BRUNO KOSTIUK

Um Sistema Híbrido de Recomendação de Testes de Software baseado nas Descrições Textuais de Falhas e Testes

Curitiba - PR, Brasil

2025

BRUNO KOSTIUK

Um Sistema Híbrido de Recomendação de Testes de Software baseado nas Descrições Textuais de Falhas e Testes

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Doutor em Informática.

Pontifícia Universidade Católica do Paraná - PUCPR

Programa de Pós-Graduação em Informática - PPGIa

Orientador: EMERSON CABRERA PARAISO

Coorientador: JÚLIO CESAR NIEVOLA

Curitiba - PR, Brasil

2025

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Luci Eduarda Wielganczuk – CRB 9/1118

Kostiuk, Bruno
K86s Um sistema híbrido de recomendação de testes de software baseado nas
2025 descrições textuais de falhas e testes / Bruno Kostiuk ;orientador: Emerson Cabrera Paraiso ; coorientador: Júlio Cesar Nievola. – 2025.
167 f. : il. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2025
Bibliografia: f. 103-109

1. Informática. 2. Processamento de linguagem natural (Computação). 3. Software – Testes. 4. Aprendizado do computador. I. Paraiso, Emerson Cabrera. II. Nievola, Júlio Cesar. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

CDD 20. ed. – 004

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

Curitiba, 14 de março de 2025.

39-2025

DECLARAÇÃO

Declaro para os devidos fins, que **BRUNO KOSTIUK** defendeu a tese de Doutorado intitulada "**Um Sistema Híbrido de Recomendação de Testes de Software baseado nas Descrições Textuais de Falhas e Testes.**", na área de concentração Ciência da Computação no dia 30 de outubro de 2024, no qual foi aprovado.

Declaro ainda, que foram feitas todas as alterações solicitadas pela Banca Examinadora, cumprindo todas as normas de formatação definidas pelo Programa.

Por ser verdade firmo a presente declaração.

gov.br Documento assinado digitalmente
EMERSON CABRERA PARAISO
Data: 18/03/2025 19:08:43-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Emerson Cabrera Paraiso
Coordenador do Programa de Pós-Graduação em Informática

Resumo

Em um ambiente de Integração Contínua, a execução de testes de regressão é uma prática custosa. Portanto, selecionar os testes de forma precisa, reduz custos. Porém, os testes manuais, que não possuem *scripts* de execução, são deixados de fora nas seleções, pois geralmente são mais custosos em questão de tempo de execução e necessitam de um testador humano. Além disso, testes que nunca apresentaram falhas, porém são de grande importância, são deixados de executar devido ao fato de que as seleções da literatura visam selecionar os testes que apresentam maior taxa de falhas. Neste trabalho, propomos um sistema híbrido de recomendação para testes manuais baseado nas descrições textuais das falhas e nas descrições de passo a passo dos testes, as quais são interligadas não somente por meio da interação entre testes e falhas, mas também por meio de seus dados textuais. Para a criação do sistema de recomendação foi utilizada a arquitetura *Transformer* por meio do modelo pré-treinado *Sentence-BERT*, que realiza a conversão das sentenças descritivas de falhas e testes em vetores numéricos que representam o texto originalmente escrito em linguagem técnica. Os experimentos foram conduzidos de duas formas: a primeira por meio da utilização da base coletada e processada e a segunda, por meio da interação com os usuários finais do sistema, que utilizam uma aplicação que auxilia na interação com o sistema de recomendação. Como resultados conseguimos por meio da avaliação dos experimentos provar que os sistemas de recomendação podem ser utilizados para recomendar testes, experimentos esses que apresentaram resultados próximos a 60% ao empregar a métrica Recall@10 quando utilizado apenas os dados coletados, bem como um resultado médio próximo a 60% ao empregar a métrica Recall@20 considerando a interação com usuários humanos reais do sistema.

Palavras-chave: Seleção de Caso de Teste, Seleção de Testes Manuais, Sistema de Recomendação, Processamento de Linguagem Natural, Transformers, Sistema Híbrido de Recomendação, Recomendação de Testes.

Abstract

In a continuous integration environment, running regression tests is a costly practice. Therefore, selecting tests accurately reduces costs. However, manual tests, which do not have execution scripts, are left out of the selections, as they are generally more costly in execution time and require a human tester. In addition, tests that have never bug but are of great importance are not executed because selections in the literature aim to select the tests that have the highest bug rate. In this work, we propose a hybrid recommendation system for manual tests based on textual descriptions of bugs and step-by-step descriptions of tests. These are interconnected not only through the interaction between tests and bugs but also through their textual data. To create the recommendation system, the Transformer architecture was used through the pre-trained Sentence-BERT model, which converts the descriptive sentences of bugs and tests into numerical vectors that represent that text originally written in technical language. The experiments were conducted in two ways: the first through the use of the collected and processed database and the second, through interaction with the end users of the system, who use an application that assists in the interaction with the recommendation system. As a result, through the evaluation of the experiments, we were able to prove that recommendation systems can be used to recommend tests, experiments that presented results close to 60% using the Recall@10 metric when using only the collected data, as well as an average result close to 60% using the Recall@20 metric when using interaction with real human users of the system.

Keywords: Test Case Selection, Manual Test Selection, Recommendation System, Natural Language Processing, Hybrid Recommendation System, Test Recommendation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AHC	*Aglomerative Hierarchical Clustering* ou Agrupamento hierárquico aglomerativo
ANN	*Artificial Neural Network* ou rede neural artificial
APFDC	*Cost-cognizant Average Percentage of Faults Detected* ou Porcentagem média de falhas detectadas com base em custos
BACP	Base Aleatório com positivos
BCCP	Base Completa com positivos
BCSP	Base Completa sem positivos
BERT	*Bidirectional Encoder Representations from Transformers* ou Representações de codificadores bidirecionais de transformadores
BMCP	Base Modelo com positivos
BNet	*Bayes Net*
BTCP	Base TOPPOP com positivos
CBR	*Case-Based Ranking*
CCS	*Complete ColdStart* ou inicialização a frio completa
CNN	*Convolutional Neural Network* ou Rede Neural Convolucional
CSV	Formato de arquivo com valores separados por vírgula.
DAI	Doutorado Acadêmico de Inovação
Devops	Operações de Desenvolvimento ou *Development-Operations*
DNN	*Deep Neural Network* ou Rede Neural Profunda
DT	*Decision Trees* ou árvore de decisão
ER	*Experience Replay* ou Replay de Experiência
GA	*Genetic Algorithm* ou Algoritmo Genético
GOMEA	*Gene-pool Optimal Mixing Evolutionary Algorithm* ou Algoritmo Evolucionário de mistura ótima de pool genético.

GSDTSR	*Google Shared Dataset of Test Suite Results*
IC	Integração Contínua ou *Continuous Integration*
ICS	*Incomplete ColdStart* ou inicialização a frio incompleta
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
InTOL	biblioteca *Intelligent Test Oracle Library*
LCS	*Learning Classifier Systems* ou Sistemas Classificadores de Aprendizado
Log	*Logistic*
LR	*Learn to Rank* ou Aprender para Ranquear
LSTM	*Long Short Term Memory* ou Memória de Longo Prazo
MT	Teste Manual ou *Manual Test*
NB	*Naive Bayes*
NLP	*Natural Language Processing* ou processamento de linguagem natural
PUCPR	Pontifícia Universidade Católica do Paraná
RF	*Random Forest* ou Floresta aleatória
RL	*Reinforcement learning* ou Aprendizagem por reforço.
RT	Teste de Regressão ou *Regression Test*
SBERT	*Sentence-BERT*
SIR	*Software-artifact Infrastructure Repository*
SP	*Spectrum Power*
SSKM	*KMeans* semi-supervisionado
SVM	*Support Vector Machine*
TFIDF	*Term Frequency-Inverse Document Frequency* ou Termo Frequência-Frequência Inversa do Documento
UI	*User Interface* ou Interface Usuário
XCS	*Classifier System with Experience Replay* ou Sistema Classificador com Replay de Experiência

Lista de símbolos

σ	Letra grega Sigma, utilizada para representar o desvio padrão.
λ	Taxa de regularização.
$\vec{A}_b$	Vetor de pesos da Falha.
$\vec{B}_n$	Vetor de pesos do Teste Negativo.
$\vec{B}_p$	Vetor de pesos do Teste positivo.
e	Exponencial.
F1	Falha 1.
K	Vetor de chave.
n	Taxa de aprendizagem.
Q	Vetor de consulta.
$\hat{r}_{bn}$	Produto vetorial entre Falha e Teste Negativo.
$\hat{r}_{bp}$	Produto vetorial entre Falha e Teste Positivo.
$\hat{r}_{bpn}$	Diferença entre os produtos vetoriais.
T1	Teste 1.
V	Vetor de valor.

Sumário

1 Introdução

Devido ao avanço das melhores práticas de Devops (Operações de Desenvolvimento ou *Development-Operations*), a prática comum de desenvolvimento de software por meio de Integração Contínua (IC) faz com que os desenvolvedores integrem seu trabalho ao código principal várias vezes ao dia. A prática do IC acelera o desenvolvimento e liberação do software, porém amplia os desafios em relação a garantia de qualidade, visto que ao realizar uma integração ao código principal, é necessário verificar se essa interação não irá gerar novas falhas ou reativar falhas já corrigidas (BAGHERZADEH, 2021).

Como forma de verificação de qualidade, o teste de software é a técnica mais importante, pois um testador, por meio de uma sequência de condições e valores de entrada, pode determinar se um software funciona ou não corretamente (WANG, 2011). O testador, sendo ele humano ou máquina (testes automatizados), interage com o software inserindo informações e analisando sua saída, com o objetivo de determinar se o software está funcionando corretamente (MYERS, 2011).

Os testes de software ocorrem durante todo o desenvolvimento do sistema, não apenas em um momento isolado, e dentre os testes realizados, os testes nomeados de testes de regressão (*Regression Test*) representam cerca de 80% do tempo total gasto com teste de software (ABELE; GÖHNER, 2014). Testes de regressão (RT), são testes que devem ser reexecutados periodicamente para verificar se o software não apresenta problemas após a implementação de uma nova funcionalidade, ou após a correção de algum problema já reportado anteriormente (IEEE, 1983).

Desempenhando um papel de importância no ciclo de desenvolvimento de software, a regressão de testes ajuda a verificar se falhas não foram inseridas em versões anteriores do software quando são realizadas integrações de novas funcionalidades ou modificações para correção de falhas (ROSERO, 2017). Quando são realizadas mudanças no código principal de um software, é possível que por descuido do desenvolvedor ou até mesmo por falta de compreensão do código, falhas podem ser geradas por meio de tal mudança.

A medida que um software é ampliado em termos de funcionalidades e modificações, os testes existentes para o mesmo software acompanham esse crescimento (ROSERO, 2017). Como exemplo deste crescimento podemos citar a empresa Google, que em 2019 executava mais de 100 milhões de testes por dia (LOU, 2019). A demanda computacional e até mesmo humana para a realização e manutenção destes testes estão entrelaçados com esse crescimento, se fazendo assim necessário formas de otimizar tal execução.

Como forma de reduzir os custos ao realizar uma regressão de testes, foram propostas abordagens como: redução ou minimização; seleção; e priorização de casos de

testes (SIMONS; PARAISO, 2010). A redução ou minimização de casos de testes busca encontrar e eliminar testes duplicados ou que estejam inclusos dentro de um ou mais testes. A seleção de casos de testes por outro lado seleciona apenas os testes que têm relação com o código alterado. Já a priorização de casos de testes ordena os testes de forma a se encontrar falhas de maneira precoce (LOU, 2019).

Em um ambiente de desenvolvimento por meio de IC, onde o desenvolvimento e manutenção são continuamente executados, a execução de RT é uma prática ainda mais custosa, pois a quantidade de testes em um determinado software pode atrasar a entrega dele, porém são necessárias para garantir que o software em questão esteja funcionando da forma esperada. Para resolver esse problema, as três abordagens anteriormente mencionadas (redução ou minimização, seleção e priorização) são de grande valia para acelerar o processo de teste do software, bem como encontrar possíveis falhas o quanto antes, acelerando assim o início de suas tratativas (YU; WANG, 2018).

Segundo (BUCHGEHER, 2013), a regressão de testes é uma atividade demorada, portanto o ideal é realizar a automação desses testes, automações essas que são executadas como parte dos processos de construção do software. Embora a automação dos RT seja altamente desejável, elas nem sempre podem ser alcançadas para grandes sistemas de software contendo um legado de componentes e tecnologias de testabilidade variável. Exemplos típicos que impedem essa automação são os testes de sistema que requerem interações diretas com o usuário ou hardware, ou testes que requerem inspeções manuais dos resultados (por exemplo, se todos os elementos da interface do usuário estão devidamente organizados e as representações gráficas são claras e/ou atraentes). Os testes que não foram automatizados precisam ser executados manualmente por um humano e são chamados de testes manuais (*manual test* - MT). Os testes manuais possuem em sua documentação, um passo a passo de como o teste deve ser executado, quais as variáveis envolvidas e quais resultados devem ser apresentados.

Na literatura, as pesquisas que abordam a seleção de casos de teste tendem a utilizar técnicas de aprendizagem de máquina como redes neurais ((JAHAN, 2019), (SPIEKER, 2017), (AL-SABBAGH, 2019), (SPIEKER, 2017), (WANG, 2011), (ABDELKARIM; ELADAWI, 2022), (ABDELKARIM; ELADAWI, 2022)), classificadores ((LACHMANN, 2016), (TONELLA, 2006), (CHEN, 2011), (HEMMATI, 2010)), agrupamento ((ROSERO, 2017), (COTRONEO, 2013)), entre outros, porém são deixados de lado os sistemas de recomendação. Os sistemas de recomendação são geralmente utilizados por *e-commerce* para recomendar a um determinado usuário uma lista de itens aos quais outros usuários com perfil parecido tiveram alguma interação (VINISKI, 2021). Utilizando sistemas de recomendação, podemos gerar uma lista de possíveis testes a serem realizados após uma determinada falha ocorrer, utilizando as informações de outras falhas que já ocorreram anteriormente, e foram testadas por algum teste.

