*)*, que maximizam a função de utilidade *f(u, i)* para cada usuário $u \in U$ e $i \in Ru$ (ADOMAVICIUS; TUZHILIN, 2005), como mostra a Equação 2.1.

$$R_u = \frac{arg\ max\ f(u,i)}{i \in I} \tag{2.1}$$

Os *ratings* (avaliações) obtidos pelos usuários são utilizados para recomendar novos itens no futuro, para isso torna-se necessário armazená-los e utilizá-los para aprender/-conhecer mais sobre o usuário e, assim, aumentar a probabilidade de recomendar itens que possivelmente o usuário irá consumir. Dessa forma, o sistema de recomendação pode ser interpretado como uma função que mapeia pares de usuários/itens para números reais que indicam a relevância do item para o usuário. Quanto maior o valor predito de relevância, maior a probabilidade de que o usuário irá gostar do item. Assim, o problema de recomendação é encontrar os itens e predizer os *ratings* que mais se aproximarão do gosto do usuário.

Nesse cenário, um dos principais avanços recentes é que os SsR não são mais algoritmos *off-line*, que fazem previsões em *batch* de acordo com os requisitos do negócio. Atu-

almente, esses algoritmos são responsáveis por orientar a experiência dos usuários a cada *feedback* fornecido por meio de recomendações on-line, baseados na teoria de aprendizado por reforço (ZHAO, 2013; SUTTON; BARTO, 2018). Na maioria dos cenários, tais SsR se comportam como um modelo de decisão sequencial (WU, 2019; ZHOU, 2020). A cada interação do usuário, o sistema deve recomendar um ou mais itens, receber o *feedback*, e atualizar seu conhecimento para a próxima recomendação (WU, 2018; ANONYMOUS, anonymous). A ideia é aprender a cada interação para aumentar o conhecimento do sistema sobre usuários e itens e, assim, maximizar a satisfação do usuário no longo prazo. Por essa razão, a literatura atual tem proposto diversos trabalhos que representam o cenário de recomendação interativa como um modelo de Multi-Armed Bandits (MAB) (WANG, 2018; SANZ-CRUZADO, 2019; ZOU, 2020; SHAMS, 2021), o qual será detalhado na seção seguinte.

2.2 Multi-Armed Bandit

O termo *Multi-Armed Bandits* (MAB) vem de um experimento hipotético no contexto de jogos caça-níqueis, em que o jogador se depara com o dilema entre *exploration*, puxando os braços (*arms*) da máquinas menos exploradas de maneira otimista, em busca da máquina que irá lhe fornecer a melhor recompensa, e *exploitation*, puxando o braço que é conhecido como o melhor até o instante atual, em termos de render a recompensa máxima (dilema *exp-exp*). Inicialmente, o modelo de MAB era tido apenas como um problema estatístico (AUER, 2002), porém, recentemente, esse modelo vem sendo amplamente estudado e acabou se tornando fundamental em diversas áreas de inteligência artificial, como aprendizado por reforço (AUER, 2002; SUTTON; BARTO, 2018) e programação evolutiva (HOLLAND, 1992). O problema de *MAB* é um problema clássico que modela um agente que deseja maximizar sua recompensa total pela qual deseja simultaneamente adquirir novos conhecimentos (*exploration*) e otimizar suas decisões com base nos conhecimentos existentes (*exploitation*). Esse dilema em descobrir *exploration* versus utilizar *exploitation* pode ser descrito como a busca da melhor relação entre gastar recursos para descobrir um ambiente ou tirar proveito dele por meio de ações possivelmente mais lucrativas (AUER, 2002).

No contexto de SRs, os esforços atuais modelam o problema de aprendizado on-line como um problema de *Multi-Armed Bandits* para lidar com a tarefa de recomendação no contexto de recomendação online (LI, 2016). Formalmente, MAB é um modelo de decisão sequencial representado por uma 3-tupla $\langle \mathcal{A}, \mathcal{R}, \mathcal{Q} \rangle$, no qual um agente deve escolher continuamente uma ação $a \in \mathcal{A}$ durante as $T \in \mathbb{N}$ tentativas, a fim de maximizar o *reward* cumulativo: $\sum_{t=1}^{T} r_t$. Aqui, $r_t = \mathcal{R}_t(a_t)$ é o *reward* alcançado quando uma ação a é realizada. Um algoritmo MAB executa uma ação a em cada tentativa t de acordo com uma política de seleção de cada ação π. Essa política é orientada a seguir uma distribuição de probabilidade, geralmente chamada de função de predição $\mathcal{Q}$, sobre cada ação a possível. A

função $\mathcal{Q}$ é a principal responsável por selecionar ou não uma ação a porque mede o *reward* esperado: $\mathcal{Q}_t(a) = \mathbb{E}[r_t|a]$. A Figura 1 ilustra essa definição, mostrando o agente e o ambiente interagindo continuamente em uma sequência de intervalos de tempo discretos t. Em cada etapa de tempo t, o agente apresenta um *arm* a_t e recebe um *reward* r_t. O histórico de ações e *rewards* orienta a seleção do agente para o tempo $t+1$ e as demais tentativas.

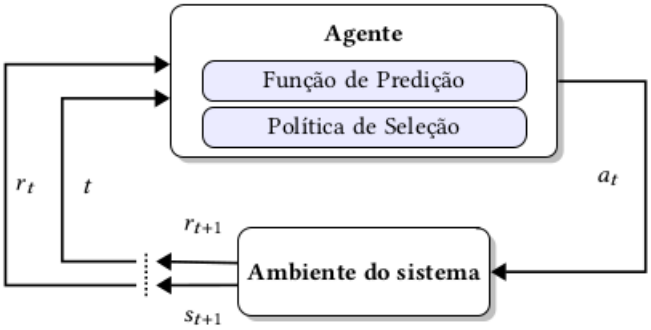

Figura 1 – Esquema tradicional de Multi-Armed Bandit

Dessa maneira, também é comum redefinirmos o objetivo principal do MAB como minimizar o *regret* (i.e., erro esperado) de cada ação a escolhida, conforme mostra a Equação 2.2. Conhecendo o *arm* a_t^* com o maior *reward* esperado no tempo t e o *reward* r_t^* ótimo, o *regret* pode ser definido como a diferença entre r_t^* e o *reward* r_t alcançado pelo agente. Tais definições são equivalentes e são usadas de acordo com o objetivo de cada tarefa.

$$\text{Maximizar} \sum_{t=1}^{T} r_t \quad \equiv \quad \text{Minimizar} \sum_{t=1}^{T} (r_t^* - r_t) \tag{2.2}$$

Se o sistema tiver conhecimento suficiente sobre o domínio, a melhor opção seria selecionar a ação que proporciona a máxima recompensa possível a todo tempo (ZHAO, 2013; SANZ-CRUZADO, 2019). Contudo, esse conhecimento é incerto e muitas vezes desconhecido. Por essa razão, o modelo MAB tem sempre que decidir entre duas opções. Uma primeira opção, mais conservadora, é selecionar os *arms* com as maiores recompensas do passado – uma abordagem de *exploitation*. Em contrapartida, outra opção é investir em diferentes *arms* a fim de obter mais informações sobre o domínio e tomar decisões futuras ainda melhores – uma abordagem de *exploration*. Tais opções caracterizam o dilema de *exploitation-exploration* (i.e. exp-exp) e exigem que o modelo seja capaz de explorar o máximo conhecimento disponível enquanto também explora o espaço de solução para adquirir ainda mais conhecimento sobre o domínio (ZHAO, 2013).

Assim sendo, diferentes métodos MAB foram propostos na literatura para considerar o dilema *exp-exp* (SUTTON; BARTO, 2018). Eles geralmente incluem os métodos tradicionais de MAB, como ϵ-*Greedy* (AUER, 2002), *Upper Confidence Bounds* (AUER, 2002) e *Thompson Sampling* (CHAPELLE; LI, 2011). Em geral, o ϵ-*Greedy* modela o problema com base em um parâmetro de diversificação ϵ para executar a fase de *exploration*.

Por sua vez, o UCB modela um intervalo de confiança para cada *arm* e seleciona primeiro o que tiver a maior incerteza. Por outro lado, as abordagens de TS traçam uma distribuição desconhecida para cada *arm* no intuito de prever a recompensa esperada associada. A seguir, falamos em detalhes sobre cada uma dessas estratégias de *Multi-Armed Bandit*.

2.2.1 *Epsilon-Greedy*

No algoritmo *epsilon-greedy*, o melhor item é considerado como sendo a opção com a maior probabilidade de recompensa (SUTTON, 1996), como o próprio nome sugere, é o mais "guloso"(*greedy*) dos três algoritmos MAB. A execução do algoritmo consiste em encontrar uma compensação entre dois objetivos concorrentes: *exploration/exploitation* por meio do parâmetro ϵ. Inicialmente, o algoritmo se depara com dois caminhos: (1) uma fração proporcional a ϵ, na qual um item deverá ser escolhido de maneira aleatória dentre todos os itens; e (2) por outro lado, a fase de exploração ocorre proporcionalmente a uma fração dada por 1 - ϵ. Por exemplo, quando tem-se $\epsilon = 0.9$, significa que em 90% das escolhas o algoritmo irá explorar a melhor opção conhecida enquanto nos 10% restantes, o algoritmo selecionará uma opção aleatória entre todas as disponíveis. Dessa forma, em um contexto de recomendação de filmes, em 90% das escolhas o algoritmo poderá recomendar filmes mais relevantes de acordo com a estratégia π definida. Por exemplo, enquanto nos outros 10% ele escolherá um filme completamente aleatório. O raciocínio por trás dessa abordagem é que, durante uma parte do tempo, o algoritmo explora aleatoriamente as ações disponíveis, mas ainda explora a melhor ação na maioria das vezes. De forma geral, o algoritmo *epsilon-greedy* possui boa acurácia em uma variedade de aplicações. No entanto, como em uma das suas escolhas o algoritmo escolhe um item de maneira aleatória, esse algoritmo é vulnerável em algumas situações, pois, por escolher igualmente entre as opções disponíveis, não existe preferência explícita por nenhum item e as opções "pior possível"e "próximo do ótimo"são igualmente prospectadas. Dessa forma, outros algoritmos MAB foram propostos, os quais focam em garantias de convergência ótimas.

2.2.2 *Upper Confidence Bounds*

A exploração aleatória nos dá a oportunidade de experimentar opções sobre as quais não conhecemos muito. Todavia, devido à aleatoriedade, é possível que acabemos explorando uma ação ruim que confirmamos no passado. Para evitar essa exploração ineficiente, uma abordagem é diminuir o parâmetro ϵ no tempo e a outra é ser otimista sobre opções com alta incerteza e, assim, preferir ações para as quais ainda não tivemos uma estimativa de valor confiável, ou, em outras palavras, favorecemos a exploração de ações com um forte potencial para ter um valor ótimo. O algoritmo UCB mede esse potencial por um limite superior de confiança do valor da recompensa. Esse algoritmo é baseado no princípio do otimismo diante da incerteza, ou seja, braços que foram pouco explorados são escolhi-

dos com maior frequência para que as incertezas com relação às suas recompensas sejam reduzidas. Para controlar o dilema de *exp-exp*, esse algoritmo mede uma incerteza Σ em torno das informações disponíveis sobre o usuário e os itens, logo, esse algoritmo modela um intervalo de confiança para cada *arm* e seleciona primeiro o que tiver a maior incerteza.

2.2.3 *Thompson Sampling*

O algoritmo *Thompson Sampling* (THOMPSON, 1933), por sua vez, utiliza uma formulação baeysiana para o problema do MAB. Com o intuito de descobrir a distribuição de probabilidade real, esse algoritmo modela uma distribuição de probabilidade a partir das recompensas e as usa para modelar o *arm* com a maior esperança de recompensa. O algoritmo possui um conhecimento a priori das opções disponíveis e, conforme passam as interações com o ambiente, ele atualiza esse conhecimento visando maximizar os ganhos (AGRAWAL; GOYAL, 2011). Uma distribuição de probabilidade bem conhecida e bem versátil é a Distribuição Beta, que, a depender dos parâmetros de *alpha* e *beta*, assume um comportamento totalmente diferente. Essa versatilidade em mudar de forma faz do TS um método muito adaptável, uma vez que, se ajustarmos os valores de *alpha* e *beta* de cada arm, teremos uma distribuição de probabilidade diferente para cada *arm* e possivelmente próxima da real.

2.3 Multi-Armed Bandits em Sistemas de Recomendação

Tradicionalmente, abordagens de Filtragem Colaborativa visam encontrar itens relevantes com base nos interesses em comum entre grupos de usuários e/ou itens (BOBADILLA, 2013). Em geral, essas abordagens são categorizadas em duas classes: *memory-based* e *model-based*. Apesar da complexidade, as abordagens *model-based* alcançam melhores resultados visto que são capazes de identificar relações não-triviais (SU; KHOSHGOFTAAR, 2009; BOBADILLA, 2013). Nessas abordagens os usuários e itens são geralmente representados por um conjunto de múltiplos interesses ou aspectos. Especificamente, cada usuário $u \in U$ tem uma probabilidade de se interessar por com cada um dos aspectos $z \in Z$, de modo que usuários com preferências similares participem dos mesmos grupos de interesses. Da mesma forma, cada item $i \in I$ tem uma probabilidade de ser interessante para os usuários em cada aspecto $z \in Z$. Com essa representação, as abordagens *model-based* definem a probabilidade de interesse de u sobre i pela combinação desses interesses, como se segue:

$$P(i|u) = \sum_{z \in Z} P(i|z) \cdot P(z|u) \tag{2.3}$$

Nos últimos anos, ambas as probabilidades têm sido representadas por uma formulação matricial probabilística via PMF (*Probabilistic Matrix Factorization*), modelando a

distribuição de *rewards* pelos fatores latentes de usuários e itens (LI, 2010; ZHAO, 2013; WANG, 2017; WANG, 2018). O PMF geralmente fatora a matriz de *ratings* $M^{m \times n}$ no produto de duas matrizes *low-rank* $P \in \mathbb{R}^{m \times z}$ e $Q \in \mathbb{R}^{z \times n}$. Enquanto a matriz $P^{m \times z}$ contém o modelos de cada usuário, representando seus múltiplos interesses nos grupos z, a matriz $Q^{z \times n}$ representa a relevância do item i para os grupos z. Assim, a probabilidade $P(u|i)$ é redefinida como a associação dos fatores dos usuários e dos itens: $s(i, u) = p_u^\top \cdot q_i$.

Igualmente, nos modelos MAB a regra de predição π tem sido transformada na combinação desses fatores latentes dos modelos CF (ZHAO, 2013; HARIRI, 2014; WANG, 2017). A recompensa esperada é modelada como o produto dos fatores latentes do usuário $\boldsymbol{p_u}$ com os fatores do item $\boldsymbol{q_i}$. Tal modelagem reformula a função objetivo da seguinte forma:

$$i^*(\cdot) = \arg\max_{i(\cdot)} \sum_{t=1}^{T} \mathbb{E}\left[r_{u,i(t)}|t\right] = \arg\max_{i(\cdot)} \sum_{t=1}^{T} \mathbb{E}\left[\boldsymbol{p_u}^\top \boldsymbol{q_{i(t)}}|t\right] \tag{2.4}$$

Assim, no cenário de recomendação interativa, os esforços atuais foram concentrados em como otimizar essa função objetivo, equilibrando *exploration* e *exploitation*. As abordagens tradicionais do MAB, como ϵ-*Greedy*, *UCB* e *Thompson Sampling*, estão sendo adaptadas para considerar essa função objetivo como regra de predição (LI, 2010; CHAPELLE; LI, 2011; ZHAO, 2013; HARIRI, 2014; WANG, 2017). Após uma vasta revisão das recentes abordagens propostas, conseguimos sumarizar as principais características de cada uma das abordagens MAB na Tabela 1.

Especificamente, a diferença entre os algoritmos ϵ-*Greedy*, *UCB* e *TS* está relacionada apenas à maneira como eles controlam o dilema de *exp-exp*. Enquanto o ϵ-*Greedy* explora a regra de predição (i.e., performa *exploitation*) com probabilidade $(1 - \epsilon)$, os algoritmos *UCB* e *TS* primeiro medem uma incerteza Σ em torno das informações disponíveis sobre o usuário e os itens. *Thompson Sampling* é um algoritmo probabilístico que extrai os vetores de usuários e itens de uma distribuição normal definida pelas informações atuais disponíveis. No entanto, mesmo essa abordagem calcula a recomendação com base na combinação de p_u e q_i. A partir dessas abordagens tradicionais aqui apresentadas, diversos outros trabalhos surgiram (*Linear TS* (ABEILLE; LAZARIC, 2017), *Linear* ϵ-*Greedy* (ZHAO, 2013), *Linear UCB* (ZHAO, 2013), *GLM-UCB* (ZHAO, 2013), *PTS* (KAWALE, 2015), dentre vários outros), adaptando esses algoritmos e implementando suas próprias estratégias de lidar com o dilema de *exploration/exploitation*. Todas essas abordagens estão disponíveis no *iRec*.

2.4 Síntese do Capítulo

Neste capítulo falamos sobre a importância dos SsR e como têm sido utilizados em uma ampla variedade de aplicações. Posteriormente, retratamos o problema de reco-

ϵ-Greedy	**UCB**	**Thompson Sampling**
- Estima $p_{u,t}$ baseado em h_t - Com probabibilidade (1 - ϵ): $i_t^* = \arg\max_{i \in \mathcal{I}} (p_{u,t}^\top \cdot q_i)$ - Caso contrário: seleciona i_t^* aleatoriamente - Recebe o *reward* $r_{u,i}$ - Atualiza h_t de acordo com $r_{u,i}$	- Estima $p_{u,t}$ baseado em h_t - Estima $\Sigma_{u,i}$ por h_t e $\{q_{i \forall i \in I}\}$ - Seleciona o item: $i_t^* = \arg\max_{i \in \mathcal{I}} (p_{u,t}^\top q_i + \Sigma_{u,i})$ - Recebe o *reward* $r_{u,i}$ - Atualiza h_t de acordo com $r_{u,i}$	- Estima $\mu_{u,t}$ baseado em h_t - Estima $\Sigma_{u,t}$ baseado em h_t - Amostra $\tilde{p}_u$ de $\mathcal{N}(p_{u,t} \vert \mu_{u,t}, \Sigma_{u,t})$ - Seleciona o item: $i_t^* = \arg\max_{i \in \mathcal{I}} (\tilde{p}_{u,t}^\top \cdot \tilde{q}_i)$ - Recebe o *reward* $r_{u,i}$ - Atualiza h_t de acordo com $r_{u,i}$

Tabela 1 – Implementação padrão dos três modelos Multi-Armed Bandits paramétricos clássicos.

mendação, definindo todo o processo de interação do usuário com um sistema de recomendação, de maneira formal. Depois, falamos sobre *Multi-Armed Bandit*, que tem sido amplamente utilizado no cenário de recomendação interativa. Inicialmente, apresentamos o termo formalmente e descrevemos em detalhes como esses modelos de MAB lidam com a tarefa de recomendação online interativa. Logo após, introduzimos os tradicionais algoritmos MAB: ϵ-*Greedy* (AUER, 2002), *Upper Confidence Bounds* (UCB) (AUER, 2002) e *Thompson Sampling* (TS) (CHAPELLE; LI, 2011). Para cada um dos algoritmos, sumarizamos suas principais características e apresentamos o funcionamento de cada uma dessas abordagens. Em geral, o ϵ-*Greedy* modela o problema com base em um parâmetro de diversificação ϵ para executar a fase de *exploration*. No que concerne ao UCB, ele modela um intervalo de confiança para cada *arm* e seleciona primeiro o que tiver a maior incerteza. Por outro lado, as abordagens de TS traçam uma distribuição desconhecida para cada *arm* a fim de prever a recompensa esperada associada. Em suma, demonstramos que a diferença entre esses algoritmos está unicamente relacionada à maneira como eles controlam o dilema de *exploration/exploitation*. Por fim, demonstramos como os atuais modelos MAB surgiram a partir de derivações dessas abordagens clássicas.

3 Avaliação de Modelos de Multi-Armed Bandits em Recomendação Interativa

O principal objetivo desse capítulo é responder à **QP1**: **Como os modelos MAB vêm sendo avaliados no cenário de sistemas de recomendação interativa?**. Para isso, realizamos uma revisão sistemática da literatura de modelos MAB aplicados em sistemas de recomendação interativos, visando compreender como esses modelos são avaliados, identificando as principais metodologias, coleções de dados e métricas de qualidade consideradas. Iniciamos o capítulo apresentando o protocolo de revisão sistemática da literatura, detalhando as palavras de busca, analisando as bibliotecas consideradas e os critérios de seleção de trabalhos. Foram inspecionados um total 1.327 artigos publicados nos últimos 20 anos (2000-2020). Em seguida, realizamos uma avaliação dos trabalhos selecionados reunindo as informações sobre as principais pesquisas realizadas na área até o momento, destacando os conceitos e métodos mais utilizados, bem como suas principais características. Finalizamos o capítulo apontando as principais limitações desses trabalhos do ponto de vista de suas avaliações empíricas. Trata-se de um capítulo que sumariza o artigo (SILVA, 2022), uma das contribuições dessa dissertação de mestrado.

3.1 Revisão Sistemática da Literatura

Uma RSL é uma metodologia científica projetada para responder a algumas perguntas de pesquisa bem formuladas. Destina-se a identificar e sintetizar toda a pesquisa acadêmica sobre um determinado tópico, aplicando um protocolo rigoroso, imparcial e reprodutível. Em geral, existe um protocolo padrão, geralmente, definido por várias etapas em alto nível para não considerar a influência do tipo de questão de pesquisa nos procedimentos de revisão. Aqui, criamos um protocolo inspirado em (ÇANO; MORISIO, 2017) para auxiliar o gerenciamento do processo de revisão em três fases principais. Primeiramente, realizamos a coleta de trabalhos definindo as palavras e/ou expressões de busca e pesquisando nas fontes principais. Em seguida, na fase 2, realizamos a seleção dos trabalhos em duas etapas: (1) uma seleção grosseira, analisando o título, ano, conferência e resumo; e (2) uma seleção detalhada das publicações, lendo o artigo completo. Por fim, na fase 3, extraímos as principais informações dos artigos de acordo com nossos critérios. A Figura 2 apresenta uma visão geral de cada etapa realizada por este protocolo RSL, representando um conjunto claro de etapas que serão discutidas nas próximas seções.

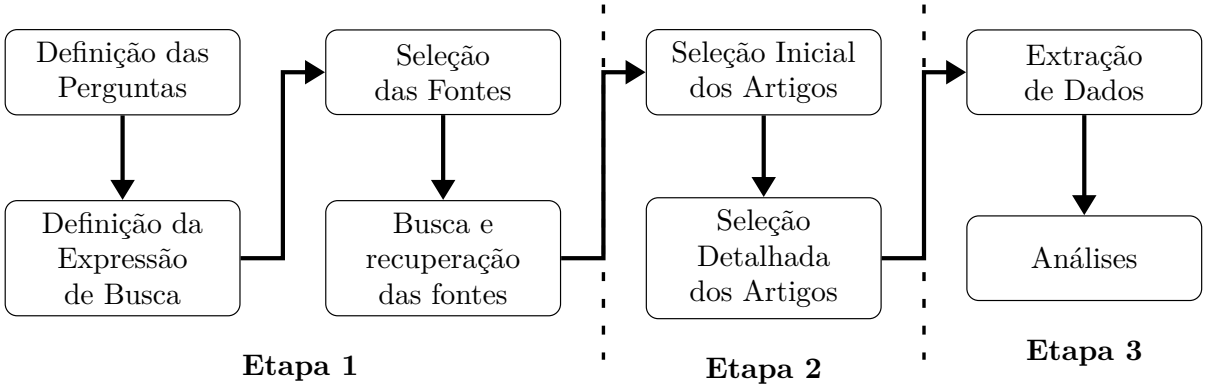

Figura 2 – Protocolo da Revisão Sistemática da Literatura.