Os sistemas de recomendação podem ser construídos de várias formas com base no tipo de informação que se tem disponível, sendo os principais o Sistema de Recomendação por Filtragem Colaborativa (que utiliza as relações entre os usuários) e o Sistema de Recomendação baseado em Conteúdo (que utiliza as informações dos itens). Além disso, é possível realizar a combinação dos dois tipos, o que é chamado de Sistema Híbrido de Recomendação, realizando por exemplo, uma inicialização de pesos por meio das informações dos itens, seguida da atualização desses pesos utilizando os dados das interações dos usuários com os respectivos itens. Os pesos são de maneira resumida vetores, que armazenam o conhecimento atrelado àquele item. Essa combinação só é possível quando os dados disponíveis possuem características presentes em ambos os sistemas de recomendação que formam o sistema híbrido. A descrição de cada tipo de sistema de recomendação está disponível na Subseção 2.2.4.

1.1 Definição do Problema

A execução de testes de regressão pode se tornar um gargalo durante o desenvolvimento de software em um ambiente de IC, devido ao fato de que quanto maior o sistema, mais testes devem ser realizados. Porém, esse problema se agrava ainda mais quando os testes disponíveis não possuem *scripts* para execução automatizada, fazendo com que o tempo que cada teste necessita para ser executado durante um teste de regressão aumente exponencialmente, tanto por conta da habilidade do testador quanto devido ao cronograma dele (KLAMMER; RAMLER, 2017).

Como forma de agilizar o processo de IC, os testes de regressão são selecionados com o objetivo de não executar todos os testes existentes (YU; WANG, 2018), porém os critérios de seleção em testes manuais podem prejudicar a qualidade do processo de teste. A probabilidade de um teste apresentar falhas é o critério mais comum de ser utilizado na seleção de casos de teste manuais, porém não garante que testes que apresentaram poucas falhas, ou que ainda não apresentaram falhas sejam selecionados, por exemplo, testes novos (TONELLA, 2006). Outra forma de selecionar testes manuais é por meio das informações categoriais do teste, como por exemplo: equipe responsável, componente testado e assim por diante, porém dessa forma, podem ocorrer casos em que testes que não precisariam ser executados, sejam selecionados, devido principalmente ao fato de que não há ligação entre o teste manual e a falha descrita (GLIGORIC, 2014).

Esta pesquisa foi realizada usando dados reais de uma empresa do ramo de tecnologia que atua na área de geração, transmissão e distribuição de energia elétrica, desenvolvendo software de gestão e controle. Ao analisar os relatórios de testes de regressão, foi constatado que a seleção de casos de testes manuais é realizada com base na expertise do testador, que se baseia no histórico de falhas e as informações categoriais (time e componente),

os quais são filtrados com base na experiência do próprio desenvolvedor, que atua como testador. Esta forma de seleção também ocorre durante a validação de correção de falha, que é o momento em que uma possível correção é desenvolvida e testada para garantir que as alterações feitas resolveram a falha e não danificaram outro ponto do software.

Com base no processo de IC, há a necessidade da atuação de um seletor de testes durante a validação de correção de uma falha, onde uma integração só será realizada após ter uma garantia de que esta integração não irá gerar problemas para outros pontos do sistema. Durante uma regressão também há a necessidade de selecionar os casos de testes manuais de forma mais precisa, incluindo testes que não seriam selecionados normalmente, mas que possuem relação com a falha que foi corrigida, e deixando de selecionar testes que não sejam relacionados àquelas falhas, independente do time e componente ao qual o teste pertence.

Para que esta seleção seja realizada, é necessário primeiramente realizar o processamento das informações, as quais são apresentadas em linguagem inglesa técnica (nas documentações das *issues*). Dentro das técnicas de extração de informações textuais, as mais robustas e recentes, utilizam de arquiteturas do tipo *Transformer*, proposto por (VASWANI, 2017), porém, as mais famosas como o BERT do Google (DEVLIN, 2019) e o GPT (ACHIAM, 2023), foram treinadas utilizando livros e textos em linguagem natural inglesa, fazendo com que o aproveitamento da técnica em textos técnicos seja prejudicada, devido ao fato de textos técnicos utilizarem termos próximos a linguagem natural, mas com significados diferentes dentro do contexto. É necessário também selecionar os testes que nunca apresentaram falhas, ou seja, testes que não têm ligação com falhas, pois estes podem ser testes importantes a serem executados após uma determinada falha ser resolvida.

Além disso, é necessário definir como será criado o modelo de recomendação, que necessita da utilização das relações entre os itens e das informações dos itens. É importante tratar também os problemas comuns em Sistemas de Recomendação, como o *Coldstart*, que é um problema que ocorre no momento em que um novo item é adicionado ao sistema.

1.2 Objetivo Principal

O objetivo principal deste trabalho é implementar um sistema de recomendação que realiza a seleção de casos de testes manuais, utilizando técnicas de processamento de linguagem natural para processar informações textuais de dados históricos, documentações de testes e documentações de falhas.

Os objetivos específicos do trabalho são:

- Gerar uma lista de recomendação que apresente testes relevantes para o testador.

- Recomendar testes que não seriam recomendados em seleções baseadas na ocorrência de falha.

- Propor um modelo incremental que possa sempre ser atualizado com novos testes e falhas.

- Gerar recomendações para qualquer falha independente de sua origem ou descrição.

- Selecionar o algoritmo de recomendação que melhor se adapta aos dados.

- Realizar comparações entre diferentes modelos de recomendação.

- Realizar testes com o usuário final do sistema.

1.3 Justificativa de Pesquisa

Os testes manuais possuem por característica uma descrição de passos que direciona o testador a executar ações na interface do sistema, porém, essa informação nos algoritmos de seleção atuais não é aproveitada, devido ao fato de que a seleção ocorre com base na ocorrência de falhas. Por esse motivo, o objetivo principal da pesquisa é utilizar um sistema de recomendação que considere os dados textuais tanto dos testes quanto das falhas. Utilizando dados textuais dos testes e das falhas é possível realizar a recomendação de testes que nunca apresentaram falhas, e até mesmo recomendar testes a falhas novas que não tem ligação alguma com testes da base.

Os dados textuais das falhas e dos testes utilizados na pesquisa não são dados em linguagem natural, ou seja, não seguem um padrão linguístico. Os dados documentais dos testes e falhas, são escritos em linguagem técnica, que apesar de estar próxima a língua inglesa, utiliza além de termos técnicos, trechos de código fonte, caminhos de arquivos do repositório do projeto, entre outras descrições as quais fazem com que o *transformer* tente quebrar um termo em N palavras, sendo que o termo não deveria ser quebrado. O mesmo ocorre para os dados textuais de código fonte, que apesar de individualmente seguirem regras de linguagem, quando em conjunto apresentam termos confusos para o *transformer*.

Além das peculiaridades das falhas e dos testes, é necessário que o sistema de recomendação possa ser atualizado com novas informações, para que o sistema não fique defasado após novas falhas serem encontradas ou novos testes serem desenvolvidos. Além disso, ele deve recomendar testes independente da falha informada, recomendando os testes mais próximos a falha em questão.

1.4 Questões de Pesquisa

Durante o desenvolvimento do trabalho, procuramos responder algumas questões pertinentes a recomendação de testes utilizando sistemas de recomendação.

1. Qual método de inicialização de pesos do modelo apresenta melhores resultados para diferentes tipos de seleção de testes?
2. Qual forma de recomendação é mais adequada para realizar a seleção de teste?
3. Como recomendar testes que nunca apresentaram falhas ou nunca tiveram ligação com uma falha?

1.5 Hipóteses de Pesquisa

Além de responder as questões listadas anteriormente, esta pesquisa visou investigar algumas hipóteses:

H01 (Hipótese Nula 01): O uso dos dados textuais descritivos das falhas não melhora a taxa de acerto na recomendação de testes em um sistema de recomendação.

HA1 (Hipótese Alternativa 01): O uso dos dados textuais descritivos das falhas melhora a taxa de acerto na recomendação de testes em um sistema de recomendação.

H02 (Hipótese Nula 02): Não é possível recomendar corretamente testes que nunca apresentaram falhas ou que não tenham ligação alguma com falhas.

HA2 (Hipótese Alternativa 02): É possível recomendar corretamente testes que nunca apresentaram falhas ou que não tenham ligação alguma com falhas.

H03 (Hipótese Nula 03): A inicialização de pesos de um sistema de recomendação, utilizando *transformers* nos dados textuais de falhas e testes para geração de vetor de *embeddings*, não é uma possível alternativa para contornar problemas relacionados a *ColdStart*, inicializando os pesos com os próprios embeddings.

HA3 (Hipótese Alternativa 03): A inicialização de pesos de um sistema de recomendação, utilizando *transformers* nos dados textuais de falhas e testes para geração de vetor de *embeddings*, é uma possível alternativa para contornar problemas relacionados a *ColdStart*, inicializando os pesos com os próprios embeddings.

1.6 Contribuições Científicas

Como contribuição científica o trabalho apresenta um sistema de recomendação híbrido para recomendação de testes de software, que utiliza os dados textuais das falhas e

dos testes para a inicialização dos pesos. Além da utilização de um sistema de recomendação para a geração da lista de testes a serem executados para uma determinada falha, a pesquisa contribui apresentando um cenário em que a presença do problema de *ColdStart* é muito presente, sendo contornado por meio da utilização de dados textuais para a inicialização dos pesos.

1.7 Agradecimentos

Esta pesquisa conta com apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) sob o número de Processo 142187/2019-4. Além disso, este projeto se caracteriza como um Doutorado Acadêmico em Inovação (DAI) e faz parte da cooperação técnica entre a Pontifícia Universidade Católica do Paraná (PUCPR) e a empresa Siemens Brasil Ltda.

1.8 Estrutura do Trabalho

O trabalho é organizado da seguinte forma: No Capítulo 1 é dada uma introdução do projeto, definindo objetivos, elencando questões de pesquisa e levantando as hipóteses que foram investigadas; O Capítulo 2 apresenta a fundamentação teórica, composta por explicações sobre temas específicos utilizados durante a pesquisa; O Capítulo 3 descreve o estado da arte sobre o tema de seleção e priorização de casos de teste utilizando algoritmos de aprendizagem de máquina e também o estado da arte sobre o tema de sistemas de recomendação, bem como os critérios de busca utilizados para levantar os estados da arte; Nos Capítulos 4 e 5 são apresentados respectivamente os procedimentos metodológicos e o método proposto para a pesquisa; Em seguida, é apresentado no Capítulo 6 os resultados dos experimentos; e por fim a conclusão é apresentada no Capítulo 7.

2 Fundamentação teórica

Para melhor compreensão dos temas abordados em cada passo deste documento, são apresentados a seguir os conceitos dos temas mais relevantes da pesquisa. Os conceitos são separados em conceitos de teste de software (Seção 2.1) e conceitos de aprendizagem de máquina (Seção 2.2).

2.1 Teste de Software

Nesta Seção são apresentadas as definições dos principais conceitos de testes utilizados no projeto, os quais são descritos no livro de Glenford J. Myers (MYERS, 2004) e na norma 829 da IEEE (IEEE, 1983). A Subseção 2.1.1 descreve o conceito básico de teste funcional, em que o teste de regressão (Subseção 2.1.2), o teste unitário (Subseção 2.1.3) e o teste de integração (Subseção 2.1.4) são exemplos. São apresentados também os conceitos de plano de teste (Subseção 2.1.5), caso de teste (Subseção 2.1.6) e Integração Contínua (Subseção 2.1.7)

2.1.1 Teste Funcional

É o processo que visa garantir a qualidade do software com base na especificação externa, ou seja, o comportamento do programa do ponto de vista do usuário final. Geralmente os testes funcionais são atividades de caixa preta, ou seja, as funcionalidades são testadas com base na entrada e na saída do software (MYERS, 2004). O teste de regressão (descrito na Subseção 2.1.2) é um tipo de teste funcional.

2.1.2 Teste de Regressão

O teste de regressão é um tipo de teste funcional que tem como principal característica reexecutar testes anteriores, com o objetivo de garantir que as funcionalidades já existentes nos sistemas, não apresentem problemas após a implementação de uma nova funcionalidade, ou após isso, a correção de algum problema previamente reportado (IEEE, 1983). Os testes de regressão são geralmente executados a partir de um plano de teste do sistema (Seção 2.1.5), o qual possui uma lista de testes documentados que geralmente são executados de forma automática (com intervalo pré-determinado) devido a grande massa de testes que são integrados a um determinado plano de teste. Os testes de regressão também podem ser executados de forma manual, comumente ocorre quando: um teste T1 é executado, T1 tem como saída uma falha F1, a falha F1 é corrigida, o teste T1 é

executado novamente para se certificar de que F1 foi realmente corrigida (o teste T1 deve seguir a definição do caso de teste descrito na Subseção 2.1.6).

2.1.3 Teste Unitário

O teste unitário é teste de nível mais baixo dentro dos testes realizados em um software. Seu objetivo é testar as unidades básicas de um software, podendo estas serem chamadas de "unidade", "módulos" ou "componentes" (LUO, 2001), a fim de eliminar defeitos de construção antes que o código seja integrado ao sistema. O teste unitário sozinho não garante a funcionalidade de um software, mas pode garantir que blocos do software funcionem independentemente uns dos outros.

2.1.4 Teste de Integração

O teste de integração visa garantir que dois ou mais componentes, quando atuam em conjunto, não apresentem defeitos. O teste de integração é realizado após o teste de unidade (Subseção 2.1.3), onde as unidades anteriormente testadas são unificadas criando módulos, os quais são testados nesta etapa (IEEE, 1983).

2.1.5 Plano de Teste

O plano de teste (*Test Plan*) é um documento ou modelo sistemático que detalha o fluxo de trabalho durante o processo de teste de hardware ou software. A IEEE 829 (IEEE, 1983), que é a norma que especifica a forma e o conjunto de artefatos que devem ou deveriam estar presentes no teste de software, descreve a estrutura do plano de teste, a qual deve conter: um identificador do plano de teste; uma introdução (que descreve o propósito do plano de teste e referência os documentos associados); os itens de teste (elementos a serem testados); os recursos que serão testados; os recursos que não serão testados; a estratégia ou abordagem que será utilizada; os critérios de aprovação ou reprovação do item; os critérios de suspensão e requisitos de retomada (atividades a serem realizadas antes de reiniciar o teste); os documentos do teste (revisa os documentos do teste que serão entregues); as tarefas necessárias para realizar o teste; as necessidades do ambiente de teste; os responsáveis pelo teste (responsáveis por gerenciar, projetar, preparar, executar, testemunhar, verificar e resolver); as necessidades de pessoal e treinamento; o cronograma de testes; os riscos e contingências; e as aprovações necessárias.

2.1.6 Caso de Teste

É definido como caso de teste o conjunto de condições usadas para teste de software. Tem como objetivo identificar defeitos na estrutura interna do software e garantir que os requisitos do software sejam completamente atendidos. O caso de teste deve especificar os

valores de entrada e os resultados esperados do processamento. Numa situação ideal, no desenvolvimento de casos de teste, se espera encontrar o subconjunto dos casos de teste possíveis com a maior probabilidade de encontrar a maioria dos erros (IEEE, 1983).

Para se criar um caso de teste estruturado é necessário primeiramente descrever de forma clara qual é a sua finalidade ou objetivo do teste, para que caso ele venha a ser utilizado futuramente, seu objetivo seja compreendido não apenas pelo seu criador, como por outros colaboradores que conduzam o teste. Deve-se também definir a pré-condição para que o caso de teste seja executado, descrevendo o estado obrigatório que um sistema deve estar antes da execução do caso de teste. Partindo de um estado inicial, são realizados estímulos nos testes por meio de entradas, as quais devem ser enumeradas e listadas. As entradas podem ser objetos ou campos de interação ou inserção de dados, os quais serão utilizados durante a execução do teste. As ações que o usuário deve realizar durante a execução do teste devem ser listadas, para que o sistema possa cumprir com o que será testado. Os resultados esperados da execução do teste devem ser especificados, podendo incluir respostas negativas (erros ou falhas) ou positivas. Após a execução o teste deve retornar a um estado descrito, o qual permita a execução de outros testes subsequentes.

2.1.7 Integração Contínua

A Integração Contínua é uma abordagem de desenvolvimento de software onde os membros da equipe realizam seu trabalho de forma desacoplada do projeto, e assim que necessário integram suas alterações ao código principal. Geralmente cada desenvolvedor integra seu trabalho diariamente, e até mesmo várias vezes ao dia, ou seja, os membros da equipe trabalham em ramificações do código fonte principal, e assim que necessário, unem essas ramificações ao código principal. Cada integração é verificada por uma compilação automatizada para detecção de erros de integração (FOWLER, 2006). O *Git* é um aliado importante no processo de integração, pois além de realizar a união de versões de código, armazena todas as versões anteriores, para que se necessário, seja possível voltar em versões anteriores do mesmo.

2.2 Aprendizagem de Máquina

Nesta Seção serão apresentados alguns conceitos de aprendizagem de máquina necessários para compreensão dos temas abordados neste projeto. Considerado um subconjunto da Inteligência Artificial (IA), o aprendizado de máquina (*Machine learning*) exibe o "aprendizado" experiencial associado à inteligência humana, ao mesmo tempo em que tem a capacidade de aprender e melhorar suas análises por meio do uso de algoritmos computacionais (HELM, 2020). A Subseção 2.2.1 apresenta uma introdução sobre o mecanismo de atenção utilizado na construção dos *transformers*. Na Subseção 2.2.2 é apresentado o

BERT, um *transformer* que utiliza de tal mecanismo. O SBERT, que é uma modificação do BERT, é apresentado na Subseção 2.2.3. A Subseção 2.2.4 apresenta todos os conceitos relacionados a Sistemas de Recomendação.

2.2.1 Mecanismo de Atenção e Transformers

Proposto por (VASWANI, 2017), o *Attention* ou mecanismo de atenção, é uma arquitetura de rede simples que é o componente fundamental dos *Transformers*, dispensando totalmente a utilização de convoluções, que são comuns em redes neurais. O mecanismo de atenção permite que os modelos capturem relações de dependência entre os *tokens* de um texto, permitindo um processamento baseado em contexto.

O mecanismo de atenção funciona calculando pesos de atenção para cada par de *tokens* em uma sequência. Esses pesos determinam a importância relativa de cada *token* para o *token* que está sendo processado. Em outras palavras, eles indicam quão relevantes são os *tokens* em relação a um *token* específico.

Em um modelo *transformer*, o processo de atenção segue os seguintes passos:

- Preparação: O texto de entrada é dividido em *tokens* (por meio de um tokenizador) e cada *token* é associado a três vetores: um vetor de consulta (Q), um vetor de chave (K) e um vetor de valor (V). Esses vetores são aprendidos durante o treinamento do modelo.

- Cálculo dos valores de consulta, chave e valor: Para cada token na sequência, o modelo calcula um vetor de consulta, um vetor de chave e um vetor de valor. Esses cálculos são feitos aplicando projeções lineares aos *embeddings* originais do *token*.

- Cálculo da matriz de atenção: A matriz de atenção é calculada multiplicando os vetores de consulta pelos vetores de chave transpostos. O resultado é escalado pela raiz quadrada da dimensão dos vetores de chave para evitar que os valores de atenção sejam muito grandes. Em seguida, é aplicada uma função de *softmax* para normalizar os valores e obter pesos de atenção para cada par de *tokens*.

- Cálculo da saída ponderada: A matriz de atenção é multiplicada pelos vetores de valor para obter uma representação ponderada dos valores. Essa representação ponderada é somada e forma a saída do mecanismo de atenção.

- Processamento multi-cabeça: O mecanismo de atenção é geralmente aplicado várias vezes, usando diferentes conjuntos de vetores de consulta, chave e valor em cada iteração. Isso é conhecido como processamento multi-cabeça e ajuda o modelo a capturar diferentes tipos de relações entre os *tokens*.

2.2.2 Bidirectional Encoder Representations from Transformers

O *Bidirectional Encoder Representations from Transformers* (BERT) é um modelo de linguagem baseado em redes *Transformers* (seção 2.2.1). Ele foi projetado para melhorar a compreensão de contexto em modelos de linguagem e foi descrito em (DEVLIN, 2019).

O BERT diferencia-se de modelos anteriores ao processar informações de maneira bidirecional, levando em conta tanto o contexto anterior quanto o contexto subsequente ao prever uma palavra ou token em um texto. Essa abordagem bidirecional permite ao BERT capturar relações semânticas mais ricas em um texto.

A arquitetura do BERT possui duas variantes principais: o BERT-base e o BERT-large. O BERT-base possui 12 camadas de codificador *Transformer*, enquanto o BERT-large possui 24 camadas. Ambas as variantes têm uma quantidade considerável de parâmetros, tornando-as computacionalmente intensivas.

Cada camada de codificador *Transformer* no BERT é composta por dois submódulos principais: a *multi-head attention* e a camada de *feed-forward*. A *multi-head attention* permite que o modelo capture informações de dependência entre *tokens*, calculando pesos de atenção para cada par de *tokens* na sequência. Esses pesos indicam a importância relativa de cada *token* para o *token* em consideração. A camada de *feed-forward* é responsável por aplicar transformações lineares aos dados de entrada e aplicar uma função de ativação não-linear.

O treinamento do BERT ocorre em duas fases principais: pré-treinamento e ajuste fino. Durante o pré-treinamento, o BERT é treinado em grandes quantidades de dados não rotulados. Esses dados podem ser obtidos a partir de textos da web ou de outros recursos. Durante o pré-treinamento, o BERT aprende a prever palavras ou *tokens* mascarados em um contexto. Para isso, durante o treinamento, algumas palavras ou *tokens* são ocultados e o modelo é treinado para recuperá-los corretamente. Além disso, o BERT também é treinado para determinar se duas sentenças são consecutivas ou escolhidas aleatoriamente.

Após o pré-treinamento, ocorre a fase de ajuste fino. Nessa fase, o BERT é adaptado para tarefas específicas de processamento de linguagem natural, como classificação de texto, reconhecimento de entidades nomeadas ou resposta a perguntas. O BERT é treinado novamente utilizando dados rotulados da tarefa específica, ajustando seus parâmetros para se adequar às características e demandas da tarefa.

Uma das principais vantagens do BERT é sua capacidade de capturar relações semânticas complexas em um texto, devido à sua arquitetura de atenção bidirecional. Neste projeto, uma variação do BERT denomidada de Sentence-BERT (Subseção 2.2.3) será utilizada como extratora de características, onde para cada sentença de entrada, a mesma irá retornar um vetor de características com tamanho fixo, baseado em seu pré-treinamento.

2.2.3 Sentence-BERT

O Sentence-BERT (SBERT), proposto em (REIMERS; GUREVYCH, 2019), funciona como uma modificação do modelo BERT, adaptado para gerar representações vetoriais (ou *embeddings*) de sentenças, otimizando tarefas que envolvem a comparação semântica entre textos. O SBERT é baseado no BERT, no entanto, o BERT foi projetado para tarefas de classificação de texto, onde a sentença de entrada é processada junto com as palavras que queremos comparar. Esse processo de concatenação das sentenças é computacionalmente caro quando se trata de comparar várias sentenças entre si. O SBERT resolve esse problema gerando *embeddings* fixos para sentenças de forma independente. Esses *embeddings* são representações numéricas em um espaço vetorial, onde sentenças com significados semelhantes ficam próximas entre si e sentenças com significados diferentes ficam mais distantes.

Para isso, SBERT utiliza o BERT padrão para codificar a sentença. Ao invés de concatenar duas ou mais sentenças no mesmo *input*, SBERT processa cada sentença separadamente. No final da arquitetura BERT, uma camada de *pooling* (normalmente do tipo média ou *max-pooling*) é adicionada para condensar a informação de todos os *tokens* da sentença em um único vetor de tamanho fixo.

O SBERT pode ser usado para várias tarefas de similaridade semântica de sentenças. A comparação entre duas ou mais sentenças se resume a calcular os seus *embeddings* e depois usar uma métrica de distância, como similaridade do cosseno, para medir o quão próximas as sentenças estão no espaço vetorial. Isso é muito mais eficiente do que o método tradicional do BERT, que exigia a concatenação das sentenças e uma nova passagem pelo modelo para cada par de comparações.

De forma resumida, o SBERT é uma adaptação do BERT que foi ajustada para tarefas de comparação de sentenças, atendendo a necessidade da pesquisa comparando informações textuais de falhas e testes.

2.2.4 Sistemas de Recomendação

Sistemas de Recomendação são definidos como uma estratégia de tomada de decisão para usuários em ambientes de informação complexos. Foi inicialmente definido sob a perspectiva do comércio eletrônico como uma ferramenta que auxilia os usuários na busca por meio de registros de conhecimento relacionados ao interesse e preferência dos usuários (ISINKAYE, 2015). Os Sistemas de Recomendação são ferramentas e técnicas de software que fornecem sugestões de itens que provavelmente são de maior interesse para um determinado usuário (RICCI, 2021).

No contexto de Falhas de Testes, os sistemas de recomendação fornecem sugestões de testes que provavelmente são de maior interesse para detectar uma determinada falha.

A Figura 1 apresenta dois testes (T1 e T2) que são recomendados a uma falha F1.

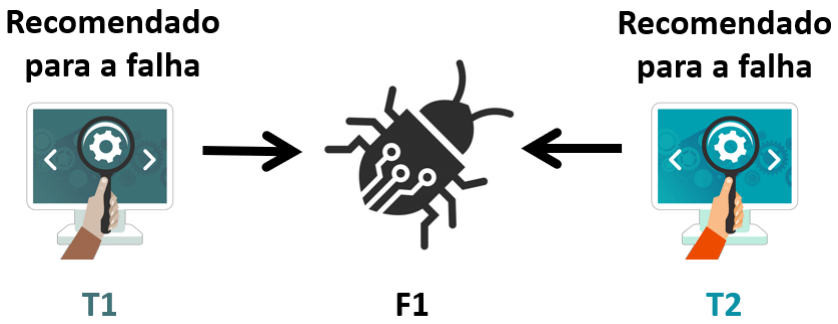


Figura 1 – Exemplo de recomendação dos Testes T1 e T2 para a Falha F1.

Fonte: Figura do autor

Os Sistemas de Recomendação podem ser divididos de acordo com o tipo de algoritmo utilizado, sendo os principais: filtragem colaborativa (Subseção 2.2.4.1), baseada em conteúdo (Subseção 2.2.4.2) e híbrida (Subseção 2.2.4.3). Além dessas, também existem outras abordagens como demográficas, baseadas em utilidade e baseadas em conhecimento, que não serão abordadas nesse trabalho. A Subseção 2.2.4.4 apresenta o problema de *ColdStart* e a Subseção 2.2.4.5 apresenta a Classificação Bayesiana Personalizada para Matriz de Fatoração (BPR-MF).