3.1.1 Etapa 1: Pergunta de pesquisa, palavras e expressões de busca e fontes digitais

Em primeiro lugar, esta RSL define as questões de pesquisa a serem aplicadas nas principais fontes digitais para identificar e coletar os artigos mais relevantes deste corpus base. Essas questões são responsáveis por orientar todos os processos da revisão sistemática da literatura. Como nosso foco é na adoção de modelos de Multi-Armed Bandits na área de recomendação interativa, desenhamos três questões principais de pesquisa:

1: Como os modelos MAB são usados em sistemas de recomendação interativa?
2: Como os modelos MAB lidam com desafios clássicos da literatura de sistemas de recomendação, tais como esparsidade, privacidade, *cold-start* e explicabilidade?
3: Como os modelos MAB são avaliados no cenário de sistemas de recomendação interativa?

A primeira questão é mais abrangente e pode referir-se à maioria dos trabalhos na área de recomendações relacionadas a MAB. Respondendo a ela, é possível consolidar um resumo das principais pesquisas realizadas nesta área nos últimos anos. Por sua vez, a segunda questão nos permite fornecer *insights* sobre como o MAB tem sido aplicado para enfrentar alguns desafios dos RSs tradicionais, como esparsidade, privacidade, *cold start* e explicabilidade. Por fim, a terceira questão é justamente nossa *QP1* e refere-se às avaliações experimentais de trabalhos de MAB em recomendação. Analisando esses trabalhos, podemos identificar como alguns conceitos do MAB são aplicados, quais são os principais métodos adotados, quais são suas características e, até mesmo, como são avaliados neste cenário. Essas análises também podem apontar direções futuras para a literatura. Observe que as duas primeiras questões não são necessariamente o foco da dissertação ora apresentada. Conforme mencionamos, essa RSL foi realizada em um contexto mais amplo que envolvia projetos de outros pesquisadores dentro de nosso laboratório de pesquisa e o detalhamento completo da mesma pode ser encontrado em (SILVA, 2022).

Buscamos artigos científicos escritos em inglês e publicadas nas últimas duas décadas, de 2000 a 2020, nos principais repositórios conhecidos, como *IEEE Explorer*, *ACM Digital Library*, *Springer*, *Scopus*, *Cielo* e outros. Para isso, consideramos a máquina de busca Google Acadêmico, pois ela busca automaticamente uma consulta nesses repositórios. Portanto, seguimos o protocolo descrito em (KEELE, 2007) e aplicamos os critérios PICOC (População, Intervenção, Comparação, Resultados, Contexto) para gerar nossas expressões de busca (EB). Neste trabalho, o *população* é a literatura do Sistema de Recomendação, o *intervenção* é a aplicabilidade dos algoritmos MAB e os *resultados* são como são usados, critérios de avaliação e a forma como os desafios clássicos de recomendação são abordados. Essa expressão é pesquisada nos títulos, destaques e resumos de cada artigo publicado até o 2020.

```
EB: ("multi-armed bandit" OR "multi-armed bandits") AND ("recommender
↪  system" OR "recommender systems" OR "recommendation")
```

3.1.2 Etapa 2: Seleção de Artigos

Aplicando a expressão de busca EB nas fontes principais, identificamos um total de 1327 artigos, eliminando duplicatas – aquelas com o mesmo título e link de publicação. Para decidir objetivamente se deviamos ou não selecionar cada estudo preliminar para processamento adicional, definimos um conjunto de critérios de inclusão e exclusão listados na Tabela 2. Em geral, decidimos incluir apenas artigos de periódicos e conferências, deixando de fora apresentações de workshops ou artigos que relatam resumos ou slides de apresentação. Em seguida, a seleção dos estudos mais relevantes foi realizada da seguinte forma: Inicialmente analisamos título, ano de publicação e tipo de publicação (periódico, conferência, workshop etc.), descartando todos os trabalhos que não realizam estudo de recomendação ou não abordam MAB. Após essa etapa, chegamos a uma lista de 408 (30.75%) artigos que foram então examinados em uma análise mais detalhada por meio de uma leitura de sua introdução, configuração experimental e conclusão. Este processo foi feito por três pessoas que leram, pelo menos, 2/3 dos trabalhos selecionados e compartilharam seus conhecimentos para decidirem os mais importantes. Assim, cada artigo foi lido por no mínimo duas pessoas distintas, que debateram suas características e obtiveram um consenso sobre sua relevância para este estudo. Dos 408 artigos, 178 deles foram rejeitados devido aos mesmos critérios listados e 230 deles foram selecionados como estudos relevantes para nossa RSL. Esses estudos representam os trabalhos mais relevantes sobre MAB na área de recomendação.

3.1.3 Etapa 3: Extração de Dados

O processo de extração de dados foi realizado durante a segunda etapa da leitura, quando examinamos a introdução, a estrutura experimental e a conclusão de cada artigo.

Critérios de Inclusão
- Artigos apresentando algoritmos MAB no contexto de sistemas de recomendação; - Artigos que não propõem especificamente um novo algoritmo MAB, mas contrastam esses MAB com outros métodos para realizar recomendações online; - Artigos de conferências e revistas. - Artigos publicados de 2000 a 2020; - Artigos escritos em inglês;
Critérios de Exclusão
- Artigos que não abordam sistemas de recomendação; - Artigos abordando sistemas de recomendação, mas não considerando uma modelagem MAB; - Artigos que relatam apenas resumos ou slides de apresentação, faltando informações detalhadas;

Tabela 2 – Critérios de Inclusão e Exclusão.

Coletamos as principais informações de cada artigo, sendo as principais para a presente dissertação: coleções de dados utilizadas nas avaliações, estratégias de preprocessamento de dados, principais políticas de avaliação e métricas de qualidade consideradas. Na próxima seção, apresentamos os principais resultados observados em nossa coleta.

3.2 Critérios de Avaliação Empírica

Nas obras consultadas, o critério de avaliação é um protocolo definido para medir se uma técnica de recomendação é ou não eficaz em um domínio (CREMONESI, 2011; CHEN; LIU, 2017). Ao longo das décadas de pesquisa em SsR, ele tem sido continuamente adaptado e aprimorado para diversos trabalhos (VARGAS; CASTELLS, 2011; CASTELLS, 2011; VARGAS, 2011), realizando dois critérios distintos para medir a qualidade da recomendação: uma experimentação off-line; e estudo de um usuário on-line. Enquanto os experimentos off-line geralmente se preocupam com o poder de previsão, que é sua capacidade de prever com precisão as escolhas do usuário, as avaliações on-line visam medir o valor real do negócio, ou seja, o ganho de valor real que um sistema de recomendação pode alcançar para o sistema. Uma avaliação on-line é mais cara que uma avaliação off-line, pois frequentemente requer um sistema real para coletar as ações e opiniões dos usuários reais (KROHN-GRIMBERGHE, 2010; SHANI; GUNAWARDANA, 2011). Portanto, é uma prática comum identificar mais experimentos off-line do que estudos de usuários on-line, especialmente quando se está estudando novos cenários e aplicativos, sendo, portanto, nosso foco principal de nossa análise. Ao revisar os critérios de avaliação utilizados nas pesquisas, foram identificadas três etapas principais para realizar o processo avaliação off-line em modelos de *Multi-Armed Bandits*:

1. Seleção ou criação de um conjunto de dados para simular o cenário de recomendação;
2. Etapas de processamento de dados quando necessário;

3. Seleção das principais métricas de qualidade com base na metodologia utilizada para simular o consumo do item.

3.2.1 Coleções de Dados Usuais

Na aplicação MAB em SsR interativos, foram identificadas as distintas coleções de dados que são consideradas, ou mesmo criadas, para medir o desempenho de um modelo *bandit*. Uma prática usual é criar coleções de dados sintéticas para serem aplicadas estritamente para algumas análises experimentais. No entanto, isso não é incentivado porque essas coleções de dados não são reproduzíveis e não podem fornecer confiança nos resultados alcançados. A aplicabilidade de coleções de dados do mundo real é, sem dúvida, a melhor prática para analisar o desempenho de qualquer nova técnica de recomendação. Felizmente, dos artigos selecionados pelo RSL, cerca de 81% aplicam pelo menos um conjunto de dados real em sua experimentação. Na Tabela 5, destacamos os conjuntos de dados mais aplicados para avaliar algoritmos *bandits* distintos e também fornecemos como os mesmos podem ser obtidos.

Repositório	**Domínio**	**Disponível em**
MovieLens	Movies	https://grouplens.org/datasets/movielens/
Yahoo! News	News	https://webscope.sandbox.yahoo.com/
LastFM	Music	https://grouplens.org/datasets/hetrec-2011/
Delicious	Bookmarking	https://grouplens.org/datasets/hetrec-2011/
Netflix	Movies	https://kaggle.com/netflix-inc/netflix-prize-data
Jester	Jokes	https://goldberg.berkeley.edu/jester-data/
KDD - Online Ads	Advertising	https://www.kaggle.com/c/kddcup2012-track2/overview
Avazu	Advertising	https://kaggle.com/c/avazu-ctr-prediction
Amazon	Products	http://jmcauley.ucsd.edu/data/amazon/links.html
Yahoo! Music	Music	https://webscope.sandbox.yahoo.com/catalog.php?datatype=r
Millionsong	Music	http://millionsongdataset.com/
Yelp	Restaurant	https://yelp.com/dataset_challenge
RS-ASM	Smart Cities	https://kaggle.com/assopavic/recommendation-system-for-angers-smart-city
Epinions	Products	http://alchemy.cs.washington.edu/data/epinions/

Tabela 3 – Os principais repositórios com as coleções de dados mais aplicados em propostas de MAB na área de recomendação interativa.

3.2.2 O Processamento de Dados

Na maioria dessas coleções de dados mencionadas anteriormente, não há apenas as informações disponíveis sobre os usuários e itens do sistema, mas também o *feedback* que foi dado por um usuário específico a um item do catálogo. Esses *feedbacks* são tradicionalmente explícitos, quando o usuário diz explicitamente o que gosta e não gosta, ou implícitos quando não fica claro se gostou ou não de algum item. Por esse motivo, em convergência com os cenários de recomendação tradicionais, cerca de 19% dos trabalhos selecionados pela RSL realizou uma etapa de processamento de dados para limpar os dados disponíveis, removendo dados incompletos, ruídos e registros redundantes. Esses

trabalhos, normalmente, realizam a normalização de seus dados por meio das pontuações atribuídas por usuários a itens (*rating*), mapeando cada um deles de acordo com escalas estáticas ou funções específicas. Alguns trabalhos, como (CAÑAMARES, 2019), simplesmente definiram que avaliações menores ou iguais a 3 significam que o usuário não gostou do item (mapeamento como 0). Em contrapartida, as avaliações maiores que 3 significam que o usuário gostou deste item (mapeamento como 1). Outros trabalhos, como o proposto in (BASU, 2019), definem uma função específica para normalizar as avaliações convertendo cada valor para um intervalo $[0, 1]$. Portanto, percebe-se que não há um consenso sobre a melhor prática por trás da normalização, praticamente, cada trabalho implementa e utiliza o melhor método de normalização que achar conveniente. Isso se repete para qualquer outra etapa de processamento de dados e pode prejudicar muito o processo de reprodutibilidade das avaliações realizadas.

3.2.3 Metodologias e Métricas

Como o algoritmo *bandit* é um método de aprendizado on-line (veja Figura 1), a maioria dos experimentos off-line deve definir uma abordagem para simular a interação do usuário com um verdadeiro sistema. Nos artigos selecionados foram identificadas cinco principais abordagens distintas aplicadas durante a avaliação experimental:

- **Trials**: quando um *rating* é estimado para um par usuário-item em cada interação (ZHAO, 2019; KATARIYA, 2019; CHATTERJI, 2020; HAO, 2019; LI, 2016; MAY, 2012);
- **Interactions**: quando mais de um item é recomendado para o usuário (WANG, 2017; LIU, 2018; TEO, 2016; SUI, 2015; CHRISTAKOPOULOU; BANERJEE, 2018; VOROBEV, 2015);
- **Replayer:** quando o sistema tenta prever os itens consumidos cronologicamente de acordo com o histórico de consumo do usuário (ZENG, 2016; WANG, 2018; WANG, 2018; LI, 2011; TANG, 2014);
- **Leave-one-out**: quando o sistema deixa um item de fora da etapa de treinamento e tenta prever esse item específico para um determinado usuário (FELÍCIO, 2017);
- **Cross-validation**: quando o sistema executa a mesma interação do usuário várias vezes aplicando tipos distintos de dados na etapa de treinamento (HARIRI, 2014; WANG, 2014)

Com base nas análises realizadas em nossa RSL, identificamos que a metodologia mais aplicada é a *trials*, por abranger mais de 45.3% dos trabalhos selecionados. A segundo é a *Interactions*, sendo aplicada em 8.9% dos trabalhos. *Replayer*, *Leave-one-out* e *Cross-validation* são aplicadas apenas em 2.6%, 0.5%, e 1.1%, respectivamente. Possivelmente, a popularidade da *trials* está relacionada a um viés tradicional que ainda existe nos crité-

rios de avaliação de diversos trabalhos. Assim como na primeira década de pesquisas em recomendação, a maioria dos algoritmos de MAB foram avaliados apenas de acordo com seu poder de previsão, sua capacidade de prever com precisão as escolhas do usuário. No entanto, agora é amplamente aceito que previsões precisas são cruciais, mas insuficientes para implantar um bom mecanismo de recomendação (CASTELLS, 2015; CASTELLS, 2011). Em muitas aplicações, as pessoas usam um SsR para mais do que uma antecipação exata de seus gostos. Os usuários também podem estar interessados em descobrir novos itens, explorar rapidamente diversos itens, preservar sua privacidade, nas respostas rápidas do sistema e muitas outras propriedades da interação com o mecanismo de recomendação.

No entanto, esse novo consenso ainda não é uma realidade para os pesquisadores de MAB. De fato, a maioria das métricas de avaliação aplicadas são para medir o poder de previsão tradicional. Fundamentalmente, os trabalhos adaptam as métricas clássicas da área de aprendizado por reforço, tais como *rewards* ou *regrets*, para o cenário de recomendação, de acordo com os *feedbacks* recebidos pelos usuários. Os algoritmos *bandits*, portanto, focam em maximizar os *rewards* ou minimizar os *regrets*. Basicamente, essas métricas representam o erro na estimativa de uma *rating* para um par específico de usuário-item. Outras métricas, como *precision*, *recall* e *nDCG*, clássicas no campo de recomendação, foram aplicadas apenas em menos de 20% dos artigos selecionados na RSL. A Figura 3 mostra as principais métricas implementadas pelos trabalhos selecionados pela nossa RSL.

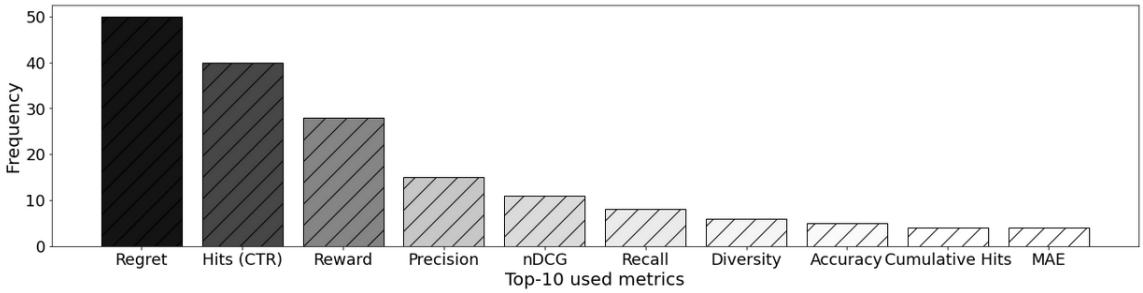

Figura 3 – Frequência de trabalhos em que é medida cada uma dessas métricas de avaliação.

Portanto, fica claro que os pesquisadores preferem adaptar as métricas tradicionais de *reward/regret* do cenário de aprendizado por reforço para esse campo de recomendação. Em geral, essas métricas são muito semelhantes às métricas de recomendação usuais aplicadas para medir o poder de previsão do sistema, como Hit Rate, *Mean Absolute Error* (MAE), *Mean squared error* (MSE) e *Root Mean Squared Error* (RMSE) (BOBADILLA, 2013). No entanto, como mencionado anteriormente, é amplamente aceito que previsões precisas são cruciais, mas insuficientes para implantar um bom mecanismo de recomendação (VARGAS; CASTELLS, 2011; VARGAS, 2014; KUNAVER; POŽRL, 2017). Assim, identificamos uma lacuna no processo de avaliação de modelos MAB aplicados em SsR interativas em que métricas de satisfação do usuário, como novidade, diversidade, seren-

dipidade, imprevisibilidade, entre outras, não são utilizadas pelos pesquisadores.

3.3 Limitações das Avaliações Empíricas

Conforme observamos em nossas análises, muitos trabalhos MAB, quase 20% deles, ainda avaliam suas propostas considerando apenas coleções de dados sintéticas de propósito específico. Por outro lado, mesmo os trabalhos que consideram pelo menos uma coleção com dados reais apresentam uma limitação importante quanto às estratégias de preprocessamento utilizadas, sem um consendo e/ou padronização de qual delas adotar. Essas duas questões trazem problemas sérios quanto à reprodutibilidade dos resultados apresentados. Outra limitação que merece destaque está nas políticas de avaliação adotadas para simular de forma off-line o comportamento interativos entre usuários e o sistema (recomendações e consumo). A estratégia mais utilizada é a *trials*, que visa estimar a avaliação de cada par usuário-item em cada interação, e ela não é ideal para medir a qualidade do sistema. Ao escolher apenas um par usuário-item para ser avaliado em cada iteração do sistema, os trabalhos atuais estão especialmente focados na capacidade de aprendizado de seu modelo. No entanto, a tarefa de recomendação não é apenas sobre o sistema. Em sua definição básica, um bom sistema de recomendação é aquele que ajuda os **usuários** a encontrar os itens mais **relevantes** de acordo com suas **necessidades** e **preferências**. Assim, pensamos que o MAB também deve caminhar na mesma direção da comunidade de recomendação, explorando algumas metodologias focadas nas interações dos usuários. Uma estratégia promissora para medir a qualidade da recomendação seria a interação de cada usuário com o sistema individualmente, pelo menos quando uma decisão é tomada online. Em um cenário do mundo real, não há apenas um usuário interagindo com o sistema por vez, como simulado por metodologias de teste. Além disso, a resposta do sistema às interações dos usuários deve ser rápida, pois eles não vão esperar por muitos testes para receberem grandes atualizações na popularidade dos itens ou nos melhores produtos. Por fim, grande parte dos trabalhos foca apenas em métricas relacionadas à acurácia das propostas apresentadas. Apesar do claro consenso existente na literatura de SsR tradicionais de que outras perspectivas de avaliação devem ser consideradas, tais como novidade e diversidade, nos trabalhos de MAB em recomendação interativas, essas não vem sendo consideradas.

Dessa forma, nossa resposta para a **QP1** é de que não há um consenso na literatura sobre métricas e políticas de avaliação padrões para o cenário de SsR interativos. Sobretudo, considerando a falta de critérios de avaliação rigorosos e bem definidos para medir a qualidade dos algoritmos MAB atuais o que é um grande problema para a comunidade. Hoje em dia, não é fácil replicar um algoritmo *bandit* simples porque não temos um protocolo de avaliação específico onde o sistema foi aplicado. Na maioria das vezes, os pesquisadores precisam replicar o algoritmo e os critérios de avaliação definidos para aquele cenário específico para garantir que implementaram tudo corretamente. Assim, um

framework desenvolvido para o cenário de recomendação, considerando as necessidades e preferências do usuário, é uma grande contribuição tanto para a academia quanto para a indústria. No próximo capítulo apresentaremos e avaliaremos os principais frameworks existentes na literatura.

3.4 Síntese do Capítulo

Nesse capítulo foi apresentada uma revisão sistemática da literatura sobre modelos Multi-Armed Bandits aplicados em sistemas de recomendação interativos. A RSL aplicada foi ampla e abordou três questões principais, a primeira mais geral, visando prover uma visão geral dos principais trabalhos realizados nos últimos anos; a segunda visando entender como MAB é utilizado para abordar os principais desafios em recomendação tradicional (esparsidade, privacidade, *cold start* e explicabilidade); a terceira visando analisar como são feitas as avaliações experimentais de propostas MAB no cenário de recomendação interativas. Trata-se de uma RSL realizada sob um contexto mais amplo, envolvendo outros projetos de pesquisa (SILVA, 2022), todavia nosso foco de análise foi na terceira questão, a qual está relacionada a nossa **QP1**: **Como os modelos MAB vêm sendo avaliados no cenário de sistemas de recomendação interativa?**.

Analisando um total 1.327 artigos publicados nos últimos 20 anos (2000-2020), observamos que existe uma grande variabilidade de coleções de dados consideradas no processo de avaliação de modelos MAB em recomendação interativa e que muitas dessas coleções, cerca de 20%, ainda são sintéticas, dificultando o processo de reprodutibilidade. Observamos também que não existe um consenso quanto às estratégias de pré-processamento que devem ser aplicadas. Quanto às políticas de avaliação verificamos limitações quanto à simulação off-line de recomendação interativa. As avaliações ora são focadas apenas no processo de aprendizagem, o que está muito relacioado à teoria de aprendizagem por reforço, a qual foi adaptada ao cenário de recomendação, ora são focadas apenas no resultado final das recomendaçãoes (acurácia das recomendações aos usuários). Não identificamos políticas de avaliação que consideram ambos objetivos. Outra lacuna identificada são nas métricas utilizadas nas avaliações, focadas quase que exclusivamente em acurácia, sem considerar outras perspectivas de satisfação dos usuários (novidade, diversidade, etc.). Em síntese, essas avaliações apontam, em resposta a **QP1**, que não existe uma padronização clara e reprodutível para estratégias de MAB aplicadas em sistemas de recomendação interativos.

4 Trabalhos Relacionados

Conforme constatamos nos capítulos anteriores, apesar dos avanços recentes e dos novos algoritmos de MAB publicados para o cenário de recomendação interativa, nenhum trabalho se preocupou em propor ou identificar uma avaliação experimental reprodutível e justa. Mesmo com o crescente consenso sobre a reprodutibilidade, os trabalhos atuais não fornecem detalhes completos sobre seus experimentos e nem mesmo seus códigos fonte. Além disso, é comum que cada artigo aplique sua metodologia específica e não siga o mesmo padrão de avaliação, e, infelizmente, contrastando com o consenso atual da comunidade RS para medir a experiência e o engajamento do usuário, a maioria dos trabalhos MAB se preocupam apenas com o poder de previsão de cada método (avaliando a métrica tradicional de *reward/regret* a partir do aprendizado por reforço).