2.2.4.1 Sistemas de Recomendação Baseado em Filtragem Colaborativa

Um Sistemas de Recomendação baseado em filtragem colaborativa decide quais itens recomendar ao usuário alvo observando as classificações de usuários similares. A similaridade é computada observando apenas as classificações dos usuários. Os usuários são considerados como similares se eles classificam os itens de forma similar. A filtragem colaborativa recomenda ao usuário-alvo itens que são altamente classificados por seus similares (ELAHI, 2016). A filtragem colaborativa é o processo de filtrar ou avaliar itens usando as opiniões de outras pessoas. A filtragem colaborativa tem suas raízes em algo que os humanos fazem há séculos, que é compartilhar opiniões com os outros (SCHAFER, 2007).

No contexto de Testes e Falhas, a filtragem colaborativa é utilizada para recomendar para falhas similares, os testes em que alguma das falhas teve interação e a outra não, ou seja, as interações diferentes entre falhas similares. A Figura 2 apresenta um exemplo com duas falhas (F1 e F2) similares, onde o teste que apenas a falha F2 teve interação será recomendado para a falha F1.

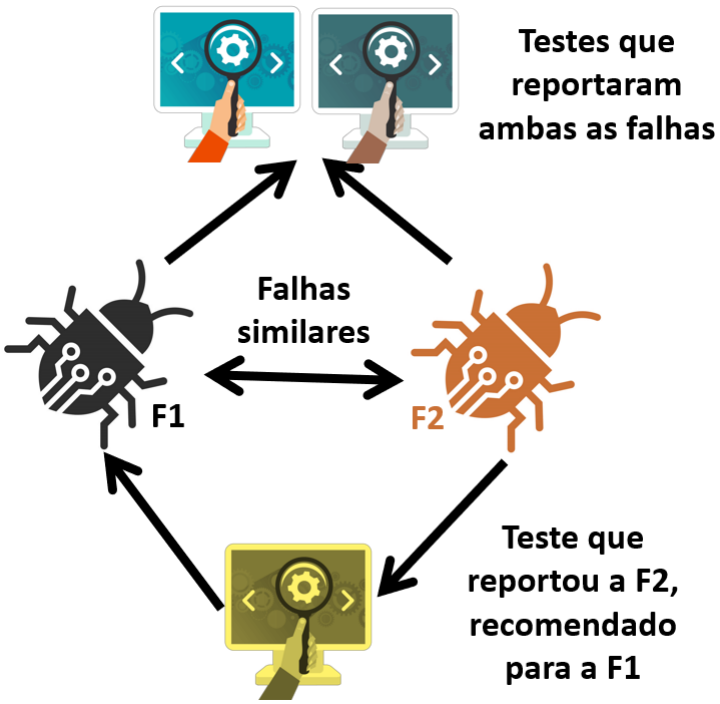


Figura 2 – Exemplo de Recomendação por filtragem colaborativa onde as falhas F1 e F2 são similares.

Fonte: Figura do autor

2.2.4.2 Sistemas de Recomendação Baseado em Conteúdo

Os Sistemas de Recomendação baseado em conteúdo sugerem itens de interesse com base nas características distintivas dos próprios itens. Por exemplo, Sistemas de Recomendação de notícias consideram os termos contidos em artigos de notícias que o usuário leu e gostou, recomendando ao usuário outros artigos de notícias com termos semelhantes (ELAHI, 2016).

Os sistemas que implementam uma abordagem de recomendação baseada em conteúdo analisam um conjunto de documentos e/ou descrições de itens previamente classificados por um usuário e constroem um modelo ou perfil de interesses do usuário com base nas características dos objetos classificados por esse usuário. O processo de recomendação consiste basicamente em comparar os atributos do perfil do usuário com os atributos de um objeto de conteúdo (LOPS, 2011).

No contexto de Testes e Falhas, a recomendação baseada em conteúdo irá recomendar a uma determinada falha, testes similares aos testes que a relataram ou que tiveram alguma interação. A Figura 3 apresenta um exemplo em que para uma falha que foi reportada por um determinado teste, é recomendado um teste similar.

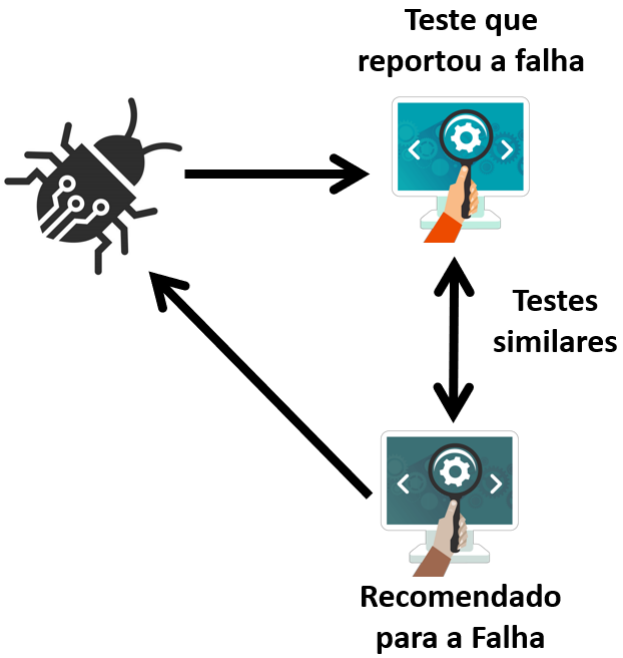


Figura 3 – Exemplo de Recomendação baseada em conteúdo onde testes similares são recomendados para a mesma falha.

Fonte: Figura do autor

2.2.4.3 Sistemas de Recomendação Híbridos

Os sistemas de recomendação híbridos combinam duas ou mais abordagens para lidar com as limitações específicas de uma única solução (ELAHI, 2016). Esse tipo de sistema busca compensar as limitações de cada abordagem individual e maximizar os acertos ao recomendar produtos, filmes, livros, músicas, ou qualquer outro tipo de item. Neste trabalho é utilizado um sistema de recomendação híbrido que combina as abordagens de filtragem colaborativa e baseada em conteúdo, oferecendo recomendações mais precisas e personalizadas. Há diversas formas de realizar a combinação da filtragem colaborativa com a baseada em conteúdo, porém nessa pesquisa a filtragem colaborativa é utilizada como forma de refinar os pesos das falhas e testes, a fim de aproximar os vetores de falhas dos vetores de testes.

A Figura 4 apresenta um exemplo de sistema de recomendação híbrido, onde para uma determinada falha B2, são recomendados testes similares aos testes que B2 já tem relação, e testes aos quais uma falha similar B1 possui ligação e B2 não possui.

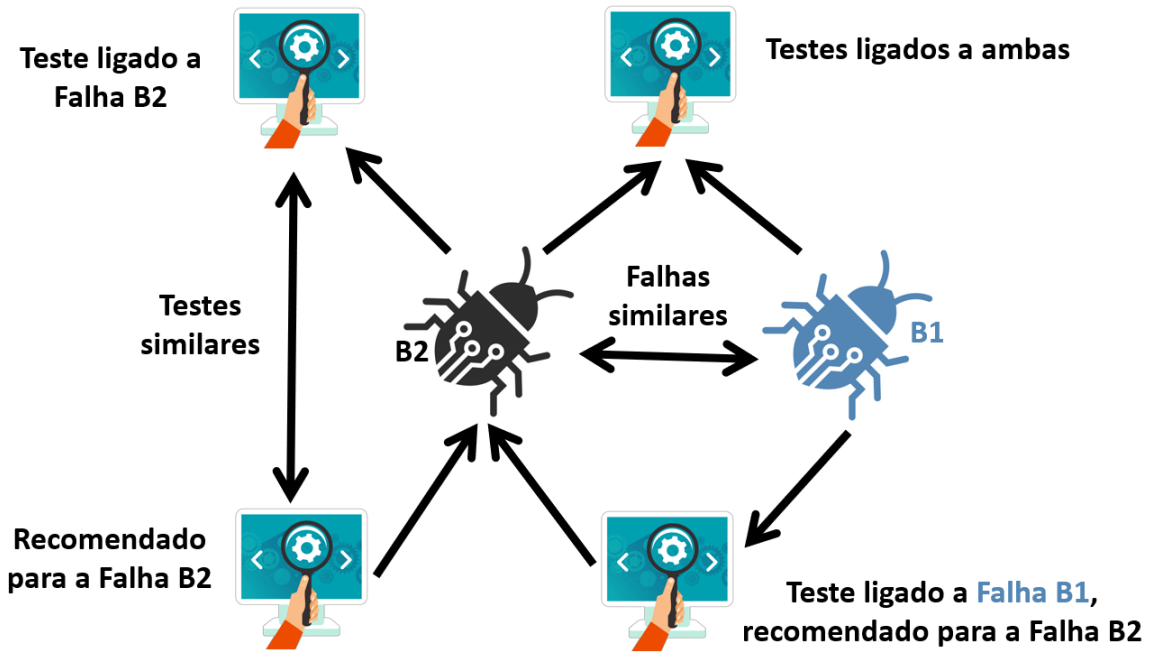


Figura 4 – Exemplo de Recomendação Hibrida onde testes são recomendados a falha B2.

Fonte: Figura do autor

2.2.4.4 *ColdStart*

O *ColdStart* (ou "início a frio") em Sistemas de Recomendação é um problema que ocorre quando o sistema não possui informações suficientes sobre novos usuários ou novos itens para gerar recomendações precisas. Esse é um desafio comum nos sistemas de recomendação, especialmente nas fases iniciais de interação com a plataforma.

O problema de *Coldstart* afeta tanto a filtragem colaborativa quanto a filtragem baseada em conteúdo, embora de formas diferentes. Quando por exemplo um novo usuário entra no sistema, ele não possui um histórico de interações (como filmes assistidos, produtos comprados ou músicas ouvidas), o que impede o sistema de entender suas preferências. Isso é um problema particularmente complicado para sistemas de filtragem colaborativa, já que eles dependem de um conjunto de preferências do usuário para compará-lo com outros usuários semelhantes e gerar recomendações.

O mesmo problema de *ColdStart* também ocorre quando novos itens são adicionados ao sistema. Isso afeta especialmente sistemas de filtragem colaborativa baseada em itens, que precisam de avaliações ou interações prévias para recomendar um item a outros usuários.

As Filtragens Baseadas em Conteúdo lidam melhor com o problema de *ColdStart* de itens, porque ele se baseia nas características do item (como gênero, palavras-chave, descrição) para recomendar itens semelhantes aos que o usuário já consumiu. No entanto, para novos usuários, a filtragem baseada em conteúdo ainda sofre, já que não há dados

suficientes sobre as preferências individuais do usuário.

No contexto de Falhas e Testes o Coldstart pode ocorrer quando uma determinada falha não possui ligações com testes, ou quando testes não possuem ligações com falhas. A Figura 5 apresenta um exemplo onde a Falha F4 não possui ligação com testes e o Teste T4 não possui ligação com falhas.

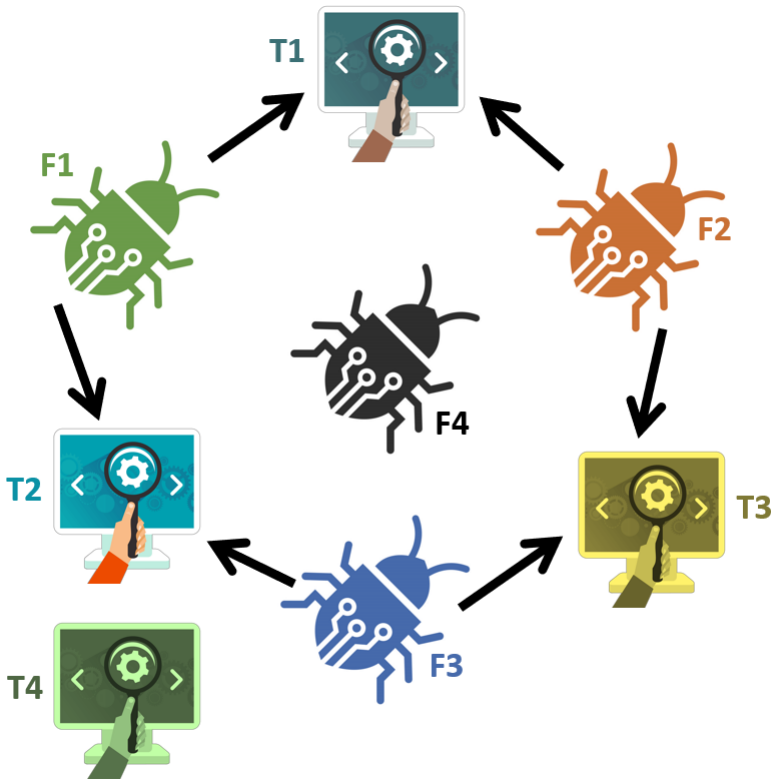


Figura 5 – Exemplo de *ColdStart*. Falha F4 não tem ligação com testes e Teste T4 não tem ligação com falhas.

Fonte: Figura do autor

2.2.4.5 Classificação Bayesiana Personalizada para Matriz de Fatoração (BPR-MF)

Definido em (RENDLE, 2012), o *Bayesian Personalized Ranking* (BPR) é uma técnica de aprendizado utilizada para otimizar modelos de recomendação baseados em Matrix Factorization (MF), com o objetivo de gerar recomendações personalizadas. O BPR é especialmente útil para lidar com tarefas de recomendação onde o *feedback* do usuário é implícito (como cliques, visualizações, curtidas), em vez de explícito (como classificações numéricas).