Dessa forma, este capítulo tem como objetivo responder à segunda pergunta deste trabalho: **QP2:** *Existe na literatura algum framework capaz de englobar todas etapas (desde a coleta dos dados até a avaliação dos resultados) para experimentação offline de um sistema de recomendação no cenário interativo?* Para responder a essa pergunta, revisamos a literatura em busca de trabalhos que proponham frameworks e/ou bibliotecas que visem lidar com a avaliação de algoritmos MAB no cenário de recomendação interativa. Após uma ampla inspeção da literatura, encontramos diversos frameworks voltados para o cenário de recomendação, como o *Elliot* (ANELLI, 2021), *Cornac* (SALAH, 2020), *LibRec* (GUO, 2015) dentre vários outros trabalhos amplamente conhecidos na área. A maioria desses trabalhos surgiram devido ao número impressionante de algoritmos de recomendação propostos, estratégias de divisão, protocolos de avaliação, métricas e tarefas tornaram a avaliação experimental rigorosa particularmente desafiadora. Intrigados e frustrados com a recriação contínua de benchmarks de avaliação apropriados, pipelines experimentais, otimização de hiperparâmetros e procedimentos de avaliação, desenvolveram uma estrutura exaustiva para atender a essas necessidades. No entanto, nenhum desses trabalhos são focados em SsR interativos utilizando modelos MAB. Sendo assim, para um estudo comparativo, buscamos trabalhos que lidam com o processo de recomendação interativa utilizando modelos MAB. Dessa forma, encontramos apenas três trabalhos que recentemente tentaram lidar com a avaliação desses modelos. Os dois primeiros são códigos abertos com algumas soluções MAB implementadas e a possibilidade de realizar uma avaliação experimental (STRONG, 2021; SAITO, 2020). Apenas um é nomeado como *framework* para o domínio *bandit* (BARRAZA-URBINA, 2018), mas mesmo este trabalho não apresenta um ambiente completo para simular cenários bandits no campo de recomendação. Sendo assim, neste capítulo apresentamos em detalhes cada um dos trabalhos encontrados, levantando todas as suas características, vantagens e desvantagens, além de realizar um estudo

de comparação completo dos mesmos com o *framework iRec*, proposto nessa dissertação.

4.1 Open Bandit Pipeline (OBP)

Open Bandit Pipeline (OBP) (SAITO, 2020) é uma biblioteca desenvolvida em *Python*, de código aberto, que inclui uma série de módulos para implementar o pré-processamento de coleções de dados e diversos métodos de recomendação. Diferente da maioria das bibliotecas e *frameworks* de recomendação existentes, essa biblioteca tenta lidar com o viés das bases de dados existentes para o cenário de recomendação. Atualmente, os *datasets* utilizados durante a fase de treinamento do modelos de recomendação são basicamente logs provenientes de um determinado recomendador, contendo informações sobre as interações entre usuários e itens. No entanto, sabemos que o processo de recomendação muda a forma com que os usuários interagem com o sistema, seja por clicks, avaliações e etc. Dessa forma, ao usar esses dados de interação dos usuários, estamos ignorando a natureza interventiva das recomendações. Como resultado, não estamos avaliando se os usuários clicariam ou comprariam mais devido às nossas novas recomendações, mas sim até que ponto as novas recomendações se ajustam aos dados registrados, e é justamente esse problema que o OBP tenta minimizar através de uma política de avaliação denominada *Counterfactual*. A avaliação *Counterfactual* tenta responder "o que teria acontecido se mostrássemos aos usuários nossas novas recomendações em vez das recomendações existentes?" (MCINERNEY, 2020). Isso nos permite estimar os resultados de testes A/B em potencial sem realmente executá-los. Em testes A/B, as recomendações são mostradas aos usuários, logo após é obtido o *feedback* dessas recomendações e, em seguida, medimos como as métricas, como taxa de cliques e conversão, mudam. No entanto, além de requerer mais esforço em relação à avaliação off-line, os ciclos de experimentos podem ser longos, pois precisamos de dados suficientes para fazer uma análise ideal. Além disso, podemos não ter acesso fácil aos testes A/B uma vez que estamos trabalhando no lado da pesquisa.

Na Imagem 4, é apresentada uma visão geral desta biblioteca, na qual podemos ver toda estrutura e seus principais módulos. Em suma, esta biblioteca pode ser dividida em quatro módulos principais: um módulo de conjunto de dados; um módulo em que são descritos os modelos implementados; um módulo de simulação; e, finalmente, um módulo de política de avaliação.

- *Dataset Module*: Como o próprio nome diz, esse módulo fornece mecanismos para trabalhar nas coleções de dados, desde a etapa de carregamento dos dados, etapas de pré-processamento básicas, até métodos para gerar dados sintéticos, dentre outros.
- *Policy Module*: Este módulo oferece interfaces para implementação de novos métodos *bandits* e novas políticas de avalição, além de já possuir diversos modelos e políticas de avaliações relevantes da literatura, tanto para o cenário online quanto off-line.

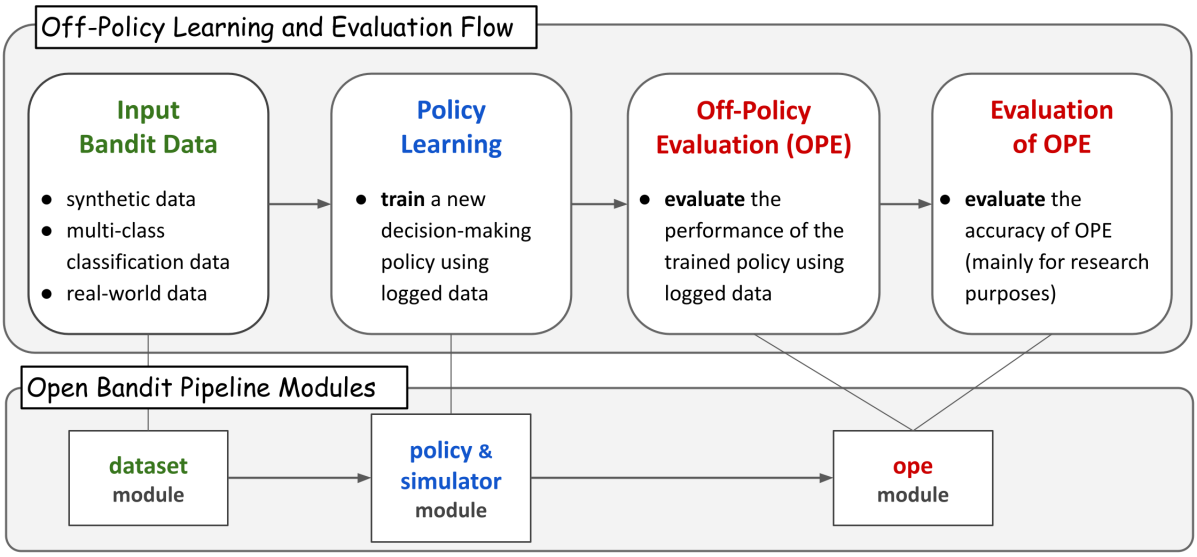

Figura 4 – Visão geral da biblioteca Open Bandit Pipeline (SAITO, 2020).

- *Simulator Module*: Este módulo fornece funções para realizar simulação off-line de modelos *bandits*. Por meio dele é possível comparar e avaliar o desempenho de algoritmos MAB.
- *Off-policy evaluation (OPE) Module*: Este módulo possui interfaces abstratas genéricas, ideais para implementações personalizadas, nas quais os usuários podem adicionar e avaliar seus algoritmos, além de poder usufruir das implementações já existentes.

4.2 Bandit-based Recommender System Evaluation (BEARS)

BEARS (BARRAZA-URBINA, 2018) é a única estrutura proposta até agora para mitigar a falta de padrão nas avaliações de algoritmos *bandits* e suportar avaliações off-line reproduzíveis. Ela é implementada em Python e consiste em blocos de construção simples que permitem configurar facilmente variantes para *Agents* (abordagens de solução) e *Environments* (configurações de problemas). Sua ideia principal é propor um ambiente flexível que possa ser estendido para incluir outros algoritmos e métricas de avaliação de última geração. No entanto, esse *framework* contém apenas alguns algoritmos clássicos e ainda está limitado às métricas de precisão tradicionais do cenário aprendizado por reforço. Ademais, o BEARS não fornece nenhuma estratégia de preparação de dados e não permite nenhum ajuste de hiperparâmetro para os algoritmos bandit mais recentes. No Capítulo 5 veremos que a estrutura utilizada no *iRec* é bem similar à utilizada pelo BEARS. Isso se deve ao fato desse ser um dos primeiros frameworks proposto para lidar com a avaliação de modelos MAB e mais ainda pelo fato de possuir uma excelente arquitetura, extremamente modularizada e flexível.

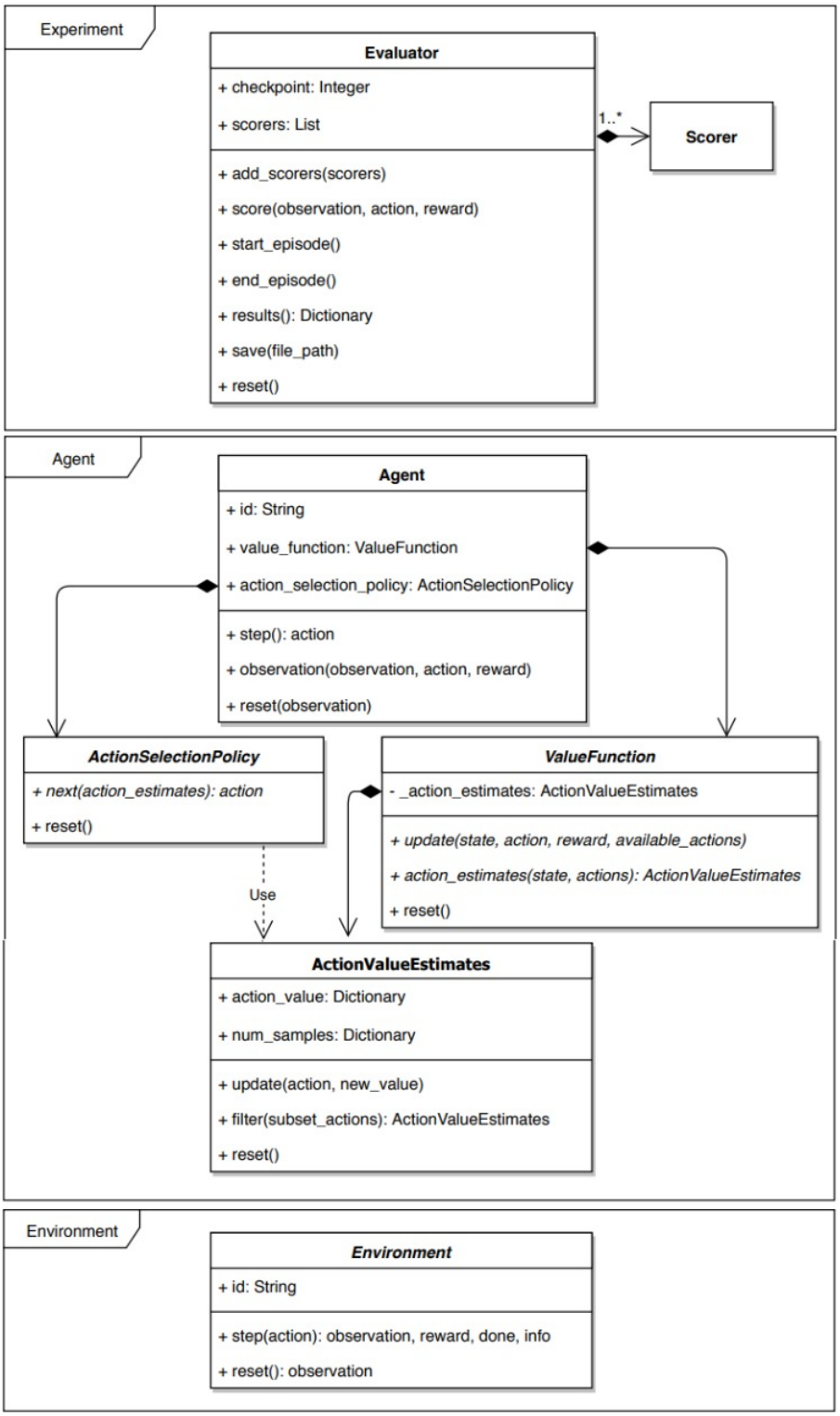

Figura 5 – Visão geral do *framework* BEARS (BARRAZA-URBINA, 2018).

Os principais componentes da estrutura BEARS podem ser vistos como um diagrama de classes na Figura 5. A estrutura foi projetada para representar a configuração de um problema de aprendizado por reforço, mas sua aplicabilidade na versão atual da estrutura só foi validada para abordagens de SsR baseadas em algoritmos MAB. Como pode ser visto na Figura 5, os elementos centrais que estruturam o BEARS são os componentes

Experiment, *Agent* e *Environment*.

- *Agent*: O recomendador é considerado um agente de aprendizagem. Em cada etapa de tempo, o *Agent* fornece uma recomendação, ou seja, executa uma ação, dado um contexto/estado, por exemplo, usuário-alvo atual, do ambiente de recomendação (*Environment*). Após oferecer uma recomendação (item único, lista de classificação), o *Agent* receberá uma recompensa do *Environment*. Em recomendação, as recompensas (*rewards*) são geralmente um valor escalar representativo do feedback explícito/implícito do usuário.

- *Environment*: O *Environment* (ambiente) é o domínio ou cenário onde o *Agent* é implantado. Ele incorpora a configuração do problema que o *Agent* pretende resolver, assim, um ambiente representa a interação dos usuários e itens específicos em um determinado cenário.

- *Experiment*: Em BEARS, um experimento é focado em avaliar o comportamento de um *Agent* interagindo com um *Environment* em um processo de tomada de decisão sequencial. Para avaliar facilmente o comportamento do agente ao longo da interação com base em várias métricas, o BEARS fornece um componente específico para avaliação.

4.3 MABWiser

MABWiser (STRONG, 2021; STRONG, 2019) é uma biblioteca de código aberto implementada em Python para lidar com algoritmos MAB com prototipagem rápida e ajuste de hiperparâmetros. Ele fornece simulações em lote e on-line para soluções MAB tradicionais e algoritmos *bandits* contextuais, ou seja, adaptações para o campo de recomendação. Além dos algoritmos clássicos, como ϵ-Greedy, UCB, TS e Softmax (KULESHOV; PRECUP, 2014), ele também contém alguns algoritmos *bandits* paramétricos e não-paramétricos. Os modelos paramétricos incluem técnicas baseadas em regressão, como LinUCB e LinTS. Em termos de não-paramétricos, eles implementam algumas abordagens de *clustering*, como k-means e algoritmos *K-Nearest Neighbors* (KNN).

A arquitetura da biblioteca *MABWiser* é apresentada na Figura 6. Essa biblioteca pode ser decomposta em três camadas com níveis crescentes de complexidade: a camada de interface pública (A), a camada de integração (B) e a camada de implementação (C).

- A. Camada de Interface Pública: Essa interface fornece a interação dos usuários da biblioteca com a classe MAB mostrada na parte A da Figura 6. Através dela, é possível acessar os métodos disponíveis publicamente para a fase de treinamento e de teste. Essa biblioteca segue um padrão de implementação semelhante ao utilizado

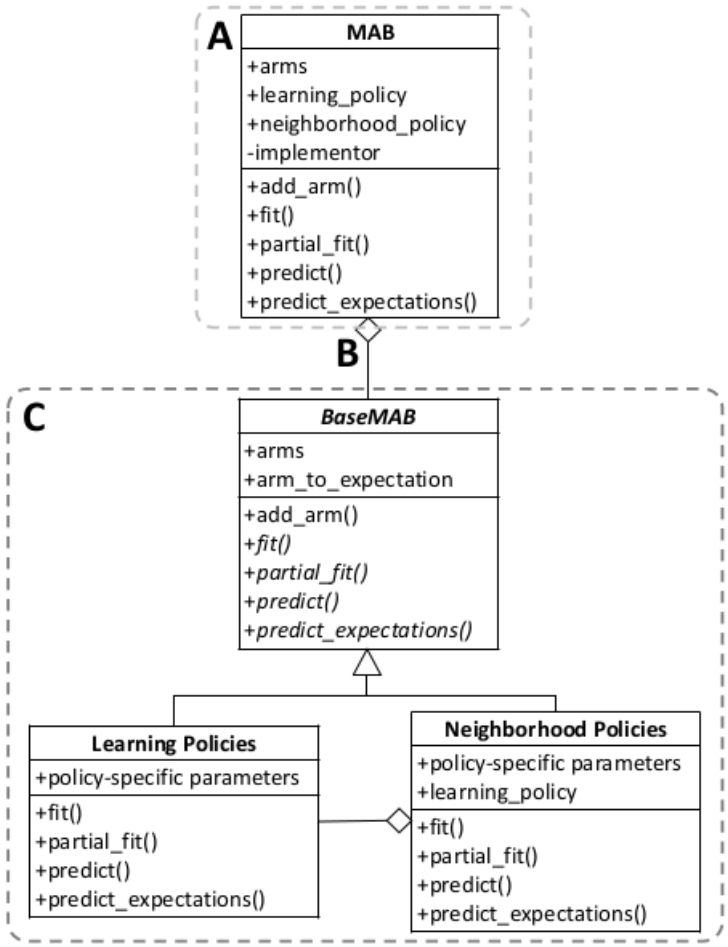

Figura 6 – Visão geral da biblioteca MABWiser (STRONG, 2021; STRONG, 2019)

por outras bibliotecas de *Python*, como *scikit-learn*. Isso permite que os modelos MAB se comportem de maneira similar aos modelos tradicionais de aprendizado de máquina, o que facilita a sua adoção e uso.

- B. Camada de Integração: A classe MAB apresentada na Figura 6 também serve como uma camada de integração entre a interface pública na parte A e a implementação subjacente, a classe BaseMAB, na parte C. Em síntese, essa camada é utilizada para desacoplar a interface pública dos detalhes de implementação dos algoritmos. O usuário interage com a interface e, por sua vez, a interface delega solicitações de treinamento e teste a camada de implementação. Isso não apenas simplifica o uso da perspectiva do usuário, mas também garante que a biblioteca possa continuar a evoluir no futuro sem interferir na camada pública.

- C. Camada de Implementação: Sugestivamente, a classe BaseMAB é a classe base da implementação de cada algoritmo. Estes herdam dessa classe e implementam seus métodos de treinamento e teste com base em seu algoritmo de aprendizado. Pesquisadores com experiência em algoritmos MAB podem introduzir novos algoritmos e novas políticas personalizadas seguindo a assinatura dos métodos *fit* e *predict*,

que incluem anotações de tipo e dicas de uso.

4.4 Avaliação Comparativa das Bibliotecas e Frameworks existentes com iRec

Nesta seção apresentamos uma avaliação comparativa entre as três bibliotecas/frameworks apresentados com o *framework* proposto nessa dissertação, o *iRec*. A Tabela 4 apresenta uma visão geral de todos os pontos cobertos pelas bibliotecas e frameworks identificados na literatura. Ela compara tudo o que se espera em uma estrutura de avaliação, desde a preparação dos dados até o processo de avaliação. Nesta tabela, também comparamos nossa proposta, o *iRec*, para destacar sua maior cobertura com as outras propostas existentes. Diferente das atuais bibliotecas e *frameworks*, o *iRec* engloba integralmente o processo experimental de avaliação de um sistema de recomendação, desde a entrada dos dados até a metodologia de avaliação. Ele permite que os usuários avaliem um caso de estudo completo da perspectiva esperada em um sistema de recomendação. Especificamente, podemos ver que, de maneira singular, nosso módulo de preparação de dados possui diversas estratégias de filtragem e divisão de dados de extrema importância no cenário de recomendação. Além disso, *iRec* conta com um dos métodos mais utilizados para fazer a tunagem de parâmetros de modelos de aprendizado de máquina: o *GridSearch*. Para modelos complexos, como os baseados em rede neural, esta etapa é essencial para garantir a qualidade de tais recomendações.

Assim como no módulo de preparação de dados, nosso *framework* também se destaca nos modelos de recomendação disponíveis, sendo o que mais possui modelos MAB e modelos de recomendação não-personalizados de última geração. Os demais trabalhos contam apenas com alguns algoritmoss clássicos de MAB. No mais, é importante destacar que nenhum dos três trabalhos mencionados (*BEARS, OBP e MABWiser*), possuem métricas de avaliação voltadas para o cenário de recomendação, o que demonstra ainda mais que os atuais pesquisadores preferem adaptar as métricas tradicionais do cenário de aprendizado por reforço para esse campo de recomendação. Por meio da Tabela 4, podemos ver que somente o *iRec* é capaz de medir, de diferentes formas, a qualidade das recomendações.

Dessa forma, as comparações apresentadas acima apontam que, em resposta à **QP2**, que não existe na literatura algum *framework* capaz de englobar todas etapas (desde a coleta dos dados até a avaliação dos resultados) para experimentação off-line de um sistema de recomendação no cenário interativo, sendo o *iRec*, portanto, a primeira proposta a contemplar todas elas. No próximo capítulo faremos uma apresentação detalhada de nosso framework.

Framework	Preparação dos Dados								Modelos de Recomendação																		Avaliação				
	Pré-filtragem			Divisão			Tunagem		Multi Armed Bandits																Não-personalizados					Métricas				
	Por rating	Por usuário	Por item	Temporal	Random	Fixed	Grid Search	Random	ϵ-Greedy	UCB	TS	Softmax	ICTR	PTS	kNN Bandit	Linear ϵ-Greedy	Linear TS	Linear UCB	NICF	COFIBA	GLM-UCB	ClusterBandit	Random	Popular	Best Rated	Entropy	log(pop)*ent	Acurácia	Novidade	Diversidade	Cobertura	Erro
BEARS									✓	✓	✓												✓									✓
OBP									✓		✓					✓	✓	✓					✓									
MABWiser								✓	✓	✓	✓	✓					✓	✓					✓	✓								
iRec	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

Tabela 4 – Uma visão geral das funcionalidades de cada *framework* proposto na literatura e nosso *iRec*

4.5 Síntese do Capítulo

O objetivo deste capítulo foi responder à segunda pergunta deste trabalho: **QP2** *Existe na literatura algum framework capaz de englobar todas etapas (desde a coleta dos dados até a avaliação dos resultados) para experimentação offline de um sistema de recomendação no cenário interativo?*. Para respondê-la, fizemos uma ampla revisão literária na qual encontramos apenas três trabalhos que tentam lidar com a avaliação dos modelos MAB no cenário de recomendação interativa: *BEARS* (BARRAZA-URBINA, 2018), *OBP* (SAITO, 2020) e *MABWise* (STRONG, 2021; STRONG, 2019). Para cada uma dessas propostas, apresentamos detalhadamente a arquitetura de cada uma delas, bem como seus principais componentes. Em seguida, realizamos uma comparação dessas três propostas encontradas com o nosso *Framework iRec*. Para demonstrar a superioridade deste, construimos uma tabela que apresenta uma visão geral do módulo de preparação dos dados, dos modelos de recomendação disponíveis em cada framework/biblioteca e do módulo de avaliação. Através da Tabela 4, foi possível verificar que nenhumm dos trabalhos existentes até o momento engloba integralmente o processo experimental de avaliação de um sistema de recomendação, desde a entrada dos dados até a metodologia de avaliação, sendo o *iRec* o primeiro e único. Dessa forma, no próximo capítulo, apresentaremos *iRec*, descrevendo toda a sua estrutura, além dos detalhes de como é feito todo esse processo experimental.

5 iRec - Um *framework* para Recomendações Interativas

De acordo com o capítulo anterior, no qual apresentamos os trabalhos relacionados a este, vimos que, apesar os esforços atuais em prover um ambiente capaz de cobrir a lacuna presente no processo de avaliação das soluções de MAB, nenhum deles está preparados para amenizar este problema. Através da Tabela 4, fizemos uma comparação das funcionalidades de cada *framework* proposto na literatura com o *iRec*. Com base nessas informações, observamos que apenas nosso *framework* está apto o suficiente para englobar todas as etapas necessárias para a criação de um experimento *offline* (i.e. coleta e preparação dos dados até a avaliação dos resultados) no cenário de SsR interativos. Sendo assim, neste capítulo, apresentamos o *iRec*[1] e, consequentemente, respondemos à **QP3: É possível prover um ambiente de execução reprodutível no cenário de recomendação interativa capaz de amenizar a lacuna presente no processo de avaliação das soluções de MAB?**. O *iRec* é um *framework* proposto para viabilizar o uso de modelos interativos, em especial aqueles baseados em modelos MAB, no domínio de recomendação.