Em Sistemas de Recomendação, frequentemente lidamos com *feedback* implícito, onde sabemos apenas quais itens um usuário interagiu, mas não sabemos explicitamente quais eles não gostam. Isso cria um problema, já que não há um rótulo claro de "negativo". O BPR foi desenvolvido para resolver esse desafio, focando em comparar pares de itens para modelar preferências. O BPR trabalha com a premissa de que os itens com os quais

um usuário interagiu (positivos) devem ser mais preferidos do que os itens com os quais ele não interagiu (negativos). No contexto de Testes e Falhas, consideramos como interação qualquer ligação que um teste e uma falha possuam, como por exemplo a informação de que um Teste T1 relatou uma Falha F1, logo T1 e T1 possuem uma ligação positiva. O BPR-MF no contexto de Testes e Falhas pode ser aplicado por meio das seguintes fórmulas matemáticas:

$$\hat{r}_{bp} \leftarrow \vec{A}_b \cdot \vec{B}_p \tag{2.1}$$

$$\hat{r}_{bn} \leftarrow \vec{A}_b \cdot \vec{B}_n \tag{2.2}$$

$$\hat{r}_{bpn} \leftarrow (\hat{r}_{bp} - \hat{r}_{bn}) \tag{2.3}$$

$$sigmoid = \frac{e^{-\hat{r}_{bpn}}}{1 + e^{-\hat{r}_{bpn}}} \tag{2.4}$$

$$\vec{A}_b \leftarrow \vec{A}_b + n \times \left(sigmoid \times (B_p - B_n) + \lambda \times \vec{A}_b\right) \tag{2.5}$$

$$\vec{B}_p \leftarrow \vec{B}_p + n \times \left(sigmoid \times (A_b) + \lambda \times \vec{B}_p\right) \tag{2.6}$$

$$\vec{B}_n \leftarrow \vec{B}_n + n \times \left(sigmoid \times (-A_b) + \lambda \times \vec{B}_n\right), \tag{2.7}$$

Onde $\vec{A}_b$ representa o vetor de pesos da Falha, $\vec{B}_p$ representa o vetor de pesos do Teste positivo e $\vec{B}_n$ representa o vetor de pesos do Teste Negativo atual. Após realizar o cálculo do produto vetorial entre Falha e Teste Positivo (representado por $\hat{r}_{bp}$ na Equação 2.1) e entre Falha e Teste Negativo (representado por $\hat{r}_{bn}$ na Equação 2.2), é calculada a diferença entre os produtos vetoriais (representado por $\hat{r}_{bpn}$ na Equação 2.3) e então é realizado o cálculo do *sigmoid* utilizando a Equação 2.4. Por meio do valor de *sigmoid*, da taxa de aprendizagem (n) e da taxa de regularização (λ), são obtidos os vetores de pesos atualizados para a Falha, o Teste Positivo e o Teste Negativo em questão (Equações 2.5, 2.6 e 2.7 respectivamente).

A função *sigmoid* é utilizada como função de perda (*loss function*) padrão para essa pesquisa, porém é possível utilizar outras funções como a *softmax* ou a *hinge loss*.

3 Trabalhos Relacionados

Neste capítulo apresentam-se os trabalhos mais relevantes e atuais relacionados a esta pesquisa. Primeiramente foi realizada uma pesquisa prévia buscando por trabalhos que abordam os temas de seleção de casos de testes utilizando sistemas de recomendação, resultado no levantamento de dois documentos. Duas novas pesquisas foram realizadas utilizando os temas isolados, onde a primeira referente ao tema de seleção de casos de testes, o qual pode ser subdividido em: minimização de casos de teste (onde o objetivo é reduzir a quantidade de testes a serem realizados eliminando redundâncias), seleção de casos de teste (que tem o objetivo de selecionar apenas os testes relevantes para serem executados) e priorização de casos de teste (no qual é realizado uma ordenação nos testes de forma a priorizar alguns testes com base em algum critério). Já na segunda pesquisa, foram levantados trabalhos relacionados a sistemas de recomendação que utilizam aprendizado incremental, os quais foram selecionados com base na proximidade com a pesquisa.

Os trabalhos relacionados levantados encontrados a partir dos critérios de busca apresentados na Seção 3.1, sendo descritos e comparados na Seção 3.2.

3.1 Critérios de Busca

A busca dos trabalhos relacionados a pesquisa é uma etapa de grande valia para o projeto, podendo impactar diretamente na qualidade e na visibilidade do projeto. Para que tenhamos as informações mais atuais e relevantes, inicialmente foi realizada uma revisão da literatura, buscando trabalhos relacionados ao tema e listando suas principais características bem como seus resultados e conclusões. Para este projeto foram realizadas buscas em três diferentes bases de artigos e periódicos: ACM[1] , IEEE[2] e *Science Direct*[3] . Para todas as bases foram utilizadas strings de busca com o objetivo de encontrar artigos em língua inglesa contendo informações sobre o tema de pesquisa. A busca foi limitada a área de computação com datas de publicação superiores a 2001 (últimos 24 anos).

A pesquisa inicial que une os temas de casos de teste e sistemas de recomendação foi realizada nas bases de artigos e periódicos por meio das strings de busca apresentadas nos Quadros 3.1, 3.2 e 3.3.

A *string* de busca geral ACM (Quadro 3.1) retornou um documento apenas, o qual foi considerado como relevante para a pesquisa por meio da análise de *abstract*. A *string* de busca geral IEEE (Quadro 3.2) resultou um documento, o qual foi considerado como

[1] ACM Digital Library. <https://dl.acm.org/>. Acesso: 30-12-2024.
[2] IEEE Xplore. <https://ieeexplore.ieee.org/Xplore/home.jsp>. Acesso: 30-12-2024.
[3] SCIENCEDIRECT.COM. <https://www.sciencedirect.com/>. Acesso: 30-12-2024.

```
[Title: "test case"] AND [[Title: "recommender system"] OR
[Title: "recommendation system"]]
```

Quadro 3.1 – *String* de Busca Geral ACM

```
(("Document Title":"Recommendation System"
OR "Document Title":"Recommender System")
AND "Document Title":"test case" )
```

Quadro 3.2 – *String* de Busca Geral IEEE

```
Title, abstract, keywords: "test case" AND
("Recommendation System" OR "Recommender System")
```

Quadro 3.3 – *String* de Busca Geral *Science Direct*

relevante para a pesquisa por meio da analise de *abstract*. A *string* de busca geral *Science Direct* (Quadro 3.3) resultou cinco documentos, os quais não foram selecionados para a pesquisa devido ao conteúdo não ter relação com testes de software.

Como dois documentos foram levantados, foram realizadas duas novas pesquisas com base nos temas abordados de forma isolada, a primeira foi realizada em todas as bases e foram utilizadas strings de busca com o objetivo de encontrar artigos em língua inglesa contendo informações sobre Priorização de Casos de Teste, Seleção de Casos de Teste ou Ranqueamento de Casos de Teste, utilizando de Aprendizagem de Máquina. A busca foi limitada a área de computação com datas de publicação superiores a 2001 (últimos 24 anos). Além das bases já mencionadas, foi realizada uma busca complementar na base Scopus[4] .

Para a busca por documentos referentes a seleção de casos de teste, na base ACM, após aplicar a *string* de busca apresentada no Quadro 3.4, foram retornados 50 trabalhos, dos quais foram selecionados onze com base no título e *abstract*.

Na base IEEE, após aplicar a *string* de busca apresentada no Quadro 3.5 limitando a busca para os últimos 24 anos, resultou em 536 trabalhos, dos quais foram selecionados 21 com base no título e *abstract*.

Na base Science Direct, após aplicar a *string* de busca apresentada no Quadro 3.6, resultou em 34 trabalhos, dos quais foi selecionado um com base no título e *abstract*.

[4] SCOPUS preview. <https://www.scopus.com/>. Acesso: 30-08-2024.

```
[Title: test case] AND [[Title: priorization] OR [Title: rank] OR
[Title: selection]] AND [Abstract: machine learning]
```

Quadro 3.4 – Primeira *String* de Busca ACM

```
("All Metadata":test case) AND (("All Metadata":prioritization)OR
("All Metadata":selection)OR("All Metadata":rank)) AND
("All Metadata":Machine Learning)
```

Quadro 3.5 – Primeira *String* de Busca IEE

```
"test case" AND(rank OR priorization OR selection)
AND "machine learning"
```

Quadro 3.6 – Primeira *String* de Busca Science Direct

```
TITLE-ABS-KEY ("test case")AND(TITLE-ABS-KEY ("prioritization")OR
TITLE-ABS-KEY ("selection" )OR TITLE-ABS-KEY ("rank") )AND
TITLE-ABS-KEY ("Machine Learning")AND PUBYEAR > 2001 AND
( LIMIT-TO ( SUBJAREA ,  "COMP" ) )
```

Quadro 3.7 – Primeira *String* de Busca Scopus

Na base Scopus, após aplicar a *string* de busca do Quadro 3.7 limitando a busca para os últimos 24 anos e área de computação, resultou em 190 trabalhos, dos quais foram selecionados 30 com base no título e *abstract*.

Após as coletas, o total de trabalhos encontrados sem verificação de duplicatas foi de 63. Após examinar os 63 documentos, nove trabalhos foram eliminados por duplicidade, ficando assim 55 documentos com possíveis contribuições para a pesquisa. Em seguida foi realizada a leitura de cada documento como última seleção, onde foram selecionados 27 documentos relevantes a pesquisa. A Figura 6 apresenta a relação de quantidade de documentos por ano de publicação.

Figura 6 – Trabalhos selecionados vs ano de publicação com tema Casos de teste

Fonte: Figura do autor

Já a segunda foi realizada em todas as bases e foram utilizadas strings de busca

```
[[Title: "recommendation system"] OR [Title: "recommender system"]]
AND [[Abstract: incremental] OR [Abstract: adaptative]
OR [Abstract: online] OR [Abstract: continual]]
AND [[Abstract: text] OR [Abstract: textual]]
```

Quadro 3.8 – Segunda *String* de Busca ACM

```
((("Document Title":"Recommendation System" OR
"Document Title":"Recommender System") AND
("Abstract": incremental OR "Abstract": adaptative OR
"Abstract": online OR "Abstract": continual) AND
("Abstract": text OR "Abstract": textual)))
```

Quadro 3.9 – Segunda *String* de Busca IEEE

```
Title, abstract, keywords: ("Recommendation System" OR
"Recommender System") and (incremental OR adaptative OR
online OR continual) AND (text OR textual))
```

Quadro 3.10 – Segunda *String* de Busca *Science Direct*

com o objetivo de encontrar artigos em língua inglesa contendo informações sobre sistemas de recomendação incrementais, adaptativos, online, contínuos que trabalhem com dados textuais. A busca foi limitada a área de computação com datas de publicação superiores a 2001 (últimos 24 anos). Os Quadros 3.8, 3.9 e 3.10 apresentam as strings aplicadas em cada base.

A coleta na base ACM, a partir da *string* apresentada no Quadro 3.8 resultou em 18 documentos os quais foram analisados por título e *abstract*. Após a filtragem foram selecionados dois documentos como relevantes para a pesquisa.

A coleta na base IEEE, a partir da *string* apresentada no Quadro 3.9 resultou em 36 documentos os quais foram analisados por título e *abstract*. Após a filtragem foram selecionados quatro documentos como relevantes para a pesquisa.

A coleta na base *Science Direct*, a partir da *string* apresentada no Quadro 3.10 resultou em 87 documentos os quais foram analisados por título e *abstract*. Após a filtragem foi selecionado um documento como relevante para a pesquisa.

Além dos sete documentos levantados, a partir das *strings* de busca, foram selecionados mais quatro documentos relevantes para a pesquisa com base nas referências de outros documentos, totalizando assim onze documentos relevantes para a pesquisa. A Figura 7 apresenta a relação de quantidade de documentos por ano de publicação.

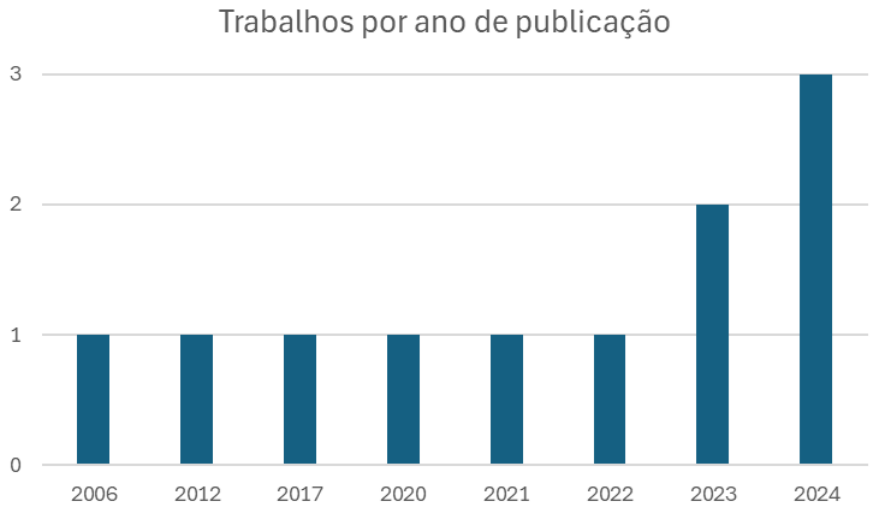


Figura 7 – Trabalhos selecionados vs ano de publicação com tema Recomendação

Fonte: Figura do Autor

3.2 Estado da Arte

Apresentando um Sistema de Recomendação de filtragem colaborativa para priorização de casos de teste em aplicativos da Web, (AZIZI; DO, 2018) em seu trabalho cria um sistema de recomendação que usa três conjuntos de dados (cobertura de código, histórico de alterações e sessões de usuário) para produzir uma lista dos componentes mais arriscados de um sistema para teste de regressão. O modelo de recomendação é treinado utilizando as interações entre componentes e usuários, combinados com as informações sobre ocorrência de falhas nesses componentes. O autor em seu trabalho também cita o chamado "problema de itens novos", onde o sistema não consegue recomendar componentes novos para serem testados, pois não há interações com usuários registradas.