O *iRec* é composto de três componentes principais que abrangem todo o processo de experimentação: (1) a construção de um **Environment**; (2) a definição de **Recommendation Agent**; e (3) a definição de uma **Experimental Evaluation**. No componente *Environment* setamos toda a estrutura dos dados a serem processados pelo *framework*. Nele, carregamos as bases de dados desejadas e definimos todos os módulos de preparação de dados a serem aplicados a elas, tais como as estratégias de pré-filtragem e divisão dos conjuntos de treino e teste. Por sua vez, no componente *Recommendation Agent* define-se o modelo que será utilizado para definir o(s) melhor(es) item(ns) para cada usuário em cada iteração. Em outras palavras, é nesse componente que implementam-se os SsR que serão utilizados na recomendação. O *iRec* possui diversos algoritmos já implementados. Por fim, o componente *Experimental Evaluation* é responsável por realizar a integração dos modelos propostos no *Recommendation Agent* sobre os dados especificados no primeiro componente - *Environment*. Primeiramente, precisamos definir como deve ser a interação entre esses dois componentes por meio de um módulo denominado *Evaluation Policy*. Nele, configuramos um ambiente de Aprendizado por Reforço, no qual um item (ou um conjunto de itens) é recomendado em cada iteração do algoritmo (tentativa). Por um tempo pré-determinado (número de tentativas), o *Recommendation Agent* realiza as recomendações dentro do *Environment*, recebendo uma recompensa positiva ou negativa

[1] O *iRec* está disponível em: https://github.com/irec-org/irec

e atualizando seu conhecimento (se necessário). Todos os *logs* (i.e., registro das ações realizadas pelo *Recommendation Agent* e recompensas fornecidas pelo usuários, de acordo com os dados no conjunto de teste) são armazenados. Por fim, o *Experimental Evaluation* analisa esses logs, aplica as métricas de avaliação de recomendações e realiza os testes estatísticos necessários para realizar uma avaliação adequada do desempenho dos SsR implementados no *Recommendation Agent*. Esses componentes são ilustrados na Figura 7 por cores diferentes e são discutidos em detalhes nas próximas seções.

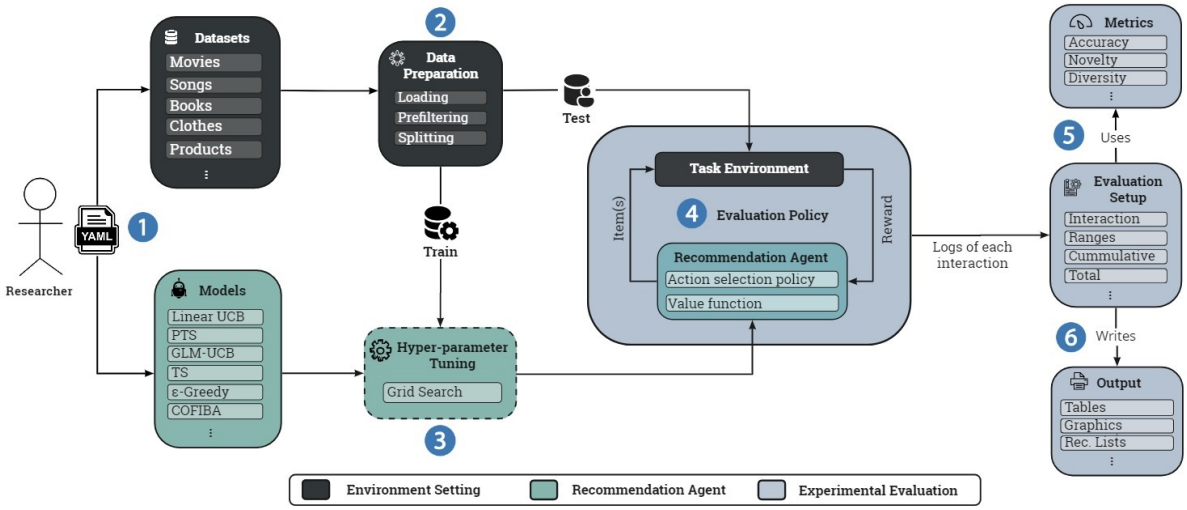

Figura 7 – Uma visão geral da estrutura iRec. É composto por três componentes principais que permitem ao pesquisador simular um cenário de recomendação interativo e comparar algoritmos distintos por meio de uma avaliação justa de sua qualidade.

5.1 Environment Setting

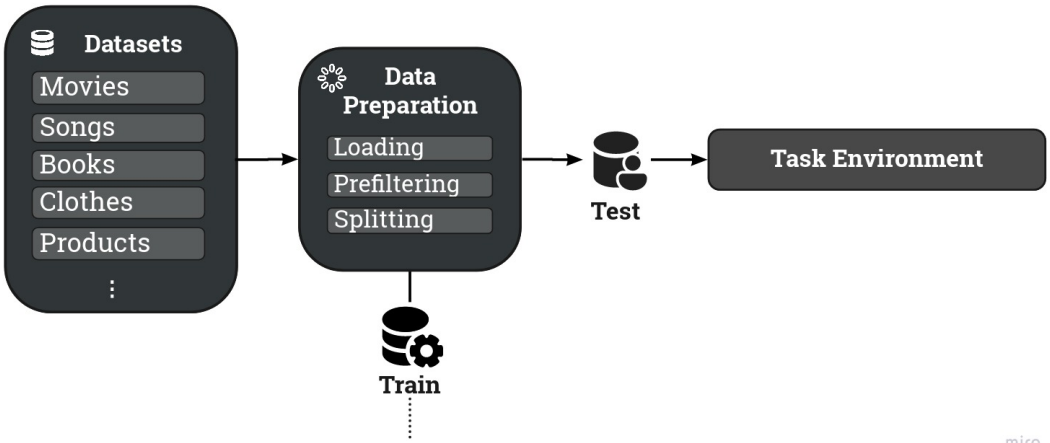

Figura 8 – Visão geral do componente *Environment*

O componente *Environment* visa preparar os conjuntos de dados a serem considerados pelo *framework*. Como podemos ver através da Figura 8, ele é gerenciado pelo

módulo *Data Preparation*, que identifica a(s) base(s) de dados selecionada(s) e aplica três operações principais: *Loading*, *Filtering* e *Splitting*. As estratégias disponíveis para cada operação estão descritas a seguir.

Loading. O *iRec* possui duas estratégias diferentes:

- *Full Data*: determina que o sistema deve ler todo o conjunto de dados e aplicar uma estratégia de divisão;
- *Split Data*: determina que os dados já estão divididos em treinamento e teste e devem ser lidos separadamente.

Em ambas, deve-se definir o caminho do conjunto de dados e seu formato. O *iRec* segue o padrão em conjuntos de dados de recomendação, em que cada linha contém o seguinte formato: *userId, itemId, rating, timestamp*. Caso se deseje alterá-lo para outros conjuntos de dados, deve-se criar uma nova estratégia que estenda o operador *Loading* com um novo formato. A Tabela 5 lista os conjuntos de dados atuais aceitáveis pelo *iRec*.

Conjunto de Dados	Domínio	Usuários	Itens
MovieLens	Filmes	138,493	26,744
Netflix	Filmes	429,584	17,770
Yahoo Music	Musica	10,000	13,214
LastFM	Musica	5,000	36,718
Good Books	Livros	53,424	10,000
Good Reads	Livros	14,639	9,999
Amazon Store	Produtos	14,776	68,921
Clothing Fit	Roupas	105,508	5,850
Yelp	POIs	1,987,929	150,346

Tabela 5 – Uma visão geral de alguns conjuntos de dados aceitaveis pelo iRec.

Filtering. Este operador visa aplicar as estratégias de pré-processamento. Em geral, o pré-processamento é uma tarefa comum na área de recomendação, pois os conjuntos de dados geralmente possuem muitas informações que podem afetar o tempo de execução de um experimento. Além disso, para evitar problemas de esparsidade, é comum nesse cenário filtrar (remover) alguns dados (usuários e/ou itens) seguindo determinados critérios. Por isso, o *iRec* possui três estratégias implementadas para filtrar os dados:

- *Users*: remove aleatoriamente um número predefinido de usuários ou aqueles que consumiram um total de itens inferior a um *threshold*;
- *Items*: remove aleatoriamente um número pré-definido de itens ou aqueles que foram consumidos por um total de usuários inferior a um *threshold*;

- *Ratings*: filtra todos os pares de usuário-item que resultaram em avaliações inferiores a um *threshold*.

Splitting. Este operador visa definir os conjuntos de treinamento e teste. Cada abordagem de divisão seleciona um grupo de usuários (e suas avaliações sobre itens) para um conjunto e o grupo restante de usuários para o outro conjunto. Esse operador também permite a criação de um conjunto de validação, a partir do conjunto de treinamento, para ser utilizado nos ajustes de hiperparâmetros dos SsR definidos no *Recommendation Agent*. As três abordagens implementadas são:

- *Random*: seleciona aleatoriamente os usuários para os conjuntos de treino e teste;
- *Temporal*: seleciona os últimos usuários a ingressar no sistema com base em seu primeiro *timestamp*. A ideia é evitar o viés de quebrar a ordem cronológica das ações do usuário (ANELLI, 2019; BELLOGÍN; SÁNCHEZ, 2017);
- *Global Timestamp*: seleciona um *timestamp* global do conjunto de dados para criar os conjuntos de treinamento e teste. Todas as interações dos usuários que aconteceram antes desse *timestamp* serão o treino. O restante de suas interações é descartado. Os usuários que acabaram de ingressar no sistema depois desse corte temporal são definidos no teste;
- *User History*: seleciona x% do histórico de cada usuário para compor o treino e o restante é direcionado para o teste.

5.2 Recommendation Agent

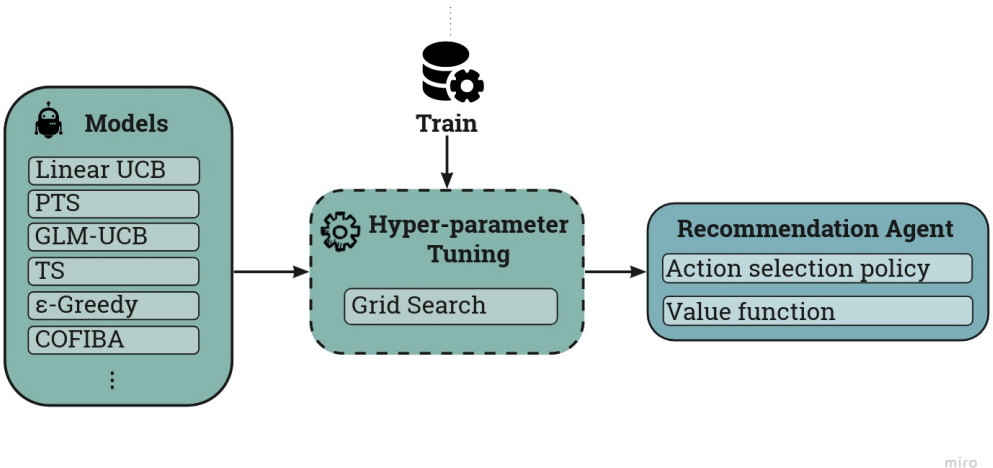

Figura 9 – Visão geral do componente *Recommendation Agent*

Este componente visa definir um agente (i.e., um modelo de recomendação) para interagir com o componente *Environment*, fazendo recomendações e recebendo *feedback*

(i.e. avaliações) do usuário. Através da Figura 9, vemos que semelhante às abordagens usadas na teoria de Aprendizado por Reforço, um agente é representado por dois operadores principais: uma **Value Function** e uma **Action Selection Policy** . A *Value Function* representa os objetivos do agente, quantificando as consequências esperadas de suas decisões. Neste caso, no qual o agente é um sistema de recomendação, a *Value Function* representa a utilidade de cada item para um usuário de acordo com a previsão do algoritmo. O *reward* geralmente consiste em um valor escalar representativo do *feedback* explícito/implícito do usuário. Por sua vez, a *Action Selection Policy* representa a política utilizada pelo agente para escolher os itens a serem recomendados. Em geral, tais políticas têm dois objetivos concorrentes: *exploitation*, que mira os itens com as maiores recompensas no passado; ou *exploration*, que considera os itens desconhecidos para melhorar o conhecimento do sistema. Assim, a definição de um agente que implementa o ϵ-greedy, por exemplo, consiste em definir uma *Value Function* como sua função objetivo original e implementar sua *Selection Policy* que explora aleatoriamente novos *arms* (ou opções) com probabilidade ϵ.

Recommendation Models. Atualmente, o *iRec* fornece vários algoritmos bandits de última geração da literatura:

- **ϵ-Greedy** (AUER, 2002): modelo clássico de *bandit* que explora aleatoriamente outros *arms* com probabilidade ϵ.
- **UCB** (AUER, 2002): calcula um intervalo de confiança para cada item e tenta reduzir os limites a cada iteração.
- **Thompson Sampling (TS)** (CHAPELLE; LI, 2011): utiliza uma distribuição gaussiana para prever com base em amostras.
- **Linear TS** (ABEILLE; LAZARIC, 2017): uma adaptação linear do TS original usando *Probabilistic Matrix Factorization* (PMF).
- **Linear ϵ-Greedy** (ZHAO, 2013): exploração linear dos fatores latentes definidos pelo PMF do ϵ-Greedy clássico.
- **Linear UCB** (ZHAO, 2013): uma adaptação do LinUCB (LI, 2010) original para medir as dimensões latentes por uma formulação PMF.
- **CLinUCB** (SILVA, 2021): utiliza uma abordagem contextual sobre o método *Linear UCB*.
- **GLM-UCB** (ZHAO, 2013): adapta o *Linear UCB* com uma *Sigmoid Function* que realiza explorações com base no tempo.
- **ICTR** (WANG, 2018): é um modelo TS de regressão de tópicos que relaciona itens por uma estratégia de aprendizado de partículas.

- **kNN Bandit** (SANZ-CRUZADO, 2019): uma variante do *nearest-neighbours* aplicando um algoritmo TS clássico sem parâmetros.
- **NICF** (ZOU, 2020): uma rede neural de filtragem colaborativa que realiza uma meta-aprendizagem das preferências do usuário.
- **COFIBA** (LI, 2016): define uma *upper-confidence-based* e a combina com um *cluster* adaptativo de usuários e itens.
- **PTS** (KAWALE, 2015): é uma formulação PMF para o TS que aplica filtragem de partículas para orientar a exploração dos itens.
- **Cluster-Bandit (CB)** (SHAMS, 2021): é um novo algoritmo *bandit* baseado em *clusters* para lidar com o problema de *cold-start*.

Hyperparameter Tuning. Analisando a Figura9, observa-se que o *iRec* permite que se façam otimizações de hiperparâmetros em seus algoritmos. Para modelos complexos, como os baseados em rede neural, esse módulo é essencial para garantir a qualidade de tais recomendações (RENDLE, 2020). Atualmente, nosso *framework* fornece uma estratégia *Grid-Search* para realizar uma busca exaustiva pela melhor combinação de hiperparâmetros. Essa busca é feita de acordo com uma faixa de valores definida pelo pesquisador como parâmetro para o módulo *Tuning*. De acordo com os parâmetros necessários para a *value function* selecionada para o experimento, o pesquisador pode definir valores fixos ou um intervalo de busca a ser explorado pelo tuner. Com base nessas definições, o *iRec* executará automaticamente o *Recommender Agent* (com sua *Value function* e *Action Selection Policy*) para todos os parâmetros da busca, este processo é feito em paralelo a fim de diminuir o tempo de execução. Ao final, este módulo armazena os melhores parâmetros encontrados. Este é um passo opcional, representado por uma caixa tracejada na Figura 7. Os modelos são otimizados de acordo com a *Evaluation policy* e *Evaluation Setup* detalhada na próxima seção.

5.3 Experimental Evaluation

Como podemos ver na Figura 8, esse componente é responsável por definir o comportamento de um *Agent* em um *Environment* e, posteriormente, avaliar a qualidade dessa interação. Seu objetivo é fornecer uma avaliação justa e reprodutível para vários algoritmos no cenário de recomendação interativa, por meio de métricas e metodologias especificamente propostas pela comunidade de SR nos últimos anos (RICCI, 2011; VARGAS; CASTELLS, 2011; SILVEIRA, 2019). Este processo é dividido em quatro módulos principais, conforme descrito a seguir.

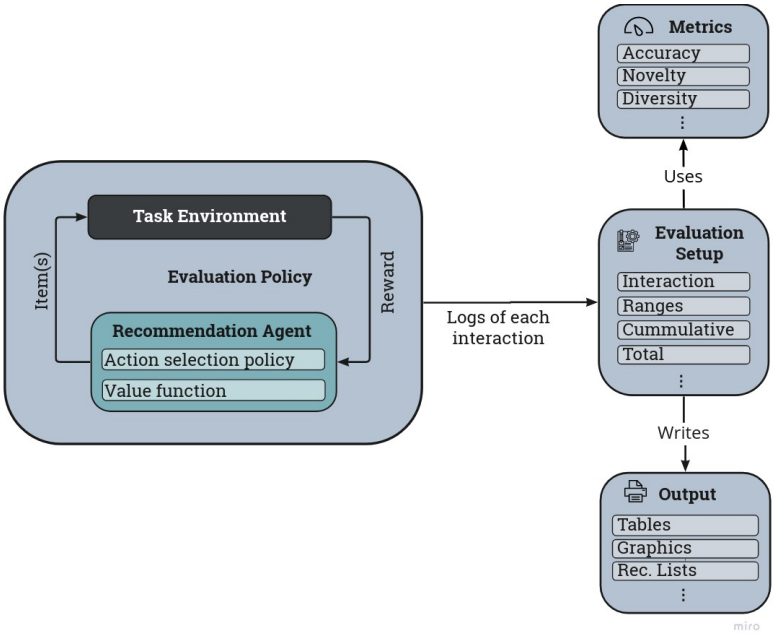

Figura 10 – Visão geral do componente *Experimental Evaluation*

Evaluation Policy. Este módulo é responsável por determinar como o *Recommendation Agent* irá interagir como o *Environment*. Basicamente, ele implementa o algoritmo clássico de Aprendizado por Reforço, onde o sistema:

(1) seleciona o usuário alvo;

(2) obtém a ação do modelo de recomendação;

(3) recebe o feedback do usuário para aquela ação específica;

(4) atualiza o conhecimento do modelo com o reward recebido.

Atualmente, o *iRec* contém duas principais estratégias já definidas: *Fixed Interactions* e *User-Driven Interactions*. Em ambos, cada usuário é selecionado aleatoriamente e cada ação não será realizada mais de uma vez para ele. Sua principal diferença está nos critérios de parada do cenário interativo. Na primeira, cada usuário será selecionado por T vezes. Assim, o sistema realizará iterações $T \times |U|$, onde $|U|$ é o número de usuários distintos disponíveis para a avaliação. O número T é pré-definido como parâmetro. No segundo, o sistema realizará novas ações até atingir todos os itens cadastrados no histórico do usuário. A ideia é fazer um experimento exaustivo para observar qual algoritmo leva mais tempo para atingir todos os itens previamente avaliados por cada usuário.

Evaluation Setup. Todas as ações realizadas pelo *Recommendation Agent* e *rewards* fornecidos pelos usuários de acordo com os dados no conjunto de teste são armazenados em *logs*. O módulo *Evaluation Setup* é que define como essas ações serão avaliadas. Na figura 10, vemos que o módulo *Evaluation Setup* recebe os logs gerados através da interação

do *agent* com o *environment* e utiliza o módulo *Metrics* para avaliar a qualidade dessas recomendações. Por fim, ele repassa os resultados obtidos para módulo de *Output*, para gerar as formas de visualizações. A seguir, apresentamos algumas estratégias previamente definidas:

- *Interaction:* avalia cada interação de cada usuário individualmente. Dado um cenário em que foram realizadas 100 interações, por exemplo, essa estratégia avalia cada uma separadamente.
- *Intervals:* primeiro agrega algumas interações consecutivas em um intervalo para, então, realizar as avaliações em cada grupo. Por exemplo, em uma execução de 10 interações, pode-se definir dois intervalos. Em seguida, ele criará dois grupos de 5 interações cada (um da 1ª à 5ª interação e outro da 6ª à 10ª interação) para realizar a avaliação.
- *Cumulative:* avalia as interações cumulativamente desde a primeira até um valor específico. Dessa forma, pode-se avaliar o resultado acumulado da 1ª à 10ª interação, depois, da 1ª à 15ª interação e assim sucessivamente.
- *Total:* avalia todo o processo de recomendação como um procedimento único. Por exemplo, se determinados itens foram recomendados durante 100 interações, a avaliação será feita apenas na 100ª interação.

Metrics: Esse módulo contém diversas métricas de avaliação adequadas ao cenário de recomendação. Elas são divididas em alguns grupos: *Accuracy* (SCHRÖDER, 2011; ZHOU, 2018), *Coverage* (GE, 2010), *Novelty* (VARGAS; CASTELLS, 2011) e *Diversity* (ZHAI, 2015). As métricas já implementadas são:

- **Precision** (RICCI, 2011): percentual de itens relevantes recomendados considerando o total de itens recomendados;
- **Recall** (RICCI, 2011): percentual de itens relevantes recomendados considerando todo o conjunto de itens relevantes;
- **Hits** (RICCI, 2011): número de recomendações que pertencem ao histórico de cada usuário;
- **EPC** (VARGAS; CASTELLS, 2011): é o número esperado de itens relevantes não vistos anteriormente pelo usuário (novidade);
- **ILD** (VARGAS; CASTELLS, 2011): é a correlação de Pearson do vetor de características dos itens na lista recomendada (diversidade);
- **EPD** (VARGAS; CASTELLS, 2011): mede a distância entre os itens do perfil do usuário e os recomendados;

- **Gini Coefficient** (SANZ-CRUZADO, 2019): é o inverso da frequência acumulada que cada item é recomendado;
- **Users Coverage** (SILVEIRA, 2019): porcentagem de usuários distintos que estão interessados nos itens recomendados.

Statistical Tests. O *iRec* possui um módulo específico para realizar testes estatísticos sobre as métricas de avaliação. A ideia é comparar os resultados obtidos por cada método e identificar qual deles supera estatisticamente os demais. O *iRec* implementa o teste de Wilcoxon (WOOLSON, 2007) para distribuições não paramétricas.

Output: Este é um módulo específico que oferece vários métodos para visualizar os resultados obtidos. O *iRec* oferece diferentes formas de visualizar os resultados através de: tabelas, nas quais o usuário pode selecionar diferentes métodos e métricas; e gráficos, nos quais os pesquisadores podem encontrar diferentes insights a partir dos dados. Se desejar, o pesquisador ainda pode acessar todo o log do experimento, que contém informações sobre as listas de recomendações, informando quantos e quais itens foram recomendados para determinado usuário.

5.4 Front-End

iRec possui um *front-end* de linha de comando chamado iRec-cmdline[2], que ilustra como preparar, executar e avaliar um experimento prático utilizando o *framework*. Sua estrutura é ilustrada na Figura 11. Em suma, o iRec-cmdline contém um repositório para: (1) settings, onde registra todas as configurações (i.e., parâmetros) exigidas por cada componente; e (2) scripts, onde vários scripts fornecem uma interface de linha de comando para acionar toda a estrutura. O settings é composto por vários arquivos *yaml*, que são responsáveis por configurar todos os parâmetros dos módulos do *framework*. Ao passo que scripts contém várias interfaces de linha de comando para iniciar as operações do *iRec*. O uso adequado desses *scripts* permite que se crie um *pipeline* de execução de acordo com uma configuração experimental. Eles estão relacionados a cada um dos três componentes principais do *iRec*.

Além disso, esses scripts são integrados ao *MLflow*, uma plataforma de código aberto que gerencia o ciclo de vida dos modelos de Aprendizado de Máquina, registrando seus logs de execução. A Figura 12 mostra uma visão geral dessa plataforma. Nela podemos ver os três principais componêntes do *iRec* sendo monitorados (*Recommendation Agent, Environment Setting* e *Experimental Evaluation*). O componenete selecionado, nessa figura, é o *Experimental Evaluation*, no qual vemos todas informações relacionados aos parâmetros dos modelos, os resultados das métricas utilizadas, dentre várias outras informações. Ademais, o *MLflow* fornece, além de todo esse monitoramento de logs de

[2] Disponível em: https://github.com/irec-org/irec-cmdline

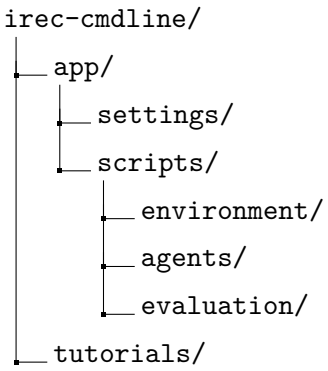

Figura 11 – iRec-cmdline: Estrutura de diretórios da aplicação

execuções, métodos de visualizações dos resultados que facilitam o entendimento e podem fornecer *insights* importantes sobre o comportamente de um modelo seguindo as configurações do experimento previamente definidas. A Figura 13, exibe um exemplo de visualização dos resultados, após a execução do *Grid-Search* no algoritmo $\epsilon-Greedy$. Nela podemos ver diferentes valores de ϵ, utilizados durante a tunagem de parâmetros, e seus respectivos resultados em relação a métrica *Hits*. Esse é um tipo de visualizacão interessante, pois podemos observar como a combinação de parâmetros afeta o desempenho final do modelo. Portando, isso significa que todos os *scripts* de todos os módulos do *framework* estão integrados a esta plataforma, facilitando o gerenciamento dos experimentos.

Por fim, a tabela 6 apresenta os principais *scripts* que tratam da execução de tarefas comuns para avaliação de sistemas de recomendação interativos, bem como os responsáveis pela configuração de hiperparâmetros, visualização de resultados, entre outros.

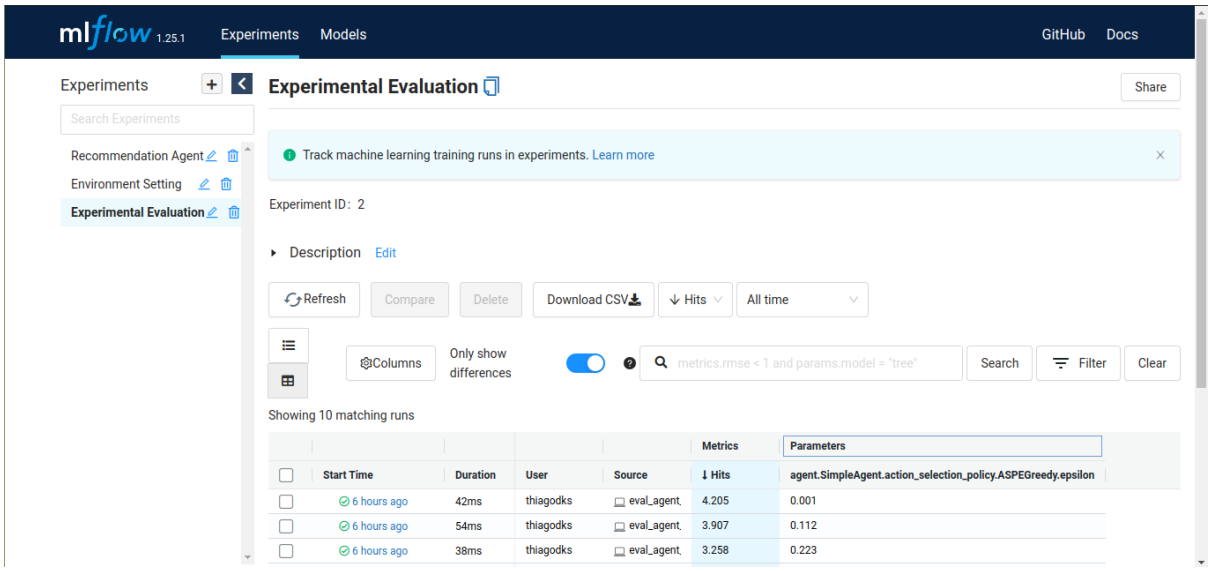

Figura 12 – Visão geral da plataforma *MLflow*, responsável por monitorar todas as execuções relacionadas aos três componentes principais do *iRec*.

Figura 13 – Visualização dos resultados obtidos pelo $\epsilon - Greedy$ após a tunagem de parâmetros. Podemos ver os diferentes valores de ϵ utilizados durante o *Grid-Search* e seus respectivos resultados relacionados à métrica *Hits*.

Script	Descrição	Parâmetros
download_data.py	Baixa os *datasets* disponíveis para o *iRec*.	−−dataset_name <dataset(s)>
generate_dataset.py	Realiza as etapas de leitura, *prefiltering* e *splitting*.	−−dataset_loaders <dataset(s)> −−agents <agent(s)>
run_agent_search.py	Faz a busca de parâmetros com base nos agentes e *datasets* especificados.	−−agents <agent(s)> −−dataset_loaders <dataset(s)> −−evaluation_policy <eval policie(s)> −−tasks <num tasks (opcional)>
eval_agent_search.py	Avalia os resultados da busca de parâmetros.	−−agents <agent(s)> −−dataset_loaders <dataset(s)> −−evaluation_policy <eval policie(s)> −−metric_evaluator <metric eval> −−metrics <metric(s)>
print_search.py	Exibe e atualiza os melhores parâmetros encontrados.	−−agents <agent(s)> −−dataset_loaders <dataset(s)> −−evaluation_policy <eval policie(s)> −−metric_evaluator <metric eval> −−metrics <metric(s)> −−t (somente os melhores parâmetros) −−d (salva os melhores parâmetros)
run_agent_best.py	Executa os agentes nos *datasets* utilizando os melhores parâmetros.	−−agents <agent(s)> −−dataset_loaders <dataset(s)> −−evaluation_policy <eval policie(s)> −−tasks <num tasks (opcional)>
eval_agent_best.py	Avalia os resultados obtidos após a execução dos agentes.	−−agents <agent(s)> −−dataset_loaders <dataset(s)> −−evaluation_policy <eval policie(s)> −−metric_evaluator <metric eval> −−metrics <metric(s)>
print_latex_table.py	Gera uma tabela com os resultados obtidos.	−−agents <agent(s)> −−dataset_loaders <dataset(s)> −−evaluation_policy <eval policie(s)> −−metric_evaluator <metric eval> −−metrics <metric(s)>

Tabela 6 – Os principais scripts do iRec que tratam das tarefas usuais para avaliação offline de sistemas de recomendação interativos. Todos os scripts também possuem a *flag* `--help` que permite ao usuário explorar todos os argumentos disponíveis.

5.5 Instanciando o iRec

Além da utilização do nosso *framework* via *scripts*, o iRec tambem pode ser instanciado como uma biblioteca, assim como qualquer outra biblioteca do *Python*, importando os módulos e executando cada etapa de um experimento. Sendo assim, nessa seção, vi-

samos demonstrar como realizar todo esse processo, através de um experimento simples, mas que utiliza todos os três componêntes do *iRec*, descritos nesse capítulo. Inicialmente, para execução desse experimento, é necessário realizar a importação dos módulos que serão utilizados. Como pode ser visto no arquivo de configuração5.1, importamos o método de *Loader FullData*, para fazer as manipulações necessárias na base de dados, em seguida, importamos os módulos para criação de um *Agent*, que neste exemplo é o $\epsilon - Greedy$ e, por fim, fazemos a importação dos módulos de avaliação.

Config. 5.1 – Importando os módulos necessários para realização do experimento

```
# Importando módulos do componente Environment Setting
from irec.environment.loader import FullData

# Importando os módulos do componente Recommendation Agent
from irec.recommendation.agents import SimpleAgent
from irec.recommendation.agents.action_selection_policies import ASPEGreedy
from irec.recommendation.agents.value_functions import EGreedy

# Importando os módulos do componente Experimental Evaluation
from irec.offline_experiments.evaluation_policies import FixedInteraction
from irec.offline_experiments.metric_evaluators import UserCumulativeInteraction
from irec.offline_experiments.metrics import Hits
```

O segundo passo do experimento é a configuração dos parâmetros do conjunto de dados, do *Agent* e da *Evaluation Policy*. No arquivo de configuração 5.2, definimos algumas informações sobre os dados e a estratégia de *splitting* que será utilizada. Logo após, criamos o *Agent*, definindo sua *Value Function* ($\epsilon - Greedy$) e sua *Action Selection Policy* (*ASPEGreedy*). Por fim, instanciamos a *Evaluation Policy* e a *Evaluation Setup*, que definirão o processo de interação do *Agent* com a base de dados (*Environment*), e o processo de avaliação dos resultados, respectivamente.

Config. 5.2 – Configurando os parâmetros

```
# Definindo os dados
dataset = {
    'path': "./app/data/datasets/MovieLens 100k/ratings.csv",
    'random_seed': 0,
    'file_delimiter': ",",
    'skip_head': True
}
# Divisão dos dados
splitting = {'strategy': "global", 'train_size': 0.8, 'test_consumes': 5}
# Instanciando o Loader
loader = FullData(dataset, splitting)
# Instanciando a Value Funcition
value_function = EGreedy()
# Instanciando a Action Selection Policy
greedy_selection = ASPEGreedy(epsilon=0.001)
# Criando o Agent
agent = SimpleAgent(value_function, greedy_selection, name="EGreedy")
```

```
# Instanciando a Evaluation Policy
eval_policy = FixedInteraction(num_interactions=100, interaction_size=1, save_info=True)
# Definindo a Evaluation Setup
evaluator = UserCumulativeInteraction(
    ground_truth_dataset=test_dataset,
    num_interactions=100,
    interaction_size=1,
    interactions_to_evaluate=[5, 10, 20, 50, 100],
    relevance_evaluator_threshold=3.99
)
```

Após esse processo de importação e configuração do experimento, nos resta apenas a execução e avaliação dos resultados. Para isso, no arquivo de configuração 5.3, iniciamos com o processamento dos dados, no qual é retornado os dados para treino e teste. Em seguida, a etapa de avaliação, passando o *Agent* e os dados de treino e teste para o método *evaluate*, que nos retorna todo o log de interação dos usuários com os itens que lhes foram recomendados. Finalmente, avaliamos esses resultados passando a métrica desejada, seguido das interações realizadas. O retorno da execução deste método são os valores de *Hits* obtidos nas iterações definidas previamente. Na próxima seção apresentamos as extensões, tal qual configurações, para utilizar o *iRec* na avaliação dos principais SsR baseados em modelos MAB no cenário de recomendação de entretenimento e de POIs, preenchendo, assim, essa lacuna da ciência em recomendação.

Config. 5.3 – Executando e avaliando o desempenho de um modelo MAB

```
import numpy as np

# Processando os dados
train_dataset, test_dataset, _, _ = loader.process()

# Executando o Agent
interactions, action_info = eval_policy.evaluate(agent, train_dataset, test_dataset)

# Obtendo os resultados
hits_values = evaluator.evaluate(metric_class=Hits, results=interactions)

print("\nResultados para métrica de Hits:")
print("top-5:", np.mean(list(hits_values[0].values())))
print("top-10:", np.mean(list(hits_values[1].values())))
print("top-20:", np.mean(list(hits_values[2].values())))
print("top-50:", np.mean(list(hits_values[3].values())))
print("top-100:", np.mean(list(hits_values[4].values())))

# Output
Resultados para a métrica de Hits:
top-5: 1.126984126984127
top-10: 2.1587301587301586
top-20: 4.01058201058201
top-50: 8.497354497354497
top-100: 14.613756613756614
```

5.6 Síntese do Capítulo

Neste capítulo, apresentamos o *iRec*, como resposta à **QP3: É possível prover um ambiente de execução reprodutível no cenário de recomendação interativa capaz de amenizar a lacuna presente no processo de avaliação das soluções de MAB?** O *iRec* é um *framework* completo para avaliação de SsR interativos. Com ele, visamos lidar com a falta de consenso sobre as melhores práticas de avaliação nesta área (SUN, 2020), fornecendo um ambiente completo para uma avaliação reprodutível e comparações justas de sistemas de recomendação. Podemos dizer que o *iRec* é composto três componentes principais que abrangem todo o processo de experimentação: (1) a construção de um
Environment; (2) a definição de *Recommendation Agent*; e (3) a definição de uma *Experimental Evaluation*. Dessa forma, neste capítulo, apresentamos em detalhes esses componentes, descrevendo todas as estratégias implementadas em cada um deles. Falamos também sobre o nosso *Front-End*, composto por diversos *scripts python*, que fornecem uma interface de linha de comando para acionar toda a estrutura do *iRec*. Através desses *scripts*, é possivel criar um *pipeline* de execução personalizado, de acordo com a configuração experimental. Além disso, exibimos os principais *scripts* disponíveis em nosso *Front-End* mediante a Tabela 6. Cada um deles está relacionado a cada um dos três componentes principais do *iRec* e são integrados ao *MLflow*. O *Mlflow* é uma plataforma que gerencia o ciclo de vida dos modelos de Aprendizado de Máquina, registrando seus logs de execução e facilitando o gerenciamento dos experimentos. Por fim, apresentamos uma forma alternativa de utilização do *iRec*, importando os módulos e executando cada etapa de um experimento. Durante esse processo, apresentamos como importar os módulos, como configurar os parâmetros e como executar e avaliar os resultados obtidos. Portanto, oferecendo essas duas formas de utilização do nosso *framework* (linhas de comando ou instanciando a biblioteca), deixamos o pesquisador livre para escolher a forma com que possui mais intimidade e facilidade de utilização. Uma vez que, independente de como utilizado, o *iRec* é extremamente intuitivo até para quem está começando com *Multi-Armed Bandits*.

6 Estudos de Caso

No capítulo anterior, apresentamos toda a estrutura do *iRec*, além das diversas estratégias implementadas nos três componentes principais (*Environment*, *Recommendation Agent*, e *Experimental Evaluation*) que abrangem todo o processo de experimentação de sistemas de recomendação interativos. Com o objetivo de demonstrar a amplitude que toda essa estrutura do *iRec* tem sobre as pesquisas atuais focadas em SsR interativos, respondendo assim, a quarta e última pergunta deste trabalho (**QP4: O quão abrangente um ambiente reprodutível pode ser a fim de suprir as necessidades dos diversos cenários em SsR interativos?**), construímos dois estudos de caso. Em cada um dos estudos, elaboramos um experimento completo intencionando exemplificar todo o processo de configuração, desde a entrada dos dados até avaliação e visualização dos resultados. O primeiro estudo é direcionado para o cenário de entretenimento, no qual selecionamos três conjuntos de dados: *Netflix* (Filmes), *Good Books* (Livros) e *Yahoo Music* (Música). Já o segundo é voltado para recomendação de POIs (Points-of-Interest). Para este estudo, selecionamos três coleções de dados reais provenientes da Yelp [1], relacionadas a três cidades americanas: Philadelphia, Nashville e New Orleans. Em ambos os estudos, selecionamos diversos algoritmos MAB disponíveis no *iRec* e avaliamos o comportamento de cada um deles, utilizando métricas de acurácia, novidade, diversidade e cobertura. Avaliando esses dois cenários mais globais, compostos de seis coleções de dados distintas, visamos comprovar toda a flexibilidade e eficiência do *iRec*, que independe do cenário em que é utilizado. Além da oportunidade de aplicação dos modelos de recomendação de forma sistematizada e simplificada, demonstramos que os pesquisadores de qualquer área podem usar o *iRec* para avançar em seus estudos e experimentos, garantindo o uso de melhores práticas em SsR.

6.1 Estudo 1: Recomendação no Cenário de Entretenimento

Para esse primeiro estudo, decidimos selecionar bases tradicionais do cenário de entretenimento. Como apresentado na Seção 3.2.1, podemos ver, através da Tabela 5, que este cenário engloba a grande maioria dos principais repositórios aplicados em propostas de MAB na área de recomendação interativa. Sendo assim, nesse estudo, decidimos abranger um cenário mais comum e mais utilizado nesses trabalhos. Para isso, avaliaremos o comportamento desses modelos MAB em diferentes categorias de entretenimento, e, consequentemente, observando qual o impacto que as características de cada domínio no comportamento desses modelos. Nesse sentido, na próxima seção, apresentaremos as

[1] https://www.yelp.com/dataset

características de cada conjunto de dados utilizados para esse primeiro experimento.

6.1.1 Bases de Dados

Para execução do primeiro estudo, selecionamos os três conjuntos de dados tradicionais: *Netflix*, *Good Books* e *Yahoo Music*, os detalhes de cada um deles estão descritos na Tabela 7. O conjunto de dados de filmes (*Netflix*) contém, originalmente, cem milhões de ratings, quase meio milhão de usuários e aproximadamente dezoito mil filmes. Todos esses dados foram extraídos do serviço de recomendação de filmes online *Netflix*. No entanto, para facilitar nossos experimentos exaustivos, selecionamos apenas dez mil usuários dessa base de dados. Os usuários foram selecionados aleatoriamente para a criação desse conjunto de dados. Para isso utilizamos o próprio módulo de pré-filtragem de dados presente no *iRec*. Para mais, todos os usuários selecionados avaliaram pelo menos 20 filmes. Os detalhes de como fizemos isso são descritos na próxima seção.

Assim como fizemos na base da *Netflix*, no conjunto de dados da *Yahoo Music R1* também precisamos aplicar algumas estratégias de pré-filtragem. Originalmente, este conjunto de dados possui onze milhões e meio de *ratings*, aproximadamente cem mil artistas e dois milhões de usuários. Portanto, também optamos por selecionar dez mil usuários aleatórios e remover aqueles que avaliaram menos de 20 músicas. Por último, mas não menos importante, o conjunto de dados do *GoodBooks*, criado com o intuito de resolver o problema da falta de dados relacionados a livros no cenário de recomendação. Esse conjunto de dados contém cerca de seis milhões de ratings para dez mil livros. Nesta base, aplicamos um único filtro também para remover os usuários com menos de 20 livros avaliados. Em todas as coleções de dados utilizadas, cada usuário e item é representado por um id e nenhuma outra informação é fornecida. Em todos eles as informações estão organizados no formato $user\_id, item\_id, rating, timestamp$, que é o mesmo formato de leitura padrão do *iRec*. Todos os *ratings* estão em uma escala de 1 a 5.

Coleções de Dados	**Domínio**	**# Usuários**	**# Itens**	**Esparsidade**
Netflix	Filmes	10,000	17,372	98.67%
GoodBooks	Livros	53,423	10,000	98.88%
Yahoo Music R1	Música	10,000	13,214	99.22%

Tabela 7 – Visão geral dos três conjuntos de dados do cenário de entretenimento aplicados no primeiro estudo.

6.1.2 Configurando iRec

De maneira geral, o *iRec* pode ser instanciado como uma biblioteca, estendendo os módulos e executando cada ação, ou mesmo executando linhas de comando, conforme mencionado no capítulo anterior. Para esse estudo, optamos por linhas de comando para

demonstrar como configurar cada arquivo *yaml* necessário no processo. Dessa forma, nas próximas subseções, detalharemos como configuramos cada componente para o cenário de entretenimento.

6.1.2.1 Componente 1: Environment

Conforme explicado na Seção 5.1, este componente é responsável por preparar as bases de dados para o processo experimental completo. No *irec-cmdline*, este componente é executado pelo script chamado *generate_dataset.py*. Ele executa as etapas de carregamento, pré-filtragem e divisão dos dados de acordo com os parâmetros configurados no arquivo *dataset_loaders.yaml*. A Configuração 6.1 demonstra como configuramos essas etapas considerando os conjunto de dados da *Netflix*, *Good Books* e *Yahoo Music*. Através desse arquivo, vemos que o nome do conjunto de dados é o primeiro parâmetro a ser inserido. Depois disso, o segundo parâmetro a ser definido é qual método de *loader* será utilizado para carregar a base de dados. Atualmente, *iRec* tem dois métodos implementados, conforme mencionado na Seção 5.1. Para todos os conjunto de dados utilizados nesse primeiro estudo, definimos o *FullData* para realizar a divisão dos dados com as abordagens implementadas no *iRec*.

Config. 6.1 – dataset_loaders.yaml

```yaml
Netflix 10k:
  FullData:
    dataset:
      path: .../Netflix/ratings.csv
      random_seed: 0
      file_delimiter: ","
      skip_head: true

    prefiltering:
      filter_users:
        min_consumption: 20
        num_users: 10000
      # filter_items:
        # min_ratings: x
        # num_items: x
    splitting:
      strategy: global
      train_size: 0.8
      test_consumes: 5

    validation:
      validation_size: 0.2
```

```yaml
Good Books:
  FullData:
    dataset:
      path: .../Good Books/ratings.csv
      random_seed: 0
      file_delimiter: ","
      skip_head: true

    prefiltering:
      filter_users:
        min_consumption: 20
        # num_users: x
      # filter_items:
        # min_ratings: x
        # num_items: x
    splitting:
      strategy: global
      train_size: 0.8
      test_consumes: 5

    validation:
      validation_size: 0.2
```

```yaml
Yahoo Music 10k:
  FullData:
    dataset:
      path: .../Yahoo Music/ratings.csv
      random_seed: 0
      file_delimiter: ","
      skip_head: true

    prefiltering:
      filter_users:
        min_consumption: 20
        num_users: 10000
      # filter_items:
        # min_ratings: x
        # num_items: x
    splitting:
      strategy: global
      train_size: 0.8
      test_consumes: 5

    validation:
      validation_size: 0.2
```

O método *FullData* requer que o pesquisador defina informações sobre o conjunto de dados, o *prefiltering* (se necessário) e a abordagem de *splitting*. Inicialmente, devemos definir o *path* onde os dados estão localizados, o $file\_delimiter$ que divide cada linha do *dataset* e se existe algum *head* a ser ignorado. Por padrão, *iRec* lerá os conjuntos de dados

disponíveis em *app/data/datasets/*. Em seguida, o pesquisador deve definir os parâmetros para o *prefiltering*. Em nosso exemplo, aplicamos os mesmos filtros para os conjuntos de dados da *Netflix* e *Yahoo Music*: (1) *min_consumption: 20* que elimina usuários com menos de vinte itens consumidos; e (2) *num_users: 10000* que seleciona dez mil usuários aleatórios da base original. Já para o conjunto da *Good Books*, aplicamos apenas o filtro *min_consumption: 20*. Se o pesquisador quiser evitar esta etapa, basta remover todas as chaves relacionadas às etapas de *prefiltering*. Por fim, deve ser definida a estratégia de *splitting* que será utilizada configurando a opção *strategy*. Usamos a estratégia *global* neste arquivo de configuração, com 80% dos dados compondo o treinamento e o restante para o teste. Além disso, neste exemplo, também adicionamos a configuração opcional para o conjunto de validação. Configurando esta opção e definindo o *size* do conjunto de validação, o *iRec* fará o ajuste do hiperparâmetro nos próximos passos.