Também trabalhando no contexto de teste e Sistemas de Recomendação, (AZIZI, 2021) propôs um sistema de recomendação baseado em *tags* para priorização de casos de teste de regressão. Para atingir esse objetivo, o autor propõe uma técnica de priorização automatizada que utiliza duas fontes de informação, informações de mudança de programa e código-fonte do programa. Os *scripts* de testes automatizados e os códigos fontes que tem ligação entre si são utilizados para geração de *tags*, e essas *tags* são recomendadas com base em alterações de código-fonte. Apesar de apresentar bons resultados, a recomendação realizada é válida apenas para testes unitários a nível de código, pois o processo de geração de *tags* realiza a varredura no código-fonte e nos *scripts* que serão executados.

Apesar de realizar recomendações de casos de teste, em ambos os trabalhos os testes abordados são testes automatizados a nível de código, os quais não dependem de um testador para serem executados. Também é necessário que o Sistema de Recomendação realize mapeamentos em código-fonte, o que pode ser inviável em grandes sistemas que sofrem alterações frequentes. A seguir, são apresentados os documentos relacionamentos a

casos de teste, os quais complementam a fundamentação.

No trabalho de (TONELLA, 2006) o autor propõe uma técnica de priorização de casos de teste que utiliza o conhecimento do usuário por meio do algoritmo de aprendizado de máquina Case-Based Ranking (CBR). O CBR extrai apenas informações de prioridade relativa do usuário, na forma de comparações de casos de teste em pares. Os resultados preliminares desse estudo de caso indicam que o CBR supera as abordagens anteriores (*Randon*, *Statement coverage - Stmt_cov*, e *Additional statement coverage - Add_stmt_cov*) ficando muito próximo da solução ideal (*Optimal Solution*). A abordagem *Stmt_cov* ordena os casos de teste com base no número de instruções coberta, se baseando na ideia de que quanto maior o número de declarações, maior a probabilidade de aparecerem falhas. Já a *Add_stmt_cov* ordena os casos de teste seguindo a regra que quando todas as instruções exercidas por algum caso de teste são cobertas, a cobertura alcançada pelos casos de teste anteriores é redefinida para zero e a priorização reinicia como no início. A priorização *Randon* ordena os casos de teste de forma aleatória.

Por sua vez (HEMMATI, 2010) propõe uma técnica de seleção de casos de teste baseada em similaridade para seleção de casos de testes em máquina de estados, utilizando gatilhos e guardas em transições nas máquinas de estados e também um algoritmo de seleção baseado em algoritmo genético. Após os testes a técnica se provou mais eficiente em detectar falhas reais, comparada com as alternativas pré-existentes. Por se tratar de um estudo industrial, a pesquisa não é replicável.

Propondo uma abordagem online de seleção de testes funcionais, o trabalho (CHANG, 2010) realiza uma seleção de testes funcionais com base na detecção de novidades, priorizando os testes funcionais mais novos para melhorar a cobertura do conjunto de programas de teste disponíveis. Para detectar a similaridade entre programas é utilizado um esquema baseado em gráfico, o qual realiza o mapeamento da codificação para obter um conjunto de vetores de características, os quais são fornecidos para o classificador SVM (Support Vector Machine). O SVM tem por sua vez o objetivo de identificar os novos testes que serão simulados. Como forma de avaliação da abordagem, foram realizados experimentos utilizando a base pública Plasma - most MIPS I[5] , onde a abordagem proposta foi comparada com outras técnicas similares já existentes mostrando maior eficácia.

O trabalho realizado por (WANG, 2011) cria um *framework* para priorizar testes com base nas falhas encontradas, para realizar os experimentos ele inseriu falhas no código a fim de testar a estrutura. Para a classificação foi utilizado o algoritmo ANN (*Artificial Neural Network*) a qual gerou um plano de testes a partir dos casos de teste. Como base de dados foi utilizada a biblioteca *Intelligent Test Oracle Library* (InTOL).

No trabalho de (CHEN, 2011) é proposta uma abordagem de seleção de testes

[5] PLASMA - most MIPS I(TM) opcodes. <https://opencores.org/projects/plasma>. Acesso: 30-08-2024.

baseada em agrupamento, onde os testes de um mesmo grupo são considerados como tendo comportamentos semelhantes. O agrupamento é realizado de forma não supervisionada, criando pequenos subconjuntos de testes, os quais se espera aproximar da capacidade de detecção de falhas do conjunto total original. Os dados utilizados no trabalho são dois programas (*Flex* e *Space*) construidos em linguagem C e disponíveis no site SIR[6] (*Software-artifact Infrastructure Repository*). Os resultados do trabalho indicam que o método de agrupamento KMeans semi-supervisionado (SSKM) pode melhorar a seleção de testes na maioria dos casos, sendo que a melhor eficácia é encontrada quando os testes que apresentam falhas estão em uma proporção média, ou seja, a quantidade de falhas na base deve ser equivalente ou próxima a quantidade de exemplos sem falhas.

Em (COTRONEO, 2013) é apresentado um método para combinação de técnicas de teste onlines e offlines baseados em aprendizagem de máquina, o qual visa maximizar o número de falhas detectadas com base na maior probabilidade de uma técnica expor uma falha para um software específico. Os programas testados são escritos em linguagem C, sendo eles separados em Offline (*Grep_v3*, *Flex_v3*, *Make_v1*) e Online (*Space*). Como forma de combinação foram utilizados os algoritmos *Naive Bayes*(NB), *Bayes Net*(BNet), *Decision Trees*(DT) e *Logistic*(Log).

No trabalho (ABELE; GÖHNER, 2014) é realizada uma classificação utilizando algoritmo genético, onde o fator fitness é encontrado a partir de lógica fuzzy com o objetivo de encontrar falhas de forma mais eficiente. Como base da dados foram utilizados exemplos de códigos de um jogo, o qual foi desenvolvido em uma competição com duração de 24 horas. O jogo é dividido em cinco módulos (*Underground*, *Obstacle*, *Player*, *Enemy* e *Background*), os quais possuem informações de complexidade e criticidade, as quais foram informadas pelos desenvolvedores. Após os experimentos os foram realizados cálculos de propensão a falhas os quais mostraram que a melhora foi significativa a priorização.

Em (LACHMANN, 2016) é realizado uma priorização de casos de testes em testes de caixa preta, onde não há acesso ao código-fonte. São utilizados como dados os casos de teste, requisitos e falhas, os quais são descritos em linguagem natural, onde são transformados em *features* utilizando de técnicas de NLP. Para realizar o ranqueamento foi utilizado o algoritmo *Ranked Support Vector Machine* (SVM Rank). A técnica de priorização proposta foi comparada com a priorização aleatória e a priorização manual realizada por especialistas, onde a técnica proposta apresentou melhores resultados.

Selecionando os testes com base no acesso ao banco de dados, (ROSERO, 2017) propõe o agrupamento dos casos de teste utilizando de matrizes de similaridade, onde o foco principal é agrupar os testes baseado no acesso ao banco de dados. Como forma de avaliar a eficácia da técnica foram utilizadas as métricas: *suite reduction capability*, *fault*

[6] Software-artifact Infrastructure Repository. <https://sir.csc.ncsu.edu/portal/index.php>. Acesso: 30-08-2024.

detection, *precision* e *F-measure*, as quais foram executadas por meio de dois softwares: *Estafeta* e *Silabo*.

Já focando em testes em Integração Contínua, (SPIEKER, 2017) propõe um método de aprendizado automatizado para seleção e priorização de testes em uma Integração Contínua. O método proposto utiliza de aprendizagem por reforço (RL), o qual seleciona e prioriza os casos de teste de acordo com a duração, a última execução e o histórico de falhas do mesmo. A execução da classificação é realizada em uma janela, onde os novos casos são criados e os mais antigos são deletados. Os resultados mostraram que o ganho no uso do método só começa a aparecer após 60 ciclos de iteração e os experimentos mostraram a necessidade de código-fonte e acessos de programas para melhor desempenho. A base de dados utilizada foi a *Google Shared Dataset of Test Suite Results* (GSDTSR) [7] .

Buscando realizar uma seleção de casos de teste diferente das técnicas habituais, (WANG, 2018) propõe uma técnica de seleção de casos de teste de regressão onde os testes são escolhidos com base nas alterações de refatoração, ou seja, a técnica chamada de *Reks*, pula os testes onde não foram alterados mas que mantém o mesmo comportamento. Os resultados mostraram que o Reks não executaria em média 33% dos testes totais (que seriam executados por outras técnicas de seleção de casos de testes), melhorando assim o custo operacional para realização dos testes de regressão.

O trabalho (AL-SABBAGH, 2019) utiliza de códigos fontes para tentar reduzir a quantidade de testes executados durante um ciclo de iteração dos testes de regressão. Foram utilizados códigos fontes de uma industria parceira, os quais foram transformados em *features* com o processo de *Bag of Words* com fatores de corte, onde palavras reservadas foram tratadas de forma distinta. Após isso foi realizado o processo de agrupamento dos testes de forma a reduzir a quantidade de testes a serem realizados, os algoritmos utilizados foram *Decision Trees*, *Random Forest*, *AdaBoost*, *Multilayer NN* e *CNN*. Os resultados se mostraram viáveis apenas em casos em que a mudança nos códigos é pequena.

O artigo de (JAHAN, 2019) propõe uma abordagem baseada em Rede Neural Artificial (ANN) para priorizar os casos de teste visando encontrar. A abordagem proposta utiliza a combinação de informações de complexidade de casos de teste e informações de modificação de software para detectar falhas críticas. A abordagem foi validada por meio de duas fases de estudo, onde foram avaliadas a confiabilidade do classificador e a capacidade de detecção de falhas da abordagem proposta. Os resultados da abordagem obtiveram 97% de precisão e melhor taxa de detecção de falhas.

Uma extensão do trabalho de (SPIEKER, 2017), o trabalho (LOUSADA; RIBEIRO, 2020) testa a capacidade do algoritmo anteriormente proposto de se adaptar a novos ambientes com dados extraídos de uma instituição financeira. Além dos testes com um

[7] Google Shared Dataset of Test Suite Results. <https://bitbucket.org/HelgeS/atcs-data/src/master/>. Acesso: 30-08-2024.

novo conjunto de dados extraídos de um ambiente de Integração Contínua (CI), foram aplicados Árvores de Decisão (DT) no contexto de aprendizagem por reforço (RL), para representar o espaço de estados. Os resultados indicaram que os algoritmos anteriormente propostos são efetivos em diferentes contextos e que o uso de DT não produz melhores resultados.

Em seu primeiro trabalho, (ROSENBAUER, 2020) utiliza o sistema classificador XCS, que é um exemplar dos sistemas classificadores de aprendizado (LCSs) da aprendizagem por reforço, para realizar a priorização dos casos de teste. Além disso realizaram a combinação do XCS com o conceito de *Experience Replay* (ER) o qual é definido em (LIN, 1992). Em seu segundo trabalho (ROSENBAUER, 2020b) ele apresenta os métodos de combinação do XCS com o ER e compara com o algoritmo de (SPIEKER, 2017), que utiliza de redes neurais combinadas ao RL, onde o algoritmo proposto obtém melhores resultados. Em seu terceiro trabalho (ROSENBAUER, 2020a) é realizada a comparação do algoritmo de (SPIEKER, 2017) com o XCS em sua forma original, mostrando que mesmo sem o uso do ER o mesmo apresenta melhores resultados em comparação com a utilização de redes neurais.

Utilizando a abordagem de priorização de casos de teste, (HUANG, 2021) realiza a priorização de casos de testes de regressão baseado em modelo que chama de *Learn to Rank - LR* (Aprender para Ranquear), o qual utiliza do algoritmo de *Randon Forest* para priorizar os testes a serem selecionados. O Algoritmo proposto foi testado e avaliado por meio de Máquinas de Estado finito onde é apresentado que o mesmo é em média 33% superior aos demais métodos comparados.

Seguindo a mesma abordagem, (HASNAIN, 2021) apresenta uma abordagem de priorização de casos de testes baseada em ontologia para priorizar os casos de testes em programas java, mapeando os atributos e métricas de software de classes java defeituosas. O objetivo é com base nas classes encontrar todos os erros de forma mais rápida sem precisar executar todos os testes. Os resultados mostram que a abordagem proposta apresenta melhores resultados comparado com outras técnicas existentes.

Como forma de selecionar um subconjunto de testes de regressão, (OLSTHOORN; PANICHELLA, 2021) utiliza um Agrupamento Hierárquico Aglomerativo (*Aglomerative Hierarchical Clustering* - AHC) combinado com o *Gene-pool Optimal Mixing Evolutionary Algorithm* (GOMEA). Os resultados obtidos foram melhores do que de outros algoritmos similares comparados.

Utilizando RL em ambiente de Integração Contínua, (BAGHERZADEH, 2021) realiza a comparação entre dez algoritmos de aprendizado de máquina utilizando oito bases públicas, a fim de determinar qual dos algoritmos apresenta melhor desempenho em encontrar falhas em bases distintas. Os resultados demonstraram que o algoritmo ACER utilizando o modelo de classificação *pairwise* obteve a melhor precisão de classificação.

O trabalho de (SHARIF, 2021) propõe o algoritmo nomeado de DeepOrder, que utiliza *Deep Learning* para realizar a priorização de casos de teste baseado em dados históricos de teste, os quais fornecem o tempo de execução e o status de cada teste que foi executado.

Com o objetivo de priorizar testes de forma rápida, (OMRI; SINZ, 2022) em seu trabalho realiza a priorização de casos de testes a partir de aprendizagem por reforço e ranqueamento por aprendizado (*Learning to Rank* - LT). Os testes utilizados na pesquisa são testes automatizados e o critério para priorização foi o seu resultado (passou ou falhou) e tempo de execução.