6.1.2.2 Componente 2: Recommendation Agent

Como mencionado na Seção 5.2, todo sistema de recomendação é um *agent* com uma *action selection policy* e uma *value function*. Enquanto a *value function* determina a importância de cada item para um usuário, uma previsão, a *selection policy* determina quais critérios devem ser seguidos para selecionar uma ação, isto é, fazer uma recomendação. Assim, no irec-cmdline, o pesquisador deve seguir um template para definir o sistema de recomendação no arquivo *dataset_agents.yaml*, conforme ilustrado no arquivo de configuração 6.3. Para cada conjunto de dados, o pesquisador deve definir agentes distintos a serem executados pelo nosso *framework*. Em nosso exemplo, apresentamos três dos 17 modelos MAB selecionados para nossa avaliação. Tratam-se de modelos clássicos para serem avaliados no conjunto de dados *Netflix*: *UCB*, ϵ-*Greedy* e *TS*. Cada agente é configurado com uma tag relacionada à implementação do agente que o algoritmo deve seguir.

Neste exemplo, definimos a tag como *SimpleAgent*, o que significa que devemos definir apenas uma *action selection policy* e uma *value function* para cada uma delas. *iRec* também permite agents baseados em estratégias de ensemble (TANG, 2014) usando a tag *EnsembleAgent* e, neste caso, mais de uma *action selection policy* e/ou mais de uma *value function* pode ser definida . Em seguida, devemos fornecer a *action selection policy* e a *value function* que cada agente utilizará. Conforme adotado por BEARS (BARRAZA-URBINA, 2018), a *action selection policy* e a *value function* são implementadas separadamente em nosso *framework*. Assim, após definir o tipo de agent, devemos informá-los em sequência. Em termos de uma *action selection policy*, o ϵ-Greedy tradicionalmente usa a política *egreedy*, que seleciona os melhores itens com base no parâmetro de diversificação ϵ para executar recomendações aleatórias (AUER, 2002). Por sua vez, UCB e TS usam outra política, chamada *greedy*, que seleciona os melhores itens naquele momento (AUER, 2002; CHAPELLE; LI, 2011). Em relação à *value function*, cada agente possui uma *value function* que se refere a si mesmo. Se o pesquisador quiser estender essas *value functions* com

novos métodos, é altamente recomendável chamá-las pelo mesmo nome do novo método.

Config. 6.2 – agents_variables.yaml

```
GridSearch:
  EGreedy:
    SimpleAgent:
      action_selection_policy:
        ASPEGreedy:
          epsilon: linspace(0.1, 1, 5)
      value_function:
        EGreedy: {}
  UCB:
    SimpleAgent:
      action_selection_policy:
        ASPGreedy: {}
      value_function:
        UCB:
          c: [0.1, 1, 5]
  ThompsonSampling:
    SimpleAgent:
      action_selection_policy:
        ASPGreedy: {}
      value_function:
        ThompsonSampling:
          alpha_0: linspace(0.1, 1, 5)
          beta_0: linspace(1, 100, 10)
```

Config. 6.3 – dataset_agents.yaml

```
Netflix:
  EGreedy:
    SimpleAgent:
      action_selection_policy:
        ASPEGreedy:
          epsilon: 0.1
      value_function:
        EGreedy: {}
  UCB:
    SimpleAgent:
      action_selection_policy:
        ASPGreedy: {}
      value_function:
        UCB:
          c: 0.1
  ThompsonSampling:
    SimpleAgent:
      action_selection_policy:
        ASPGreedy: {}
      value_function:
        ThompsonSampling:
          alpha_0: 1
          beta_0: 100
```

6.1.2.3 Componente 3: Experimental Evaluation

Para configurar o componente *Experimental Evaluation*, primeiramente definimos a *Evaluation Policy* por meio do arquivo *evaluation_policies.yaml*. No exemplo apresentado no arquivo de configuração 6.4, selecionamos a política *Fixed Interactions* e definimos seus três parâmetros: 1 - o número de interações; 2 - o tamanho da lista de itens recomendados em cada interação; e 3 - uma variável que informa se deve armazenar (ou não) todo o *log* do processo. Outras políticas podem ser aplicadas – deixamos comentada no arquivo de configuração uma outra possibilidade. A execução da avaliação experimental é realizada por um script *run_agent_best.py*, que é executado de acordo com os parâmetros configurados nos arquivos *dataset_loaders.yaml*, *dataset_agents.yaml* e *evaluation_policies.yaml*. Os resultados de todas as interações são registrados em arquivos de *log* para uma avaliação posteriormente.

A definição de como avaliar essas interações é realizada pelo script *eval_agent_best.py*. Além dos arquivos *dataset_loaders.yaml*, *dataset_agents.yaml* e *evaluation_policies.yaml*, este script também usa os parâmetros configurados no *metric_evaluators.yaml*, que define como o pesquisador deseja avaliar as ações realizadas pelos agentes de recomendação após todas as interações. A lista de métricas a serem consideradas é informada ao script como parâmetro – consulte a Tabela 6. O irec-cmdline fornece diferentes *evaluation se-*

tups. A Configuração 6.5 apresenta um exemplo, considerando uma estratégia cumulativa definida pela tag *Interaction*. Para essa estratégia, o pesquisador deve estabelecer quatro parâmetros principais:

1. o número de itens a serem selecionados por interação (por exemplo, 1);
2. o intervalo de interações para medir as métricas de avaliação (por exemplo, nas interações número 10, 50 e 100);
3. o número total de interações (por exemplo, 100);
4. o threshold para que um item seja considerado relevante para um usuário – um threshold importante para as métricas de avaliação, como Recall, EPD e EPC.

Config. 6.4 – evaluation_policies.yaml

```yaml
FixedInteraction:
  num_interactions: 100
  interaction_size: 1
  save_info: True
#UserDrivenInteractions:
  #interaction_size: 1
  #recommend_test_data_rate_limit: 0.1
```

Config. 6.5 – metric_evaluators.yaml

```yaml
Interaction:
  interaction_size: 1
  interactions_to_evaluate: [10, 50, 100]
  num_interactions: 100
  relevance_evaluator_threshold: 3.999
```

As outras opções estão comentadas no exemplo ilustrado em Configuration 6.5. Todos eles são bastante afins em termos de configuração, mas podem realizar análises distintas para o pesquisador. Após definir os arquivos de configuração e executar os scripts de operação, podemos escolher como visualizar os resultados finais. Atualmente, irec-cmdline oferece recursos (configurados como arquivos do tipo scripts) que geram automaticamente diferentes formas de visualização em gráficos e tabelas. O script *print_latex_table.py* fornece uma tabela para visualizar os resultados – veja a Tabela 8. Como o nome indica, este arquivo gera automaticamente uma tabela com todos os modelos, datasets e métricas informados pela execução de parâmetros. Além disso, ele também realiza os testes estatísticos implementados no *iRec* (i.e., teste de Wilcoxon) sobre os resultados obtidos, destacando os ganhos e perdas significativas na tabela impressa.

Além disso, conforme mencionado na Seção 3.1, *iRec* fornece um módulo opcional para realizar um ajuste de hiperparâmetros. O pesquisador deve selecionar o método de tunagem e definir a faixa de parâmetro(s) do(s) agente(s). No irec-cmdline, isso pode ser feito definindo essas informações no arquivo *agents_variables.yaml*, conforme ilustrado em Configuration 6.2. O primeiro nome deve se referir à abordagem de ajuste selecionada (por exemplo, o Grid-Search). Então, o arquivo de configuração restante é bastante semelhante ao que representa os modelos de agente ilustrados em Configuration 6.3. No entanto, em vez de passar apenas um parâmetro, o pesquisador deve definir uma lista de parâmetros a

serem explorados. Este exemplo apresenta a configuração para os algoritmos bandidos clássicos de ϵ-Greedy ($\epsilon \in [0.1, 1]$), UCB ($c \in [0.1, 1]$) e TS ($alpha \in [0.1, 1]$; $beta \in [1, 100]$). Em seguida, executando os scripts *run_agent_search.py* e *eval_agent_search.py* oferecido pelo irec-cmdline, iniciará o processo de recomendação do agent com a evaluation policy previamente selecionada para pesquisar entre os parâmetros e avaliar todas as execuções. Ambos os scripts consideram os arquivos *dataset_loaders.yaml*, *evaluation_policies.yaml* e *metric_evaluators.yaml* previamente definidos. Ao final, o pesquisador também pode executar o script *print_search.py* para visualizar os melhores parâmetros encontrados e atualizar automaticamente o arquivo *dataset_agents.yaml* com eles. Assim, nas próximas execuções, o irec-cmdline executará todos os algoritmos com o melhor parâmetro identificado.

6.1.3 Resultados e Discussões

A Tabela 8 mostra os resultados experimentais gerados pela configuração de *iRec* com arquivos de configuração descritos anteriormente nesta seção. Essa tabela foi gerada automaticamente pelo nosso *framework*, no qual podemos analisar o comportamento de diferentes modelos de recomendação nos diferentes cenários selecionados. Nesta avaliação experimental, aplicamos o teste estatístico *Wilcoxon* (computado em pares de modelos com sua melhor configuração) e utilizamos métricas de accuracy, coverage, diversity, e novelty, apresentando diferentes formas de avaliar o desempenho de um modelo. Além dos modelos MAB clássicos (ϵ-Greedy, UCB e Thompson Sampling), executamos outros modelos MAB mais recentes para enriquecer a análise dos dados obtidos.

Analisando os resultados, é importante observar que nenhum deles obteve os melhores resultados para todas as métricas avaliadas, demonstrando a importância de utilizar diferentes métricas de avaliação adequadas ao cenário de recomendação. Os SsR devem ser capazes de fornecer itens relevantes, diversos e novos, suprindo as necessidades de consumo da maioria dos usuários de um sistema de recomendação. Outro ponto a destacar é a importância dos testes estatísticos para validar a comparação dos resultados. Por exemplo, analisando a métrica *Hits* na 10^{a} interação, no conjunto de dados da *Yahoo Music*, vemos que, apesar da superioridade do NICF, o modelo do PTS empata estatisticamente com ele. Dessa forma, esses resultados mostram o quão importante é definir bem o protocolo de avaliação e analisar o comportamento dos modelos em diferentes etapas. Podemos ver que nenhum modelo foi superior em todas as interações. O CLinUCB, apesar de perder nas primeiras 10 interações, em *Hits* e *Recall* em todas as bases, observa-se que a médio prazo (50 interações) e longo prazo (100 interações), este mesmo modelo se torna extremamente superior aos demais. Conforme apresentado na Seção 1.1, o número de tentativas a serem realizadas, o número de itens a serem recomendados em cada tentativa e se o usuário pode receber as mesmas recomendações nas próximas tentativas, entre muitos outros parâmetros de avaliação, impactam diretamente nos resultados finais.

Dataset	Yahoo Music			Netflix			Good Books		
Métrica	Hits			Hits			Hits		
T	10	50	100	10	50	100	10	50	100
CLinUCB	2.336	**17.922**▲	**27.260**▲	1.461	**15.184**▲	**25.947**▲	1.776	**11.759**▲	**18.623**▲
UCB	1.358	7.330	13.277	1.284	5.440	9.915	0.764	2.984	5.493
e-Greedy	1.460	7.424	13.360	1.320	5.390	9.936	0.800	3.072	5.633
TS	1.907	8.356	14.720	1.882	7.498	12.959	1.361	4.528	7.216
Linear UCB	3.157	15.514	25.361	1.980	12.076	22.361	1.586	6.848	12.593
Linear e-Greedy	0.011	0.316	1.059	0.158	2.303	6.037	0.060	0.815	2.543
Cluster Bandit	2.428	10.095	16.777	2.082	8.649	16.265	2.187	7.863	12.173
NICF	**3.345**●	11.457	14.951	**2.837**▲	10.268	15.690	**2.960**▲	7.785	10.452
PTS	**3.329**●	14.363	19.634	0.442	3.635	8.901	1.687	7.386	12.951
ICTRTS	0.138	6.149	13.446	0.046	2.865	6.205	1.159	7.013	11.293
Métrica	Recall			Recall			Recall		
T	10	50	100	10	50	100	10	50	100
CLinUCB	0.059	**0.455**▲	**0.648**▲	0.022	**0.283**▲	**0.442**▲	0.024	**0.154**▲	**0.241**▲
UCB	0.034	0.179	0.321	0.017	0.075	0.141	0.010	0.039	0.072
e-Greedy	0.036	0.183	0.323	0.017	0.074	0.141	0.011	0.041	0.074
TS	0.048	0.204	0.354	0.025	0.101	0.177	0.018	0.060	0.095
Linear UCB	0.086	0.388	0.597	0.031	0.208	0.360	0.021	0.089	0.163
Linear e-Greedy	0.000	0.006	0.018	0.001	0.014	0.042	0.001	0.011	0.034
Cluster Bandit	0.064	0.254	0.413	0.043	0.147	0.273	0.030	0.105	0.161
NICF	**0.086**●	0.280	0.352	**0.051**▲	0.168	0.247	**0.040**▲	0.103	0.137
PTS	**0.088**●	0.359	0.473	0.007	0.061	0.155	0.023	0.098	0.170
ICTRTS	0.003	0.153	0.327	0.001	0.045	0.092	0.015	0.093	0.149
Métrica	ILD			ILD			ILD		
T	10	50	100	10	50	100	10	50	100
CLinUCB	0.462	0.423	0.435	0.399	0.372	0.378	0.472	0.453	0.464
UCB	0.465	0.463	0.465	0.404	0.416	0.421	**0.490**▲	**0.495**▲	**0.495**▲
e-Greedy	0.461	0.462	0.465	0.401	0.416	0.421	0.489	0.494	0.495
TS	0.431	0.452	0.459	0.336	0.373	0.387	0.467	0.487	0.492
Linear UCB	0.387	0.418	0.436	0.375	0.387	0.394	0.428	0.469	0.476
Linear e-Greedy	0.466	**0.478**▲	**0.488**▲	0.481	**0.482**▲	**0.481**▲	0.487	0.488	0.488
Cluster Bandit	0.437	0.448	0.455	0.429	0.392	0.391	0.384	0.451	0.468
NICF	0.405	0.433	0.464	0.339	0.368	0.394	0.392	0.461	0.479
PTS	0.406	0.434	0.461	0.475	0.464	0.462	0.465	0.471	0.477
ICTRTS	**0.490**▲	0.475	0.467	**0.493**▲	0.460	0.446	0.480	0.465	0.474
Métrica	UsersCoverage			UsersCoverage			UsersCoverage		
T	10	50	100	10	50	100	10	50	100
CLinUCB	0.850	**0.983**▲	**0.992**▲	0.690	**0.967**▲	**0.986**●	**0.764**▲	**0.934**▲	**0.968**▲
UCB	0.664	0.960	0.986	0.544	0.872	0.949	0.483	0.842	0.930
e-Greedy	0.684	0.960	0.985	0.545	0.857	0.954	0.499	0.853	0.932
TS	0.748	0.960	0.986	0.631	0.895	0.957	0.636	0.893	0.944
Linear UCB	0.806	0.967	0.990	0.588	0.920	0.963	0.489	0.819	0.915
Linear e-Greedy	0.005	0.038	0.075	0.078	0.129	0.244	0.037	0.181	0.306
Cluster Bandit	0.814	0.965	0.986	0.844	0.963	**0.986**●	0.673	0.926	0.964
NICF	0.839	0.956	0.967	**0.845**▲	0.959	0.977	0.747	0.920	0.957
PTS	**0.858**▲	0.977	0.989	0.320	0.866	0.952	0.673	0.908	0.961
ICTRTS	0.117	0.954	0.982	0.045	0.806	0.928	0.608	0.922	0.962

Tabela 8 – Performance dos modelos *bandit* no cenário de entretenimento. Os resultados foram comparados com o teste de Wilcoxon com p-value = 0.05. O símbolo ▲ denota ganhos estatísticos e o símbolo ● representa empates.

Além disso, Métricas de diversidade, novidade e cobertura também devem ser consideradas ao avaliar um sistema de recomendação. Se tratando de MAB, recomendar itens mais diversos, está relacionado ao conceito de *exploration*, ou seja, o sistema está optando por recomendar um item mais distinto capaz de agregar mais conhecimento ao sistema. Por outro lado, ao recomendar itens mais populares, por exemplo, o sistema está em busca do item potencialmente mais relevante até o momento para satisfazer ao usuário imediatamente. No entanto, com base no próprio consenso existente na literatura, sabemos que explorar apenas *exploration* ou *explotation* pode trazer prejuízos ao sistema. Um sistema que exige um esforço inicial ou que apresenta itens pouco relevantes para os usuário pode

se comprometer facilmente. Em contrapartida, apresentar itens potencialmente relevantes, explorando o conhecimento existente no sistema, como por exemplo, a popularidade dos itens, adiciona pouco conhecimento sobre o usuário, uma vez que todo mundo tende a se interessar por tais itens (ou eles não seriam os populares). Logo, estamos atrás de um modelo que consiga equilibrar esses dois conceitos. Sendo assim, se analisarmos a tabela 8, apesar de não haver nenhum algoritmo capaz de demonstrar superioridade em todas métricas e interações apresentadas, vemos que o CLinUCB é o que possui melhor desempenho em todas as bases de dados. Além dos melhores resultados em *Hits* e *Recall*, este modelo consegue cobrir uma grande porcentagem de usuários distintos que estão interessados nos itens a eles recomendados (*Users Coverage*).

Portanto, em um cenário real, para dizer que um modelo é melhor que outro, não basta avaliar apenas uma métrica e os resultados a curto, médio ou longo prazo. Um modelo de recomendação deve permanecer constante, apresentando bons resultados em diferentes etapas e demonstrar superioridade ou ao menos competitividade em relação a diferentes métricas de avaliação. Ademais, o *framework iRec* também permite aos pesquisadores melhorar a comparação dos modelos e garantir a reprodutibilidade de todos os experimentos. O processo de ajuste dos hiperparâmetros para todos os modelos de recomendação permite aos pesquisadores obter os melhores resultados para cada modelo e fornecer uma comparação justa entre as linhas de base (DACREMA, 2021). Desconsiderando o processo de ajuste e definindo arbitrariamente os parâmetros *alpha_0* para 1 e *beta_0* para 10, por exemplo, o TS marcou 1,572, 6,943 e 11,291, na base da *Netflix*, em termos de Hits nas interações 10^{o}, 50^{o} , e 100^{o}, respectivamente. Como esperado, após o ajuste (configuração *alpha_0*: 1 *beta_0*: 100), o algoritmo TS obteve os melhores resultados para essas interações, conforme mostrado em 8. Além de tudo, como o *iRec* armazena todas as sementes aleatórias desde a preparação dos dados até o processo de avaliação, além dos melhores parâmetros obtidos após o ajuste dos hiperparâmetros, podemos garantir que os resultados sejam os mesmos independentemente do computador. Esses experimentos, por exemplo, foram executados em cinco computadores diferentes com diferentes configurações de hardware e obtivemos os mesmos resultados para os métodos determinísticos.

6.2 Estudo 2: Recomendação no Cenário de POI

Para nosso segundo estudo, selecionamos três coleções de dados reais provenientes da Yelp, relacionadas a três cidades americanas: *Philadelphia*, *Nashville* e *New Orleans*. A aplicação de recomendadores interativos no cenário de recomendação de POIs é recente e pouco estudada na literatura, sendo que há poucos trabalhos em recomendações de POIs (Point-of-Interest) interativas (OMIDVAR-TEHRANI, 2020; YU, 2020; WANG, 2022). Embora existam diversas abordagens eficazes de recomendação nos cenários tradicionais, elas não são igualmente efetivas para o cenário de recomendação de POIs. Dentro do

domínio de recomendação, o cenário de POIs se difere dos demais devido às suas características peculiares. Primeiro, muitos trabalhos destacam que o problema de esparsidade é ainda maior no domínio de POIs, haja visto que o usuário enfrenta obstáculos físicos para visitar lugares (LIU, 2014; YE, 2011). Para mais, outros trabalhos destacam que a fonte de informação sobre os usuários difere dos demais. Contudo, nenhum deles adota modelos de Multi-Armed Bandits (MAB), considerados estado-da-arte em outros cenários de SsR. Por essa razão, esta seção descreve a aplicação do *framework iRec*, para modelar e avaliar distintas abordagens MAB para o cenário de POIs. Apresentamos os conjuntos de dados considerados, detalhamos as configurações realizadas e avaliamos os resultados alcançados por cada modelo.

6.2.1 Bases de Dados

Como mencionado anteriormente, nesse segundo estudo, selecionamos o conjunto de dados Yelp [2]. É um conjunto de dados disponível publicamente de um LBSN do mundo real. O conjunto de dados do Yelp contém quatro tipos de informações: informações comerciais, de review, de usuário e de check-in. O dataset completo compreende um total sete milhões avaliações de usuários com informações sobre cento e cinquenta mil empresas em onze áreas metropolitanas. No entanto, neste estudo, consideramos apenas o arquivo de review, composto por *user_id*, *business_id*, *rating*, *timestamp*, que é o mesmo formato utilizado nas conjuntos de dados de entretenimento. Todos os ratings também estão em uma escala de 1 a 5. Além disso, para execução de nossos experimentos, selecionamos três cidades disponíveis na base de dados da Yelp, sendo elas: (1) Philadelphia (2) Nashville e (3) New Orleans. Para todas as bases de dados, aplicamos um filtro no consumo de cada usuário, utilizando o módulo de prefiltering do próprio *iRec*, o qual requer que todos usuários tenham pelo menos 10 itens consumidos. A seguir, na Tabela 9, podemos ter uma visão geral das bases de dados selecionadas.

Coleções de Dados	**# Usuários**	**# Itens**	**# Esparsidade**
Philadelphia	15.913	14.024	99,80%
Nashville	5.772	6.437	99,63%
New Orleans	8.202	5.948	99,62%

Tabela 9 – Visão geral dos três conjuntos de dados aplicados no segundo estudo.

6.2.2 Configurando iRec

Da mesma forma como apresentado no estudo 1, optamos por linhas de comando para demonstrar como configurar cada arquivo *yaml* necessário no processo. Dessa forma,

[2] https://www.yelp.com/dataset

análogas a seção de configuração apresentada anteriormente, nas próximas subseções, detalhamos como configuramos cada componente para o cenário de POIs. No entanto, a seguir, apresentaremos apenas as configurações que diferem das apresentadas no estudo anterior.

6.2.2.1 Componente 1: Environment

De maneira muito semelhante parecida na Seção 6.1.2.1, neste componente configuramos os parâmetros para preparar as bases de dados para o processo experimental. Logo, a única diferença dessa seção para a anterior está relacionada aos parâmetros que escolhemos para o cenário de POIs. Sendo assim, através do arquivo de configuração 6.6, vemos que para todos os conjunto de dados descritos foi aplicado um único filtro: *min_consumption: 10*, que eliminará os usuários com menos de 10 avaliações. Em seguida, a próxima configuração que difere do cenário de entretenimento está relacionada à estratégia de *splitting* utilizada. Para esse parâmetro selecionamos a estratégia denominada *user_history*, ideal para o cenário de POIs, descrita no Capítulo 5.

Config. 6.6 – dataset_loaders.yaml

```yaml
Philadelphia:
  FullData:
    dataset:
      path: .../Philadelphia/ratings.csv
      random_seed: 0
      file_delimiter: ","
      skip_head: true

    prefiltering:
      filter_users:
        min_consumption: 10
        # num_users: x
      # filter_items:
        # min_ratings: x
        # num_items: x
    splitting:
      strategy: user_history
      train_size: 0.8
      test_consumes: 5

    validation:
      validation_size: 0.2
```

```yaml
Nashville:
  FullData:
    dataset:
      path: .../Nashville/ratings.csv
      random_seed: 0
      file_delimiter: ","
      skip_head: true

    prefiltering:
      filter_users:
        min_consumption: 10
        # num_users: x
      # filter_items:
        # min_ratings: x
        # num_items: x
    splitting:
      strategy: user_history
      train_size: 0.8
      test_consumes: 5

    validation:
      validation_size: 0.2
```

```yaml
New Orleans:
  FullData:
    dataset:
      path: .../New Orleans/ratings.csv
      random_seed: 0
      file_delimiter: ","
      skip_head: true

    prefiltering:
      filter_users:
        min_consumption: 10
        # num_users: x
      # filter_items:
        # min_ratings: x
        # num_items: x
    splitting:
      strategy: user_history
      train_size: 0.8
      test_consumes: 5

    validation:
      validation_size: 0.2
```

6.2.2.2 Componente 2: Recommendation Agent

Como esse componente é praticamente independente do conjunto de dados utilizados, as configurações definidas para este segundo estudo são basicamente as mesmas. Como mencionado na Seção 6.1.2.2, para cada conjunto de dados, o pesquisador deve definir agentes distintos a serem executados pelo nosso framework. Logo, para este es-

tudo, simplesmente adicionamos uma chave (nome do *dataset*) em *dataset_agents.yaml* para cada uma dos conjuntos de dados desse experimento: *Philadelphia*, *Nashville* e *New Orleans*. Por fim, inserimos os mesmo templates de agentes MAB em todas elas.

6.2.2.3 Componente 3: Experimental Evaluation

Para esse componente, selecionamos os mesmos métodos de avaliação utilizados durante o processo de configuração para cenário de entretenimento. Entretanto, conforme podemos ver nos Arquivos 6.7 e 6.8, os parâmetros utilizados são diferentes. Estes parâmetros foram ajustados especialmente para o cenário de POIs. Na Seção 6.1.2.3, foi definido o número de interações como 100 (*num_interactions: 100*), porém esse número de interações no cenário de POIs, é demasiadamente alto, uma vez que os usuários possuem uma média de avaliações extremamente baixa. Dessa forma, decidimos diminuir o número de interações realizadas nesse experimento para 20. Já o tamanho das listas de recomendações fornecidas (*interaction_size*) mantivemos como 1. Além disso, seguimos o mesmo processo de configuração e execução utilizado na tunagem de parâmetros descrito no último parâgrafo da Seção 6.1.2.3.

Config. 6.7 – evaluation_policies.yaml

```yaml
FixedInteraction:
  num_interactions: 20
  interaction_size: 1
  save_info: True
  #save_info: False
```

Config. 6.8 – metric_evaluators.yaml

```yaml
Interaction:
  interaction_size: 1
  interactions_to_evaluate: [5, 10, 20]
  num_interactions: 20
  relevance_evaluator_threshold: 3.999
```

6.2.3 Resultados e Discussões

Apresentamos os resultados obtidos pelos algoritmos interativos na Tabela 10. Assim como no cenário descrito no Estudo 1, os algoritmos passaram pelo mesmo processo de ajuste de hiperparâmetros, onde tentamos maximizar o nível de *hits* no conjunto de validação (20% do conjunto de treino). Para a avaliação dos resultados utilizamos as mesmas métricas de acurácia, cobertura e diversidade, dessa vez considerando intervalos de 5, 10 e 20 interações acumuladas. Para todos os algoritmos, aplicamos o teste estatístico *Wilcoxon* para validar se os ganhos encontrados são significantes. Como pode-se notar, com os resultados de todos esses algoritmos *bandits*, conseguimos realizar uma avaliação detalhada do uso de modelos MAB no cenário de recomendação interativa de POIs, o que é uma novidade deste trabalho em relação ao que existia até aqui na literatura.

Devido à completude do *iRec*, podemos analisar o desempenho de diversas estratégias interativas sob diferentes perspectivas, tais como acurácia e diversidade. Em geral, todos os algoritmos de recomendação em POIs sofrem com a esparsidade do conjunto de dados e apresenta baixos níveis de acurácia. Podemos notar que algoritmos clássicos de Aprendizado por Reforço adaptados ao cenário de recomendação (p.ex., ϵ-Greedy, UCB,

Dataset	Philadelphia			New Orleans			Nashville		
Métrica	Hits			Hits			Hits		
T	5	10	20	5	10	20	5	10	20
CLinUCB	**0.057**▲	**0.107**▲	**0.184**▲	**0.081**●	**0.141**●	0.234	**0.072**●	**0.123**▲	**0.219**▲
UCB	0.013	0.025	0.044	0.020	0.036	0.074	0.016	0.032	0.063
e-Greedy	0.011	0.021	0.042	0.018	0.039	0.075	0.017	0.034	0.065
TS	0.023	0.041	0.068	0.007	0.019	0.044	0.033	0.059	0.100
Linear UCB	0.039	0.084	0.144	**0.076**●	**0.136**●	0.218	**0.069**●	0.109	0.169
Linear e-Greedy	0.002	0.004	0.007	0.001	0.003	0.006	0.003	0.005	0.011
Cluster Bandit	0.040	0.086	0.163	0.061	0.128	0.232	0.038	0.084	0.160
NICF	0.027	0.070	0.084	0.058	0.103	0.147	0.037	0.079	0.107
PTS	0.044	0.081	0.143	0.060	0.124	**0.250**▲	0.055	0.109	0.202
ICTRTS	0.039	0.087	0.149	0.037	0.096	0.188	0.020	0.061	0.129
Métrica	Recall			Recall			Recall		
T	5	10	20	5	10	20	5	10	20
CLinUCB	**0.004**▲	**0.006**▲	**0.010**▲	**0.007**●	**0.011**●	0.018	**0.005**●	**0.009**●	**0.015**●
UCB	0.001	0.001	0.002	0.001	0.003	0.005	0.001	0.002	0.005
e-Greedy	0.001	0.001	0.002	0.001	0.003	0.005	0.001	0.002	0.004
TS	0.001	0.002	0.004	0.000	0.001	0.003	0.002	0.004	0.007
Linear UCB	0.003	0.005	0.008	**0.007**●	**0.012**●	0.018	**0.006**●	**0.008**●	0.012
Linear e-Greedy	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001
Cluster Bandit	0.002	0.005	0.009	0.005	0.011	**0.019**●	0.003	0.007	0.012
NICF	0.001	0.004	0.005	0.004	0.008	0.012	0.003	0.006	0.008
PTS	0.003	0.005	0.008	0.005	0.009	**0.019**●	0.004	0.008	**0.014**●
ICTRTS	0.002	0.005	0.009	0.003	0.008	0.015	0.002	0.005	0.009
Métrica	ILD			ILD			ILD		
T	5	10	20	5	10	20	5	10	20
CLinUCB	0.466	0.459	0.457	0.472	0.474	0.475	0.471	0.466	0.464
UCB	0.491	0.491	0.490	0.488	0.488	0.488	0.490	0.490	0.490
e-Greedy	0.491	0.490	0.490	0.488	0.488	0.488	0.491	0.490	0.490
TS	0.476	0.480	0.483	0.494	0.493	0.492	0.477	0.479	0.482
Linear UCB	0.432	0.461	0.473	0.460	0.471	0.480	0.454	0.465	0.473
Linear e-Greedy	**0.499**▲	**0.499**▲	**0.499**▲	**0.497**▲	**0.495**▲	**0.496**▲	**0.497**▲	**0.498**▲	**0.498**▲
Cluster Bandit	0.475	0.456	0.460	0.487	0.478	0.478	0.484	0.477	0.471
NICF	0.486	0.473	0.486	0.474	0.481	0.485	0.481	0.478	0.489
PTS	0.476	0.476	0.476	0.483	0.482	0.481	0.476	0.477	0.478
ICTRTS	0.477	0.466	0.465	0.488	0.484	0.482	0.491	0.485	0.480
Métrica	UsersCoverage			UsersCoverage			UsersCoverage		
T	5	10	20	5	10	20	5	10	20
CLinUCB	**0.055**▲	**0.093**▲	**0.146**▲	0.076	0.127	0.197	0.068	0.107	0.175
UCB	0.013	0.024	0.043	0.019	0.035	0.071	0.157	0.161	0.170
e-Greedy	0.011	0.021	0.040	0.018	0.038	0.071	0.017	0.033	0.062
TS	0.023	0.039	0.063	0.006	0.018	0.043	0.018	0.026	0.039
Linear UCB	0.037	0.075	0.119	**0.084**▲	**0.134**▲	0.193	**0.231**▲	**0.250**▲	**0.275**▲
Linear e-Greedy	0.002	0.004	0.007	0.001	0.003	0.005	0.045	0.046	0.050
Cluster Bandit	0.039	0.078	0.136	0.059	0.117	0.196	0.037	0.080	0.139
NICF	0.026	0.065	0.078	0.055	0.096	0.134	0.037	0.075	0.100
PTS	0.042	0.074	0.122	0.058	0.112	**0.210**▲	0.052	0.099	0.172
ICTRTS	0.038	0.081	0.129	0.036	0.089	0.167	0.020	0.059	0.116

Tabela 10 – Performance dos modelos *bandit* sob as três cidades extraídas da base de dados da Yelp. Os resultados foram comparados com o teste de Wilcoxon com p-value = 0.05. O símbolo ▲ denota ganhos estatísticos e o símbolo ● representa empates.

e TS) não são capazes de recuperar itens relevantes para o usuário. Mesmo com uma etapa de treinamento, os modelos não conseguem capturar as preferências dos usuários e fazer recomendações satisfatórias. Por outro lado, os demais algoritmos, que combinam as estratégias interativas com conceitos clássicos de Filtragem Colaborativa como *Clustering* e *Matrix Factorization*, conseguem alcançar melhores resultados, por isso, destacam-se as estratégias de *Linear UCB* e *CLinUCB*. O primeiro consiste em uma combinação do UCB

com abordagens lineares de *Matrix Factorization*. Por sua vez, o *CLinUCB* consegue um desempenho ainda superior pois adiciona um contexto não personalizado, baseado apenas nas características dos itens, às abordagens lineares. Contudo, para atingir níveis ainda maiores de acurácia, é importante aprofundar nos conceitos explorados no cenário tradicional de POIs, adicionando aos modelos informações de influência temporal, social e, principalmente, geográfica.

Considerando a diversidade, podemos notar níveis similares de ILD para todos algoritmos, com exceção do ϵ-Greedy. Tal resultado é também relacionado à característica do cenário, uma vez que os POIs tendem a ser visitados por grupos distintos de usuários – em geral, os usuários que vão a bares não necessariamente também irão a museus e bibliotecas. O ϵ-Greedy destaca-se dos demais, pois tem um fator de diversificação com uma probabilidade $1 - \epsilon$ de produzir recomendações aleatórias. Por outro lado, quando analisamos a métrica de *Users Coverage*, os algoritmos de *Linear UCB* e *CLinUCB* se destacam, pois essa métrica mensura o número de usuários distintos que receberam pelo menos 1 item relevante durante as recomendações. Assim, não basta apenas diversificar as recomendações para alcançar distintos nichos de usuários, pois é necessário ter também itens relevantes para eles. Nesse caso, adicionar a informação social entre os usuários poderia também ajudar a maximizar tais indicadores, pois usuários no mesmo círculo de amizade tendem a visitar lugares parecidos.

Em síntese, pode-se notar a relevância do *iRec* para permitir que modelos MAB possam ser avaliados em um contexto ainda não explorado. No cenário de POIs, por exemplo, as poucas propostas de recomendações interativas não consideram modelos MAB sequer como linhas de bases (*baselines*). Com a aplicação do *framework* criamos perspectivas para novas oportunidades de trabalhos, realizando comparações justas e reprodutíveis.

6.3 Síntese do Capítulo

Neste capítulo foi apresentada a reposta para a quarta e última pergunta deste trabalho: **QP4: O quão abrangente um ambiente reprodutível pode ser a fim de suprir as necessidades dos diversos cenários em SsR interativos?** Para se chegar à resposta, avaliamos o *iRec* com base em dois estudos de caso para demonstrar toda a abrangência que sua estrutura pode trazer nas pesquisas atuais focadas em SsR interativos. Em cada um dos estudos, elaboramos um experimento completo a fim de exemplificar todo o processo de configuração, desde a entrada dos dados até avaliação e visualização dos resultados. Em suma, no primeiro estudo focamos no cenário de entretenimento, no qual selecionamos três conjuntos de dados amplamente conhecidas em seus domínios: *Netflix* (Filmes), *Good Books* (Livros) e *Yahoo Music* (Música). Por sua vez, no segundo estudo, focamos em um cenário novo de aplicação de modelos de MAB para re-

comendação interativa de POIs. Para este segundo cenário, selecionamos três coleções de dados reais provenientes da Yelp, relacionadas a três cidades estadunidenses: *Philadelphia*, *Nashville* e *New Orleans*. Dessa forma, para ambos, demonstramos o processo de configuração de um experimento, em detalhes, para os três componentes principais de *iRec*: *Environment*, *Recommendation Agent* e *Experimental Evaluation*. Durante esse processo, definimos as configurações de cada conjunto de dados utilizados, as configurações de cada agente, como realizar a tunagem de parâmetros e, por fim, como avaliar adequadamente os resultados dos modelos MAB. Enfim, para cada estudo, apresentamos os resultados nos quais pudemos avaliar o comportamento desses modelos nos diferentes cenários utilizados. Analisando os resultados obtidos no Estudo 1, vimos a capacidade que o *iRec* possui em promover um ambiente totalmente completo para avaliação em um dos cenários mais utilizados nos trabalhos atuais: o de entretenimento. Por sua vez, no Estudo 2, apesar dos modelos MAB ainda não serem explorados no cenário de POI, demonstramos como é fácil a criação de um experimento nessa área. Portanto, deixamos claro que o *iRec* pode ser estendido, incluindo diferentes coleções de dados, políticas de avaliação, estratégias de recomendação e fontes de dados, de maneira extremamente intuitiva e prática. Assim, com bases nessas análises, em reposta à **QP4** , demonstramos a importância prática da nossa proposta, por proporcionar um ambiente de comparação imparcial e justa entre distintos modelos de SsR com diversas metodologias de avaliação amplamente testadas e utilizadas na literatura. Trata-se de um marco na literatura de sistemas de recomendação interativos, uma vez que com o *iRec* é possível estender os trabalhos na área ou mesmo estabelecer um *benchmark* para os trabalhos da literatura de sistemas de recomendação.

7 Conclusões & Trabalhos Futuros

Este trabalho teve por objetivo atenuar a lacuna presente no processo de avaliação das soluções de Multi-Armed Bandits (MAB). Para isso, propusemos um novo *framework* para avaliar sistemas de recomendação interativos, denominado *iRec*, o qual tem como objetivo fornecer uma comparação imparcial e justa entre distintos modelos de SsR com diversas metodologias de avaliações amplamente testadas e utilizadas na literatura. Nesse sentido, para conduzir a realização da pesquisa, levantamos quatro questões de pesquisa, que respondemos ao longo do trabalho, são elas: **QP1**: Como os modelos MAB vêm sendo avaliados no cenário de sistemas de recomendação interativa?; **QP2**: Existe na literatura algum *framework* capaz de englobar todas etapas (desde a coleta dos dados até a avaliação dos resultados) para experimentação offline para o cenário de sistemas de recomendação interativos?; **QP3**: É possível prover um ambiente de execução reprodutível no cenário de sistemas de recomendação interativos capaz de amenizar a lacuna presente no processo de avaliação das soluções de MAB?; **QP4**: O quão abrangente um ambiente reprodutível pode ser a fim de suprir as necessidades dos diversos cenários em SsR interativos?

Para alcançar esse objetivo, inicialmente, falamos sobre a importância dos sistemas de recomendação e como têm sido utilizados em uma ampla variedade de aplicações. Em seguida, apresentamos toda a fundamentação teórica necessária para compreender o problema de recomendação e como o mesmo vem sendo modelado/utilizado em cenários interativos por meio de abordagens de aprendizado por reforço. De forma sucinta, recomendações iniciais são realizadas para os usuários com base em seu perfil de consumo atual e, a partir do *feedback* fornecido pelos usuários, esse perfil de consumo é atualizado online de forma que as próximas recomendações sejam também atualizadas. Nesse sentido, destacamos e apresentamos os modelos MAB, que têm sido amplamente utilizados no cenário de recomendação interativa, descrevendo em detalhes como esses modelos de MAB lidam com a tarefa de recomendação online e as abordagens mais tradicionais que são base para o amplo conjunto de modelos existentes.

Apresentada a fundamentação teórica, nosso primeiro passo foi responder à **QP1**. Para isso, no Capítulo 3, descrevemos uma revisão sistemática da literatura (RSL) sobre modelos Multi-Armed Bandits aplicados em sistemas de recomendação interativo visando compreender como esses modelos vêm sendo avaliados, identificando as principais metodologias, coleções de dados e métricas de qualidade consideradas. Analisamos um total 1.327 artigos publicados nos últimos 20 anos (2000-2020), observamos que existe uma grande variabilidade de coleções de dados consideradas no processo de avaliação de modelos MAB em recomendação interativa e que muitas dessas coleções ainda são sintéticas. Observamos também que não existe um consenso quanto às estratégias de preprocessamento que

devem ser aplicadas. Essas duas questões impactam significativamente a capacidade reprodutibilidade das avaliações das estratégias propostas na literatura. Quanto às políticas de avaliação, observamos limitações quanto à simulação offline de recomendação interativa. As avaliações ora são focadas apenas no processo de aprendizagem, o que está muito relacionado à teoria de aprendizagem por reforço, a qual foi adaptada ao cenário de recomendação, ora são focadas apenas no resultado final das recomendação (acurácia das recomendações aos usuários). Não identificamos políticas de avaliação que consideram ambos objetivos. Outra lacuna identificada são nas métricas utilizadas nas avaliações, focadas quase que exclusivamente em acurácia, sem considerar outras perspectivas de satisfação dos usuários (novidade, diversidade, etc.). *Dessa forma, nossa RSL apontou que, em resposta à **QP1**, não existe uma padronização clara e reprodutível para estratégias de MAB aplicadas em sistemas de recomendação interativos.*

Respondida a primeira pergunta, avançamos nossa pesquisa visando responder à **QP2**. Para isso, no Capítulo 4 apresentamos uma revisão da literatura de trabalhos que propuseram frameworks e/ou bibliotecas aspirando lidar com a avaliação de algoritmos MAB no cenário de recomendação interativa. Após uma ampla inspeção da literatura, encontramos apenas três trabalhos que recentemente tentaram lidar com a avaliação de algoritmos MAB na área de recomendação: *BEARS* (BARRAZA-URBINA, 2018), *OBP* (SAITO, 2020) e *MABWise* (STRONG, 2021; STRONG, 2019). Apresentamos detalhadamente a arquitetura de cada uma dessas propostas, bem como seus principais componentes. Em seguida, realizamos uma comparação dessas três propostas encontradas na literatura com o nosso *Framework iRec*. Apresentamos uma visão geral do módulo de preparação dos dados, dos modelos de recomendação disponíveis em cada framework/biblioteca e do módulo de avaliação. Através desse estudo comparativo, foi possível verificar que nenhum dos trabalhos existentes até o momento engloba integralmente o processo experimental de avaliação de um sistema de recomendação. Assim, em resposta à **QP2**, constatamos que não existe na literatura nenhum *framework* capaz de englobar todas etapas (desde a coleta dos dados até a avaliação dos resultados) para experimentação offline de um sistema de recomendação no cenário interativo.

Com base nas constatações acima, vislumbramos então a proposta do *iRec*, um *framework* completo para avaliação de sistemas interativos de recomendação. Nosso *framework* visa lidar com a falta de consenso sobre as melhores práticas de avaliação nesta área (SUN, 2020), fornecendo um ambiente completo para uma avaliação reprodutível e comparações justas de sistemas de recomendação. Com o objetivo de responder nossa **QP3**, no Capítulo 5, apresentamos toda a arquitetura do *iRec*, como seus principais componentes. O *iRec* fornece três módulos para preparar os conjuntos de dados, criar novos agentes de recomendação e simular diferentes cenários interativos. Além disso, também contém dezessete algoritmos de última geração, um módulo de ajuste de hiperparâmetros, um vasto conjunto de métricas de avaliação com objetivos distintos (i.e., acurácia, novi-

dade, diversidade), diferentes formas de visualização dos resultados e um processo de validação estatística para comparar o desempenho dos algoritmos. Ao final deste capítulo, demonstramos as diferentes formas de utilização do *iRec* (linhas de comando ou instanciando a biblioteca), o que permite que o pesquisador escolha a forma com que possui mais intimidade e facilidade de utilização. Dessa forma, respondemos positivamente à **QP3**, provendo um ambiente de execução reprodutível no cenário de sistemas de recomendação interativos.

Por fim, visando responder nossa **QP4** e demonstrar os benefícios proporcionados pelo nosso framework, no Capítulo 6, realizamos um estudo de caso no qual demonstramos a amplitude que nosso *framework iRec* tem nas pesquisas atuais sobre SsR interativos. Para isso, apresentamos todo o processo de configuração em dois cenários distintos de recomendação: filmes, músicas, livros, POI (points-of-interest). Em cada um destes cenários, descrevemos todas as bases de dados utilizadas, configuração necessária para cada experimento e os resultados obtidos em cada um deles. Tais resultados mostram a efetividade do *iRec* para tal fim, apontando o desempenho de todos os algoritmos interativos nos diversos conjuntos de dados utilizados. Em cada destes cenários, fizemos uma comparação de desempenho entre dezessete algoritmos de última geração, considerando cinco métricas de avaliação diferentes. Analisando brevemente os resultados, é possível observar que nenhum dos algoritmos obteve os melhores resultados para todas as métricas avaliadas em todas as etapas de interação, mostrando a importância de utilizar diferentes métricas de avaliação e de definir adequadamente o protocolo de avaliação para analisar o comportamento dos modelos nas diferentes etapas. Também demonstramos a relevância do ajuste de hiperparâmetros comparando os resultados dos algoritmos com e sem este processo. Todas essas análises foram possíveis pois o *iRec* permite realizar testes estatísticos para empreender uma avaliação justa. O processo foi realizado em cinco computadores com configurações de hardware distintas, obtendo-se as mesmas conclusões. Assim, respondemos à **QP4** demonstrando a importância da nossa proposta no que se refere a proporcionar um ambiente que fornece uma comparação imparcial e justa entre distintos modelos de SsR com diversas metodologias de avaliação amplamente testadas e utilizadas na literatura. Em nossa opinião, trata-se de um marco na literatura de sistemas de recomendação interativas, uma vez que, além da oportunidade de aplicação dos modelos de recomendação de forma sistematizada e simplificada, os pesquisadores da área podem usar o *iRec* para avançar em seus estudos e experimentos, garantindo o uso de melhores práticas em SsR. Com o *iRec* é possível estender os trabalhos na área e também estabelecer um *benchmark* para os trabalhos da literatura de sistemas de recomendação.

Para trabalhos futuros, de forma mais imediata, planejamos estender *iRec* para incluir novas estratégias de filtragem e divisão de dados ideais para cada um dos cenários suportados por *iRec*. Sabemos que, dependendo do cenário aplicado, algumas técnicas de divisão de dados, por exemplo, podem ser melhores para desenvolvimento de um processo experimental. Além disso, temos como objetivo adicionar novas técnicas de ajuste de pa-

râmetros, como a estratégia Bayesiana, randômica, dentre outras que são amplamente utilizadas no processo de otimização (tunagem) de parâmetros de algoritmos de aprendizado de máquina. Novas políticas e métricas de avaliações também podem ser adicionadas para tornar ainda mais completa a avaliação de modelos *Multi-Armed Bandits*. A longo prazo, nossa meta é utilizar o *iRec* nos demais trabalhos de nosso grupo de pesquisa relacionado a modelos MAB em recomendação interativas. Nosso laboratório conta hoje com outros alunos de iniciação científica, mestrado e doutorado trabalhando com soluções para diversos problemas de recomendação interativas, tais como esparsidade, privacidade, *cold-start* e explicabilidade. Além de utilizarem o *iRec* como ferramenta de apoio, em alguns casos também serão feitas propostas de melhorias e extensões para que o mesmo também seja capaz de realizar experimentações detalhadas de recomendação interativas para os problemas mencionados.