Por sua vez, (ROZA, 2022) realizou a priorização de casos de testes baseado na ocorrência de falhas em janelas temporais. Ao contrário dos trabalhos anteriores que utilizavam de RL, o autor utilizou da técnica de *Random Forest* (RF) e uma rede de aprendizado profundo de *Long Short Term Memory* (LSTM). Como resultado se concluiu que a RF foi estatisticamente melhor em cerca de 72% dos casos, e a rede LSTM em 55%.

Utilizando uma rede neural profunda (*Deep Neural Network* - DNN), (ABDELKARIM; ELADAWI, 2022) realizou uma priorização de casos de teste a partir das informações do teste e do código-fonte ligado ao mesmo. A ligação entre os testes e o código-fonte foram obtidas devido ao fato de que os testes utilizados são testes a nível de função, e diretamente ligados aos códigos.

Para priorizar casos de testes automatizados, (YARAGHI, 2023) manteve o foco principal de sua pesquisa no trabalho com os dados utilizados, utilizando de dados de *script* de teste, descrição de falhas reportadas, código-fonte do sistema, entre outros. Os testes utilizados na pesquisa são testes automatizados e a nível de unidade (diretamente ligados ao código-fonte). Para coleta de informações textuais foi utilizado o algoritmo BERT, e o objetivo do projeto foi encontrar falhas no sistema. Como forma de avaliação, foi utilizado a métrica de porcentagem média de falhas detectadas com conhecimento de custo (*Cost-cognizant Average Percentage of Faults Detected* - APFDC).

A Tabela 1 apresenta um comparativo entre os trabalhos citados com relação as bases e as técnicas de aprendizagem de máquina utilizadas. Podemos notar que as técnicas mais utilizadas são as de aprendizagem por reforço (RL) e redes neurais (DNN, ANN, CNN). Quanto as bases de pesquisa, as mais utilizadas são de projetos do Github, que por possuírem característica de livre acesso, facilitam a coleta de informações e replicações.

Este projeto de pesquisa ora em apresentação realiza a seleção de casos de teste manuais. Diferente dos trabalhos anteriormente descritos, serão utilizados apenas casos de testes do tipo manual, ou seja, testes que são executados a partir de um passo a passo realizado por um humano, e não por um *script* executado pela máquina. A seleção dos casos de testes será realizada por meio de um sistema de recomendação híbrido, que terá

Trabalho	**Base de dados**	**Técnicas utilizados**
(TONELLA, 2006)	–	CBR e CBR max
(HEMMATI, 2010)	–	baseado em GA
(CHANG, 2010)	Plasma - most MIPS I	SVM
(WANG, 2011)	InTOL	ANN
(CHEN, 2011)	SIR	SSKM
(COTRONEO, 2013)	SIR	NB, BNet, DT, Log
(ABELE; GÖHNER, 2014)	–	GA+Lógica Fuzzy
(LACHMANN, 2016)	–	SVM Rank
(ROSERO, 2017)	–	Matriz de similaridade
(SPIEKER, 2017)	GSDTSR	RL+NN
(WANG, 2018)	Projetos do GitHub	Não utiliza
(AL-SABBAGH, 2019)	–	DT, RF, AdaBoost, MNN e CNN
(JAHAN, 2019)	–	ANN
(LOUSADA; RIBEIRO, 2020)	–	RL+DT
(ROSENBAUER, 2020)	–	RL+XCS-ER
(ROSENBAUER, 2020b)	–	RL+XCS-ER
(ROSENBAUER, 2020a)	–	RL+XCS
(HUANG, 2021)	–	RF
(HASNAIN, 2021)	PROMISE, GitHub, ECLIPSE, Maven	Ontologia
(OLSTHOORN; PANICHELLA, 2021)	–	AHC+GOMEA
(BAGHERZADEH, 2021)	IOFROL, Paint Control, IMAG, IO, COMP, LANG, CODEC e MATH	RL
(SHARIF, 2021)	IOFROL, Paint Control, GSDTSR, Cisco	Deep Learning
(OMRI; SINZ, 2022)	GSDTSR	RL+LR
(ROZA, 2022)	–	RF e LSTM
(ABDELKARIM; ELADAWI, 2022)	Calibre Auto-Waivers Calibre PERC e Calibre nmDRC	DNN
(ABDELKARIM; ELADAWI, 2022)	Projetos do GitHub	DNN
(YARAGHI, 2023)	Projetos do GitHub	RF

Tabela 1 – Comparativo entre trabalhos.

como entrada as descrições textuais de falhas e testes, assim como as interações entre eles, gerando uma lista de testes recomendados a serem executados para uma determinada falha em questão. Abaixo são listados os trabalhos selecionados com o tema de Sistemas de Recomendação, os quais serão base para a criação do modelo proposto.

A recomendação sequencial por reprogramação de *transformer* pré-treinado apresentada por (TANG, 2024), realiza uma reprogramação do *transformers* GPT-2 para recomendação. A partir de uma base temporal de relações entre usuários e itens, realiza a recomendação sequencial de itens para os usuários. Os experimentos realizados apresentaram melhora média de 6,5% no NDCG@5, 6,2% no NDCG@10, 6,1% no Recall@5 e 5,4% no Recall@10 em comparação com as linhas de base.

O trabalho de (SCHWARZER, 2017) apresenta o *Citolytics* - Um Sistema de Recomendação baseado em *links* para a Wikipédia. O *Citolytics* é uma aplicação desenvolvida para *android* que analisa a utilização do usuário e gera uma lista de *links* recomendados, os quais são apresentados na própria aplicação Front-End.

O SAM (*Self-adaptive Attention Module*) proposto por (LIU, 2024) é uma abordagem geral baseada na atenção do módulo SAM para aliviar o viés de seleção derivado da utilização de informações textuais em Sistemas de Recomendação. O módulo proposto pode ser perfeitamente integrado em modelos que contenham componentes de aprendizagem das informações textuais. O modelo aproveita efetivamente a atenção aprendida da representação de informações textuais para aumentar a precisão da previsão alterando a função objetiva de Matriz de Fatoração.

Os sistemas de recomendação desempenham um papel importante para aprimorar a experiência de compra dos usuários, sugerindo tipos semelhantes de produtos com base no histórico de pesquisa anterior do usuário. Em seu trabalho, (DASH, 2023) propõe o uso de esquemas de recomendação de vestuário baseados em semântica de texto para vestuário feminino. Foram adquiridos os dados da *Amazon Product Advertising API* dos quais os títulos dos produtos são processados com os modelos *Average Word-to-Vector* e *Inverse-Document-Frequency Weighted Word-to-Vector* para prosseguir com os cálculos de similaridade. O autor ressalta que deve-se tomar cuidado com a similaridade, caso dois produtos sejam muito similares, eles não deveriam ser recomendados, pois são considerados como cópias.

O trabalho de (SAAKI, 2022) apresenta uma estrutura unificada com reconhecimento de contexto para recomendar os usuários mais adequados para responder às consultas fornecidas durante a restrição de tempo esperada. A abordagem proposta extrai os principais domínios de conhecimento dos conteúdos temporais e contextuais, categorizando os usuários de uma maneira inteligente, com reconhecimento de tempo. Posteriormente, são rastreados os padrões de comportamento do usuário por meio de um procedimento de tempo contínuo infinito. Finalmente, é realizada a classificação dos usuários com base na disponibilidade temporal e na expertise prevalente na consulta de entrada. Recomendando assim os usuários mais qualificados para responder as consultas realizadas. Os resultados experimentais demonstram que a abordagem proposta pode atingir maior eficácia e eficiência em comparação a outros rivais de tendência em sistemas de recomendação de especialistas online.

O trabalho de (ASHWIN, 2006) apresenta um Sistema de Recomendação baseado em mineração de texto para auxiliar na tomada de decisão do cliente na personalização de produtos online. O sistema permite que os clientes descrevam seus interesses em formato textual e assim realiza o processamento textual e geração de recomendações precisas. O sistema emprega técnicas de mineração de texto para aprender as características do

produto e faz a recomendação dos produtos que correspondem as preferências dos clientes. O sistema proposto utiliza técnicas de processamento de linguagem natural como remoção de *stopwords* e *tagging*. Apesar de obter um F1 de 92%, o sistema não utiliza recomendações anteriores como ponto de melhoria, ou seja, não é realizada uma atualização de pesos. Além disso, as técnicas de processamento textual utilizadas não consideram o contexto do texto.

Abordando o contexto incremental, (VINISKI, 2021) em sua pesquisa apresenta um caso de estudos de uma base de dados com característica incremental que apresenta problemas de desvio de conceito e *coldstart*. A base utilizada na pesquisa é uma base de supermercados, onde é realizada uma recomendação por filtragem colaborativa, a qual utiliza a interação entre clientes e itens como entrada do modelo

Desenvolvendo um Sistema de Recomendação que se baseia nas características do perfil do usuário combinadas com o comportamento em tempo real do usuário para recomendar itens em comércio eletrônico, (LUO, 2024) utiliza de processamento textual (TF-IDF, Word2Vec) para extrair informações do perfil do usuário. Para realizar a recomendação é utilizado o textCNN, uma rede neural que utiliza dados textuais como entrada. Como resultados dos experimentos, o autor obteve resultados superiores a 90% por meio da acurácia. Como problemas da pesquisa, o autor menciona a necessidade de tratar o problema de *coldstart*.

Projetando um sistema de recomendação de hotéis de partida a frio, que usa o texto das avaliações como seus dados principais, (LEVI, 2012) trabalha inteiramente no contexto de *ColdStart*. Partindo de um ambiente com poucas informações, o autor utiliza os comentários disponíveis em sites de reserva em hotéis para criar um sistema de recomendação que cria perfis de usuários e os agrupa com base nos comentários realizados. Para contornar o problema de *ColdStart*, é realizado uma verificação de preferência nesses grupos, criando assim um perfil comum do grupo, que pode ser utilizado para realizar recomendações para novos usuários com base nas características fornecidas pelo mesmo.

Para tentar contornar o problema de *Coldstart*, (SHARMA; SHAKYA, 2023) propôs um sistema de recomendação para filmes utilizando filtragem híbrida. Em seu artigo comenta que ainda há muita pesquisa sobre a recomendação de itens de *Coldstart* para sistemas de recomendação porque ainda é uma questão de pesquisa aberta que precisa ser resolvida. Itens de inicialização a frio vêm em dois tipos: um item de inicialização a frio completo (CCS) que não recebe nenhuma classificação, e um item de inicialização a frio incompleto (ICS), que recebe classificações diferentes de zero, mas muito poucas.

Também buscando aliviar o problema de *Coldstart*, (WANG, 2020) desenvolveu uma estrutura que combina um domínio de compras *online* com informações de uma plataforma de anúncios. A estrutura emprega aprendizado profundo para construir um Sistema de Recomendação entre domínios com base em usuários compartilhados nesses dois domínios.

As informações textuais de usuários e itens foram transformadas em vetores latentes pelo Word2Vec. Em seguida, foram aplicados diferentes tipos de modelos de filtragem colaborativa para construir um sistema de recomendação entre domínios. Os resultados experimentais mostram a eficácia de abordagens neurais profundas com conhecimento transferido de outro domínio para o problema de *Coldstart*.

Os trabalhos descritos na literatura tentam selecionar casos de testes principalmente a partir da ocorrência de falhas, o que pode fazer com que testes importantes sejam deixados de lado. Além disso, testes manuais são tratados da mesma forma que testes automatizados, porém são testes totalmente diferentes e seus conteúdos são desprezados. O presente trabalho tem como objetivo realizar a seleção de casos de testes de forma que mesmo que um teste não apresente falhas, ele poderá ser selecionado caso sua relevância seja comprovada. Para isso, será utilizado um sistema de recomendação que, ao contrário dos anteriormente mencionados, utiliza de técnicas de processamento textual mais avançadas (*transformers*) como forma de auxiliar a criação de um sistema híbrido de recomendação, que será capaz de tratar problemas como *Coldstart*.

4 Procedimentos Metodológicos

O Sistema de Recomendação foi idealizado com base na necessidade da empresa parceira de escolher os testes manuais a serem realizados após cada ciclo de desenvolvimento. Atualmente o processo é realizado por um testador, o qual possui experiência e conhecimento sobre os testes disponíveis e alterações realizadas, porém abre brechas para que erros humanos aconteçam. Após a análise dos dados, foi constatada a falta de ligações entre falhas e testes, isso se deve ao fato de que a maioria dos testes nunca apresentaram uma falha, porém sua execução é necessária para garantir que o sistema esteja funcionando corretamente. Algumas falhas também são identificadas por clientes, os quais reportam a falha e ela é registrada no sistema, fazendo com que a ligação entre essa falha e qualquer teste não exista nesse momento. Além disso, ocorre também a falha humana, que no caso deixa de registrar uma determinada falha após um teste apresentar erro em sua execução.

Neste capítulo serão descritos os procedimentos metodológicos utilizados durante a pesquisa experimental. O capítulo é separado em três Seções, as quais vão explicar a origem dos dados utilizados na pesquisa (Seção 4.1), como esses dados foram tratados para que sua utilização na pesquisa seja possível (Seção 4.2), e como esses dados serão utilizados nos experimentos para que o objetivo seja alcançado (Seção 4.3).

4.1 Base de Dados

Os dados utilizados neste projeto são de propriedade da empresa Siemens Brasil Ltda[1] , a qual é parceira do projeto por meio do programa DAI (Doutorado Acadêmico de Inovação). Um dos produtos da empresa parceira é o *Spectrum Power* (SP), o qual foi o foco desta pesquisa. O SP é de forma resumida um sistema de sistemas, onde o foco principal é o gerenciamento e controle de centros operacionais, como por exemplo os centros operacionais que controlam a geração, transmissão e distribuição de energia elétrica, e os centros operacionais que controlam o tratamento, distribuição e armazenamento de água potável. O SP é um software com vários sistemas (chamados de componentes) possuindo diversas funcionalidades e inúmeros colaboradores, várias equipes são responsáveis pela manutenção e continuidade desses sistemas. Durante a pesquisa serão utilizados os dados da versão 7 do SP (SP7).