Referências

SAITO, Y.; SHUNSUKE, A.; MEGUMI, M.; YUSUKE, N. Open bandit dataset and pipeline: Towards realistic and reproducible off-policy evaluation. *arXiv preprint arXiv:2008.07146*, 2020. Citado 7 vezes nas páginas 9, 17, 36, 37, 38, 43 e 74.

BARRAZA-URBINA, A.; KOUTRIKA, G.; D'AQUIN, M.; HAYES, C. Bears: Towards an evaluation framework for bandit-based interactive recommender systems. *REVEAL 18, October 6-7, 2018, Vancouver, Canada*, NUI Galway, 2018. Citado 8 vezes nas páginas 9, 17, 36, 38, 39, 43, 61 e 74.

STRONG, E.; KLEYNHANS, B.; KADIOGLU, S. MABWiser: parallelizable contextual multi-armed bandits. *Int. J. Artif. Intell. Tools*, v. 30, n. 4, 2021. Citado 7 vezes nas páginas 9, 17, 36, 40, 41, 43 e 74.

STRONG, E.; KLEYNHANS, B.; KADIOGLU, S. Mabwiser: A parallelizable contextual multi-armed bandit library for python. In: *31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4-6, 2019*. IEEE, 2019. p. 909–914. Disponível em: <https://doi.org/10.1109/ICTAI.2019.0012>. Citado 6 vezes nas páginas 9, 17, 40, 41, 43 e 74.

SCHWARTZ, B.; SCHWARTZ, B. The paradox of choice: Why more is less. In: ECCO NEW YORK. [S.l.], 2004. Citado na página 13.

RICCI, F.; ROKACH, L.; SHAPIRA, B. Recommender systems: introduction and challenges. In: *Recommender systems handbook*. [S.l.]: Springer, 2015. p. 1–34. Citado na página 13.

BOBADILLA, J.; ORTEGA, F.; HERNANDO, A.; GUTIÉRREZ, A. Recommender systems survey. *Knowledge-based systems*, Elsevier, v. 46, p. 109–132, 2013. Citado 3 vezes nas páginas 13, 24 e 33.

MOURAO, F. H. de J. A hybrid recommendation method that combines forgotten items and non-content attributes. Universidade Federal de Minas Gerais, 2014. Citado 3 vezes nas páginas 13, 14 e 20.

ADOMAVICIUS, G.; TUZHILIN, A. 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*, IEEE, v. 17, n. 6, p. 734–749, 2005. Citado 3 vezes nas páginas 13, 14 e 20.

ZHAO, X.; ZHANG, W.; WANG, J. Interactive collaborative filtering. In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. [S.l.: s.n.], 2013. p. 1411–1420. Citado 5 vezes nas páginas 14, 21, 22, 25 e 48.

WU, Q.; WANG, H.; GU, Q.; WANG, H. Contextual bandits in a collaborative environment. In: *Proceedings of the 39th International ACM SIGIR Conference on Development in Information Retrieval*. [S.l.: s.n.], 2016. Citado na página 14.

WANG, Y.-X.; HEBERT, M. Learning to learn: Model regression networks for easy small sample learning. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2016. p. 616–634. Citado na página 14.

WANG, H.; WU, Q.; WANG, H. Factorization bandits for interactive recommendation. In: *Thirty-First AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2017. Citado 2 vezes nas páginas 14 e 25.

SANZ-CRUZADO, J.; CASTELLS, P.; LÓPEZ, E. A simple multi-armed nearest-neighbor bandit for interactive recommendation. In: *Proceedings of the 13th ACM Conference on Recommender Systems*. [S.l.: s.n.], 2019. p. 358–362. Citado 5 vezes nas páginas 14, 21, 22, 49 e 52.

AUER, P.; CESA-BIANCHI, N.; FISCHER, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, Springer, v. 47, n. 2-3, p. 235–256, 2002. Citado 7 vezes nas páginas 14, 18, 21, 22, 26, 48 e 61.

AUER, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, v. 3, n. Nov, p. 397–422, 2002. Citado na página 14.

CHAPELLE, O.; LI, L. An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, Citeseer, v. 24, p. 2249–2257, 2011. Citado 7 vezes nas páginas 15, 18, 22, 25, 26, 48 e 61.

ABEILLE, M.; LAZARIC, A. Linear thompson sampling revisited. In: PMLR. *Artificial Intelligence and Statistics*. [S.l.], 2017. p. 176–184. Citado 3 vezes nas páginas 15, 25 e 48.

WANG, Q.; ZENG, C.; ZHOU, W.; LI, T.; IYENGAR, S. S.; SHWARTZ, L.; GRABARNIK, G. Y. Online interactive collaborative filtering using multi-armed bandit with dependent arms. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 31, n. 8, p. 1569–1580, 2018. Citado 5 vezes nas páginas 15, 21, 25, 32 e 48.

SHAMS, S.; ANDERSON, D.; LEITH, D. Cluster-based bandits: Fast cold-start for recommender system new users. 2021. Citado 3 vezes nas páginas 15, 21 e 49.

SUN, Z.; YU, D.; FANG, H.; YANG, J.; QU, X.; ZHANG, J.; GENG, C. Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison. In: *Fourteenth ACM conference on recommender systems*. [S.l.: s.n.], 2020. p. 23–32. Citado 3 vezes nas páginas 15, 57 e 74.

SILVA, N.; WERNECK, H.; SILVA, T.; PEREIRA, A.; ROCHA, L. Multi-armed bandits in recommendation systems: A survey of the state-of-the-art and future directions. *Expert Systems With Applications*, v. 196, n. -, p. –, 2022. ISSN 0957-4174. Disponível em: <>. Citado 6 vezes nas páginas 15, 16, 18, 27, 28 e 35.

ANONYMOUS. anonymous. In: *anonymous*. [S.l.]: anonymous, anonymous. Citado 3 vezes nas páginas 17, 18 e 21.

SILVA, T.; SILVA, N.; WERNECK, H.; PEREIRA, A. C.; ROCHA, L. The impact of first recommendations based on exploration or exploitation approaches in recommender systems' learning. In: *Proceedings of the Brazilian Symposium on Multimedia and the Web*. [S.l.: s.n.], 2020. p. 173–180. Citado na página 18.

SILVA, N.; WERNECK, H.; SILVA, T.; PEREIRA, A. C.; ROCHA, L. A contextual approach to improve the user's experience in interactive recommendation systems. In: *Proceedings of the Brazilian Symposium on Multimedia and the Web*. [S.l.: s.n.], 2021. p. 89–96. Citado 2 vezes nas páginas 18 e 48.

AMATRIAIN, X.; BASILICO, J. Recommender systems in industry: A netflix case study. In: *Recommender systems handbook*. [S.l.]: Springer, 2015. p. 385–419. Citado na página 20.

AMATRIAIN, X.; BASILICO, J. Past, present, and future of recommender systems: An industry perspective. In: *Proceedings of the 10th ACM conference on recommender systems*. [S.l.: s.n.], 2016. p. 211–214. Citado na página 20.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018. Citado 2 vezes nas páginas 21 e 22.

WU, Q.; ZHANG, H.; GAO, X.; HE, P.; WENG, P.; GAO, H.; CHEN, G. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In: *The World Wide Web Conference*. [s.n.], 2019. p. 2091–2102. Disponível em: <https://dl.acm.org/doi/abs/10.1145/3308558.331344>. Citado na página 21.

ZHOU, S.; DAI, X.; CHEN, H.; ZHANG, W.; REN, K.; TANG, R.; HE, X.; YU, Y. Interactive recommender system via knowledge graph-enhanced reinforcement learning. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. [S.l.: s.n.], 2020. Citado na página 21.

WU, Q.; IYER, N.; WANG, H. Learning contextual bandits in a non-stationary environment. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. [S.l.: s.n.], 2018. p. 495–504. Citado na página 21.

ZOU, L.; XIA, L.; GU, Y.; ZHAO, X.; LIU, W.; HUANG, J. X.; YIN, D. Neural interactive collaborative filtering. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. [S.l.: s.n.], 2020. p. 749–758. Citado 2 vezes nas páginas 21 e 49.

HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. [S.l.]: MIT press, 1992. Citado na página 21.

LI, Y.; LAN, C.; XING, J.; ZENG, W.; YUAN, C.; LIU, J. Online human action detection using joint classification-regression recurrent neural networks. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 203–220. Citado na página 21.

SUTTON, R. S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In: *Advances in neural information processing systems*. [S.l.: s.n.], 1996. p. 1038–1044. Citado na página 23.

THOMPSON, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, JSTOR, v. 25, n. 3/4, p. 285–294, 1933. Citado na página 24.

AGRAWAL, S.; GOYAL, N. Analysis of thompson sampling for the multi-armed bandit problem. *Journal of Machine Learning Research*, v. 23, 11 2011. Citado na página 24.

SU, X.; KHOSHGOFTAAR, T. M. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, Hindawi, v. 2009, 2009. Citado na página 24.

LI, L.; CHU, W.; LANGFORD, J.; SCHAPIRE, R. E. A contextual-bandit approach to personalized news article recommendation. In: *Proceedings of the 19th international conference on World wide web*. [S.l.: s.n.], 2010. p. 661–670. Citado 2 vezes nas páginas 25 e 48.

HARIRI, N.; MOBASHER, B.; BURKE, R. Context adaptation in interactive recommender systems. In: *Proceedings of the 8th ACM Conference on Recommender systems*. [S.l.: s.n.], 2014. p. 41–48. Citado 2 vezes nas páginas 25 e 32.

KAWALE, J.; BUI, H. H.; KVETON, B.; THANH, L. T.; CHAWLA, S. Efficient thompson sampling for online matrix-factorization recommendation. *Advances in Neural Information Processing Systems*, v. 28, p. 1297–1305, 2015. Citado 2 vezes nas páginas 25 e 49.

ÇANO, E.; MORISIO, M. Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, IOS Press, v. 21, n. 6, p. 1487–1524, 2017. Citado na página 27.

KEELE, S. *Guidelines for performing systematic literature reviews in software engineering*. [S.l.], 2007. Citado na página 29.

CREMONESI, P.; GARZOTTO, F.; NEGRO, S.; PAPADOPOULOS, A. V.; TURRIN, R. Looking for "good" recommendations: A comparative evaluation of recommender systems. In: SPRINGER. *IFIP Conference on Human-Computer Interaction*. [S.l.], 2011. p. 152–168. Citado na página 30.

CHEN, M.; LIU, P. Performance evaluation of recommender systems. *International Journal of Performability Engineering*, v. 13, n. 8, 2017. Citado na página 30.

VARGAS, S.; CASTELLS, P. Rank and relevance in novelty and diversity metrics for recommender systems. In: *Proceedings of the fifth ACM conference on Recommender systems*. [S.l.: s.n.], 2011. p. 109–116. Citado 4 vezes nas páginas 30, 33, 49 e 51.

CASTELLS, P.; VARGAS, S.; WANG, J. Novelty and diversity metrics for recommender systems: Choice, discovery and relevance. *Proceedings of International Workshop on Diversity in Document Retrieval (DDR)*, 01 2011. Citado 2 vezes nas páginas 30 e 33.

VARGAS, S. New approaches to diversity and novelty in recommender systems. In: *Fourth BCS-IRSG Symposium on Future Directions in Information Access (FDIA 2011) 4*. [S.l.: s.n.], 2011. p. 8–13. Citado na página 30.

KROHN-GRIMBERGHE, A.; NANOPOULOS, A.; SCHMIDT-THIEME, L. A novel multidimensional framework for evaluating recommender systems. In: *LWA*. [S.l.: s.n.], 2010. p. 113–120. Citado na página 30.

SHANI, G.; GUNAWARDANA, A. Evaluating recommendation systems. In: *Recommender systems handbook*. [S.l.]: Springer, 2011. p. 257–297. Citado na página 30.

CAÑAMARES, R.; REDONDO, M.; CASTELLS, P. Multi-armed recommender system bandit ensembles. In: *Proceedings of the 13th ACM Conference on Recommender Systems*. [s.n.], 2019. p. 432–436. Disponível em: <https://dl.acm.org/doi/abs/10.1145-/3298689.334698>. Citado na página 32.

BASU, S.; SEN, R.; SANGHAVI, S.; SHAKKOTTAI, S. Blocking bandits. In: *Advances in Neural Information Processing Systems*. [s.n.], 2019. p. 4784–4793. Disponível em: <http://papers.nips.cc/paper/8725-blocking-bandit>. Citado na página 32.

ZHAO, C.; YANG, B.; HIRATE, Y. A reward optimization model for decision-making under budget constraint. *Journal of Information Processing*, Information Processing Society of Japan, v. 27, p. 190–200, 2019. Disponível em: <https://www-.jstage.jst.go.jp/article/ipsjjip/27/0/27_190/_article/-char/ja>. Citado na página 32.

KATARIYA, S.; KVETON, B.; WEN, Z.; POTLURU, V. K. Conservative exploration using interleaving. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. [s.n.], 2019. p. 954–963. Disponível em: <http://proceedings.mlr.press/v89-/katariya19a.htm>. Citado na página 32.

CHATTERJI, N.; MUTHUKUMAR, V.; BARTLETT, P. Osom: A simultaneously optimal algorithm for multi-armed and linear contextual bandits. In: *International Conference on Artificial Intelligence and Statistics*. [S.l.: s.n.], 2020. p. 1844–1854. Citado na página 32.

HAO, B.; YADKORI, Y. A.; WEN, Z.; CHENG, G. Bootstrapping upper confidence bound. In: *Advances in Neural Information Processing Systems*. [s.n.], 2019. p. 12123–12133. Disponível em: <http://papers.nips.cc/paper/9382-bootstrapping-upper-confidence-boun>. Citado na página 32.

LI, S.; WANG, B.; ZHANG, S.; CHEN, W. Contextual combinatorial cascading bandits. In: *ICML*. [s.n.], 2016. v. 16, p. 1245–1253. Disponível em: <http://www.jmlr.org-/proceedings/papers/v48/lif16-supp.pd>. Citado na página 32.

MAY, B. C.; KORDA, N.; LEE, A.; LESLIE, D. S. Optimistic bayesian sampling in contextual-bandit problems. *The Journal of Machine Learning Research*, JMLR. org, v. 13, n. 1, p. 2069–2106, 2012. Disponível em: <https://dl.acm.org/doi/abs/10.5555-/2503308.234371>. Citado na página 32.

WANG, L.; WANG, C.; WANG, K.; HE, X. Biucb: A contextual bandit algorithm for cold-start and diversified recommendation. In: IEEE. *2017 IEEE International Conference on Big Knowledge (ICBK)*. [S.l.], 2017. p. 248–253. Citado na página 32.

LIU, W.; LI, S.; ZHANG, S. Contextual dependent click bandit algorithm for web recommendation. In: SPRINGER. *International Computing and Combinatorics Conference*. 2018. p. 39–50. Disponível em: <https://link.springer.com/chapter/10.1007-/978-3-319-94776-1_>. Citado na página 32.

TEO, C. H.; NASSIF, H.; HILL, D.; SRINIVASAN, S.; GOODMAN, M.; MOHAN, V.; VISHWANATHAN, S. Adaptive, personalized diversity for visual discovery. In: *Proceedings of the 10th ACM conference on recommender systems*. [s.n.], 2016. p. 35–38. Disponível em: <https://dl.acm.org/doi/abs/10.1145/2959100.295917>. Citado na página 32.

SUI, Y.; GOTOVOS, A.; BURDICK, J.; KRAUSE, A. Safe exploration for optimization with gaussian processes. In: PMLR. *International Conference on Machine Learning*. 2015. p. 997–1005. Disponível em: <http://proceedings.mlr.press/v37/sui15.htm>. Citado na página 32.

CHRISTAKOPOULOU, K.; BANERJEE, A. Learning to interact with users: A collaborative-bandit approach. In: SIAM. *Proceedings of the 2018 SIAM International Conference on Data Mining*. 2018. p. 612–620. Disponível em: <https://epubs.siam.org-/doi/abs/10.1137/1.9781611975321.6>. Citado na página 32.

VOROBEV, A.; LEFORTIER, D.; GUSEV, G.; SERDYUKOV, P. Gathering additional feedback on search results by multi-armed bandits with respect to production ranking. p. 1177–1187, 05 2015. Citado na página 32.

ZENG, C.; WANG, Q.; MOKHTARI, S.; LI, T. Online context-aware recommendation with time varying multi-armed bandit. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.: s.n.], 2016. p. 2025–2034. Citado na página 32.

WANG, Q.; LI, T.; IYENGAR, S.; SHWARTZ, L.; GRABARNIK, G. Y. Online it ticket automation recommendation using hierarchical multi-armed bandit algorithms. In: SIAM. *Proceedings of the 2018 SIAM International Conference on Data Mining*. [S.l.], 2018. p. 657–665. Citado na página 32.

LI, L.; CHU, W.; LANGFORD, J.; WANG, X. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In: *Proceedings of the fourth ACM international conference on Web search and data mining*. [S.l.: s.n.], 2011. p. 297–306. Citado na página 32.

TANG, L.; JIANG, Y.; LI, L.; LI, T. Ensemble contextual bandits for personalized recommendation. In: *Proceedings of the 8th ACM Conference on Recommender Systems*. [S.l.: s.n.], 2014. p. 73–80. Citado 2 vezes nas páginas 32 e 61.

FELÍCIO, C.; PAIXÃO, K.; BARCELOS, C.; PREUX, P. A multi-armed bandit model selection for cold-start user recommendation. In: *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. [S.l.: s.n.], 2017. Citado na página 32.

WANG, X.; WANG, Y.; HSU, D.; WANG, Y. Exploration in interactive personalized music recommendation: a reinforcement learning approach. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, ACM New York, NY, USA, v. 11, n. 1, p. 1–22, 2014. Citado na página 32.

CASTELLS, P.; HURLEY, N. J.; VARGAS, S. Novelty and diversity in recommender systems. In: *Recommender systems handbook*. [S.l.]: Springer, 2015. p. 881–918. Citado na página 33.

VARGAS, S. Novelty and diversity enhancement and evaluation in recommender systems and information retrieval. In: *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. [S.l.: s.n.], 2014. p. 1281–1281. Citado na página 33.

KUNAVER, M.; POŽRL, T. Diversity in recommender systems–a survey. *Knowledge-Based Systems*, Elsevier, v. 123, p. 154–162, 2017. Citado na página 33.

ANELLI, V. W.; BELLOGÍN, A.; FERRARA, A.; MALITESTA, D.; MERRA, F. A.; POMO, C.; DONINI, F. M.; NOIA, T. D. Elliot: a comprehensive and rigorous framework for reproducible recommender systems evaluation. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. [S.l.: s.n.], 2021. Citado na página 36.

SALAH, A.; TRUONG, Q.-T.; LAUW, H. W. Cornac: A comparative framework for multimodal recommender systems. *J. Mach. Learn. Res.*, v. 21, p. 95–1, 2020. Citado na página 36.

GUO, G.; ZHANG, J.; SUN, Z.; YORKE-SMITH, N. Librec: A java library for recommender systems. In: CITESEER. *Umap Workshops*. [S.l.], 2015. v. 4. Citado na página 36.

MCINERNEY, J.; BROST, B.; CHANDAR, P.; MEHROTRA, R.; CARTERETTE, B. Counterfactual evaluation of slate recommendations with sequential reward interactions. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. [S.l.: s.n.], 2020. p. 1779–1788. Citado na página 37.

KULESHOV, V.; PRECUP, D. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014. Citado na página 40.

ANELLI, V. W.; NOIA, T. D.; SCIASCIO, E. D.; RAGONE, A.; TROTTA, J. Local popularity and time in top-n recommendation. In: SPRINGER. *European Conference on Information Retrieval*. [S.l.], 2019. p. 861–868. Citado na página 47.

BELLOGÍN, A.; SÁNCHEZ, P. Revisiting neighbourhood-based recommenders for temporal scenarios. In: *RecTemp RecSys*. [S.l.: s.n.], 2017. p. 40–44. Citado na página 47.

LI, S.; KARATZOGLOU, A.; GENTILE, C. Collaborative filtering bandits. In: *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. [S.l.: s.n.], 2016. p. 539–548. Citado na página 49.

RENDLE, S.; KRICHENE, W.; ZHANG, L.; ANDERSON, J. Neural collaborative filtering vs. matrix factorization revisited. In: *Fourteenth ACM Conference on Recommender Systems*. [S.l.: s.n.], 2020. p. 240–248. Citado na página 49.

RICCI, F.; ROKACH, L.; SHAPIRA, B. Introduction to recommender systems handbook. In: *Recommender systems handbook*. [S.l.]: Springer, 2011. p. 1–35. Citado 2 vezes nas páginas 49 e 51.

SILVEIRA, T.; ZHANG, M.; LIN, X.; LIU, Y.; MA, S. How good your recommender system is? a survey on evaluations in recommendation. *International Journal of Machine Learning and Cybernetics*, Springer, v. 10, n. 5, p. 813–831, 2019. Citado 2 vezes nas páginas 49 e 52.

SCHRÖDER, G.; THIELE, M.; LEHNER, W. Setting goals and choosing metrics for recommender system evaluations. In: *UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA*. [S.l.: s.n.], 2011. v. 23, p. 53. Citado na página 51.

ZHOU, G.; ZHU, X.; SONG, C.; FAN, Y.; ZHU, H.; MA, X.; YAN, Y.; JIN, J.; LI, H.; GAI, K. Deep interest network for click-through rate prediction. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. [S.l.: s.n.], 2018. p. 1059–1068. Citado na página 51.

GE, M.; DELGADO-BATTENFELD, C.; JANNACH, D. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In: *Proceedings of the fourth ACM conference on Recommender systems*. [S.l.: s.n.], 2010. p. 257–260. Citado na página 51.

ZHAI, C.; COHEN, W. W.; LAFFERTY, J. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In: ACM NEW YORK, NY, USA. *ACM SIGIR Forum*. [S.l.], 2015. v. 49, n. 1, p. 2–9. Citado na página 51.

WOOLSON, R. F. Wilcoxon signed-rank test. *Wiley encyclopedia of clinical trials*, Wiley Online Library, p. 1–3, 2007. Citado na página 52.

DACREMA, M. F.; BOGLIO, S.; CREMONESI, P.; JANNACH, D. A troubling analysis of reproducibility and progress in recommender systems research. *ACM Transactions on Information Systems (TOIS)*, ACM New York, NY, USA, v. 39, n. 2, 2021. Citado na página 66.

OMIDVAR-TEHRANI, B.; VISWANATHAN, S.; ROULLAND, F.; RENDERS, J.-M. Sage: Interactive state-aware point-of-interest recommendation. In: *WSDM Workshop SUM*. [S.l.: s.n.], 2020. v. 20. Citado na página 66.

YU, Z.; WANG, Y.; CAO, J.; ZHU, G. Poi recommendation with interactive behaviors and user preference dynamics embedding. In: IEEE. *2020 3rd International Conference on Artificial Intelligence and Big Data (ICAIBD)*. [S.l.], 2020. p. 252–258. Citado na página 66.

WANG, D.; LIU, K.; XIONG, H.; FU, Y. Online poi recommendation: Learning dynamic geo-human interactions in streams. *arXiv preprint arXiv:2201.10983*, 2022. Citado na página 66.

LIU, Y.; WEI, W.; SUN, A.; MIAO, C. Exploiting geographical neighborhood characteristics for location recommendation. In: *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*. [S.l.: s.n.], 2014. p. 739–748. Citado na página 67.

YE, M.; YIN, P.; LEE, W.-C.; LEE, D.-L. Exploiting geographical influence for collaborative point-of-interest recommendation. In: *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. [S.l.: s.n.], 2011. p. 325–334. Citado na página 67.