Um ponto importante a se destacar é que a base de dados utilizada é de um projeto real que segue em desenvolvimento e aprimoramento contínuo, sendo assim, a cada nova coleta de informações, a quantidade de dados será maior do que a da coleta anterior. Os

[1] SIEMENS Brasil. <https://new.siemens.com/br/pt.html>. Acesso: 30-12-2024.

dados compartilhados pela empresa parceira são oriundos do Software Jira[2] que realiza o gerenciamento e controle de execuções de testes e correção de falhas (*Bugs*).

Assumindo um papel próximo a registro de Plano de Teste (descrito na Seção 2.1.5), o Jira é utilizado como documentação *online* de todos os testes e defeitos a nível de interface de usuário (*User Interface - UI*). Os registros no Jira são realizados por meio de *Issues*, as quais podem ser do tipo "*Test Execution*", "*Test*" ou "*Bug*". Dentro das equipes existem mais tipos que são utilizados como forma de organização de dados, porém nesse projeto serão utilizados apenas os três tipos mencionados.

- *Issue Test Execution* - tem como objetivo registrar a execução de um ciclo de Testes, evidenciando os tempos de início e fim, juntamente com o executor e o resultado de cada teste (Passou ou Falhou). Em caso de falhas, é possível referenciar a falha gerada a partir da execução do teste.
- *Issue Test* - descreve e detalha um teste, descrevendo seu propósito, precondições, passos a serem seguidos para execução do mesmo e resultados esperados.
- *Issue Bug* - descreve e notifica uma falha. Contém em sua descrição o detalhamento da falha, como o mesmo ocorreu e quais as implicações dele. Após o fechamento da *issue*, ela apresenta a análise e solução da falha, juntamente com informações de referência ao *branchs* do git para sua correção.

Os ciclos de testes são realizados por cada equipe em um determinado tempo. Os ciclos são chamados internamente pela empresa parceira de Interação e Incremento. A interação é realizada a cada duas semanas, e tem como objetivo avaliar se os componentes alterados ou corrigidos no período do ciclo estão funcionando corretamente. O Incremento é realizado a cada três meses em que são executados todos os testes disponíveis no Jira a fim de garantir que as alterações realizadas naquele período não afetaram outros pontos no sistema. Basicamente os testes realizados no ciclo de interação tem como base apenas os componentes alterados no ciclo de duas semanas, caso um componente não tenha sido alterado nesse período, ele não será testado. Já no incremento, todos os componentes são testados, incluindo aqueles que não sofreram alteração, pois o foco aqui é garantir que tudo está funcionando.

4.2 Preparo da Base

As informações contidas no software Jira tem como objetivo relatar as informações dos testes, sendo essas informações referentes a estado inicial para realização de um teste, o passo a passo de como executar um teste, e os resultados esperados de um teste. O

[2] JIRA. <https://www.atlassian.com/software/jira>. Acesso: 30-12-2024.

software relata também as informações sobre as falhas descobertas a partir dos testes, onde é apresentado um descritivo da falha, e na maioria dos casos o teste que encontrou a falha, como a falha foi resolvida (caso a falha já esteja resolvida). Além das informações anteriormente listadas, o Jira apresenta diversas outras informações para cada tipo de *Issue*, sendo elas separadas em: Teste (*Test*), Execução de Teste (*Test Execution*) e Falha (*Bug*).

4.2.1 Coleta de Dados

A coleta inicial das informações do Jira é realizada a partir da biblioteca Python *Jira*, a qual realiza o acesso as informações da mesma forma que é realizada no site da aplicação. As coletas são realizadas em grupos de 1000 *Issues*, devido ao limite da ferramenta para exportação de informações.

Para que informações relevantes para o projeto não sejam perdidas, a coleta é realizada com todos os atributos disponíveis no Jira, independente se aquele atributo está sendo ou não utilizado por aquele tipo de *Issue*. Após a coleta, é realizada uma filtragem dos atributos coletados, eliminando atributos que não estão sendo utilizados.

Além da coleta das informações das *issues* a partir da biblioteca python, são realizadas coletas via prompt de comando, onde para cada *Issue* com tipo "*Test Execution*", é baixado um arquivo CSV contendo as informações de todos os testes realizados durante a chamada "Execução de Testes", dentre as informações estão: o responsável pelo teste, o horário de início de término do teste, o resultado do teste e em caso de falhas o código de falha gerada por aquele teste. Tais informações não são armazenadas dentro das *issues* "*Test*", pois devido à natureza do software, ele armazena apenas o último status do teste, ou seja, se um mesmo teste é executado em vários "*Test Execution*", o mesmo irá conter apenas a última informação de status.

Após a coleta de informações, os dados serão processados por um transformador a fim de se quantificar os dados textuais. Como os dados do Jira são mais próximos da linguagem natural da língua inglesa, eles serão tratados por meio do algoritmo de SBERT (descrito na Seção 2.2.3). O objetivo aqui é transformar com o auxílio do SBERT, os dados em vetores numéricos, os quais serão utilizados para comparação entre diferentes instâncias da base. Os dados inicialmente coletados serão utilizados para o processo de treinamento, teste e validação do modelo. Durante a utilização do sistema de recomendação, mais informações serão coletadas e o modelo irá se incrementar com base na interação com o usuário.

4.2.2 Geração de arquivos para Experimentos

Para que os experimentos sejam possíveis, os dados coletados devem ser filtrados e preparados para o processo de recomendação. Os dados coletados das *issues* do tipo *Bug* e do tipo *Test* serão transformadas em dois arquivos numéricos, contendo vetores de características das informações relevantes de cada *issue*. Para as *issues* do tipo *Bug*, serão utilizadas as informações de: Identificação (número inteiro que identifica aquela falha); Time (número inteiro que representa o time responsável), Componente (número inteiro que representa o componente ao qual a falha tem relação); e Descrição textual (texto em língua inglesa com termos técnicos que descrevem a falha ocorrida). Para as *issues* do tipo *Bug*, serão utilizadas as informações de: Identificação (número inteiro que identifica aquele Teste); Time (número inteiro que representa o time responsável pela execução daquele teste), Componente (número inteiro que representa o componente ao qual o Teste tem relação); e Passos (ou *steps*, que é um texto em língua inglesa com termos técnicos que descrevem o passo a passo a ser realizado para executar um determinado teste). Os dados textuais dos Passos do Teste e da Descrição da Falha são transformados individualmente por meio do SBERT (Subseção 2.2.3) em vetores com 512 (tamanho máximo de *embedding*) valores em ponto flutuante (*float*). As Figuras 8 e 9 exemplificam os vetores de informações que agora são totalmente numéricos com tamanho padrão de 515 elementos, sendo que cada vetor representa um e apenas um item da base.

```
5692, 10, 18, 0.011587395332753658, - 0.05894535779953003, ..., 0.04975506663322449
```

Figura 8 – Exemplo de Vetor de Teste.

Fonte: Autoria própria

```
25051, 1, 11, -0.02579629421234131, 0.0920877456665039, ..., -0.031126298010349274
```

Figura 9 – Exemplo de Vetor de Falha.

Fonte: Figura do autor

Para os dados coletados das *issues* do tipo *Test Execution*, foram utilizadas apenas as informações referentes a "Testes executados que apresentaram Falhas", ou seja, os Testes que nunca falharam, não são utilizados, assim como as falhas que não tem ligações com os testes, também não são utilizadas. Sobre essas *issues*, são utilizadas as informações de: Identificador do Teste (número inteiro que identifica o teste que foi executado); Identificador da Falha (número inteiro que identifica a Falha que foi encontrada na execução do Teste); *timestamp* (número inteiro que registra o momento em que o teste foi executado). A Figura 10 exemplifica os dados processados para um Teste que apresentou uma falha.

Os vetores gerados são utilizados para gerar três arquivos distintos, os vetores com informações de Falhas, são armazenados no arquivo correspondente as informações de

```
5692, 25051, 11440226
```

Figura 10 – Exemplo de Vetor de Testes que apresentaram Falhas.

Fonte: Figura do autor

falhas. Os vetores com informações de Testes, são armazenados no arquivo correspondente as informações de testes. Já os vetores extraídos das *issue* do tipo *Test Execution*, são armazenados no arquivo de Interações, o qual será utilizado para treinar o modelo. A relação entre falhas e testes tem relação de muitos para muitos, onde uma falha pode ser relatada por vários testes enquanto um teste pode relatar várias falhas.

4.3 Experimentos

Os experimentos são realizados de duas maneiras distintas, a primeira utiliza apenas as informações coletadas, ou seja, o modelo é treinado e testado a partir dos dados coletados (Processamento em Lote). Já a segunda se dá por meio da interação com o usuário final do Sistema de Recomendação, o qual irá validar as recomendações em tempo real e durante o dia a dia na empresa (Processamento Incremental).

4.3.1 Processamento em Lote

Os experimentos com dados coletados são aqueles que utilizam apenas as informações obtidas pela execução dos passos anteriormente descritos. Os dados coletados serão inseridos e utilizados no Sistema de Recomendação seguindo algumas configurações, sendo elas a inicialização dos pesos, separação dos dados de entrada, forma de seleção dos testes e forma de avaliação dos resultados. Os vetores de pesos serão inicializados de três formas, a primeira é de forma aleatória, onde um vetor com 512 posições é gerado para cada teste e cada falha, e carregado com valores aleatórios entre -1 e 1 tais vetores passam a ser os pesos iniciais daqueles testes e falhas. A segunda forma é por meio de dados obtidos do SBERT, o qual fornece um vetor com 512 posições representando semanticamente os textos de cada teste e falha. Já a terceira realiza uma média aritmética entre todos os vetores de um determinado tipo (teste ou falha) e atribui o mesmo valor para todos os itens daquele tipo.

A separação dos dados de entrada se dará a partir de uma validação cruzada de 10 *folds*, onde cada *fold* é testado baseado no treinamento utilizando os demais *folds*. Os dados de cada *fold* são separados de forma sequencial com base no *timestamp*, ou seja, testes que encontraram falhas em datas próximas tendem a ficar no mesmo *fold*. Após o teste com todos os *folds* são obtidos os valores médios e desvio padrão para a base.

A seleção ou recomendação dos testes será realizada de cinco formas, tentando

assim se aproximar do que é realizado na literatura de Sistemas de Recomendação. Na primeira forma todos os testes presentes na base são listados e ordenados por meio da similaridade entre os cossenos com o vetor de pesos da Falha. Na segunda forma todos os testes presentes na base são listados, porém agora são removidos os Testes Positivos (testes que já tiveram interação com aquela falha no treinamento), com o intuito de recomendar apenas testes novos (testes que não tiveram interação durante o treinamento), assim como é geralmente realizado em Sistemas de Recomendações de vendas. Já na terceira forma, todos os testes presentes na base são listados, porém os Testes Positivos são posicionados no início da lista, com o intuito de recomendar testes que aquela falha já teve interação em algum momento, como, por exemplo o teste que relatou a falha. Na quarta forma, é realizada uma ordenação aleatória, onde os testes são dispostos de forma aleatória na lista de recomendação. A quinta forma é a chamada TOPPOP, onde são recomendados os testes que já tiveram interação com alguma falha, de forma que os testes que tiveram mais interações (mais populares), são posicionados no início da lista.

Para avaliar o Sistema de Recomendação, serão utilizadas as métricas Recall@K e NDCG@K, onde K representa os TOP K elementos da lista. O Recall@K descrito na Equação 4.1 irá verificar se o teste rótulo da entrada está presente nos K primeiros elementos da lista de recomendação, caso esteja é considerado como um *Hit*.

$$Recall@K = \frac{1}{|N|} \sum_{t \in N} \begin{cases} 1, & \text{se t está entre os K primeiros} \\ 0, & \text{caso contrário} \end{cases} \tag{4.1}$$

Onde $|N|$ representa a quantidade total interações, t representa o teste rótulo de cada interação (N). Já o NDCG@K (descrito na Equação 4.2) aplica um peso para cada teste que está entre os K primeiros elementos, com base em sua posição na lista (*rank*).

$$NDCG@K = \frac{1}{|N|} \sum_{t \in N} \begin{cases} 0, & \text{se t não está entre os K primeiros} \\ \frac{1}{\log_2(rank+1)}, & \text{caso contrário} \end{cases} \tag{4.2}$$

Onde $|N|$ representa a quantidade total interações, t representa o teste rótulo de cada interação (N) e $rank$ é a posição do emento na lista gerada, onde um $rank$ igual a Zero representa que o teste está na primeira posição da lista. Para cada combinação de inicialização de pesos e forma de seleção, serão realizadas cinco avaliações com as métricas Recall@K e NDCG@K com K variando de 10 até 50.

4.3.2 Processamento Incremental

Os experimentos com interação do usuário são aqueles em que o avaliador do sistema é o próprio usuário final dele, ou seja, as recomendações serão apresentadas para o usuário e ele irá determinar quais recomendações foram úteis. Para que essa recomendação

seja possível, foi necessário o desenvolvimento de uma interface, a qual recomenda testes ao usuário, com base nas informações da falha, podendo essas informações serem novas ou pertencerem a uma falha já registrada.

A interface apresenta ao usuário uma lista de testes, onde o usuário seleciona os Testes Positivos (testes que realmente são boas recomendações), já os demais testes não selecionados serão considerados como Testes Negativos (testes não relevantes para aquela falha). Para a interface, o modelo de recomendação utiliza a combinação de inicialização de pesos e de forma de recomendação que apresentaram os melhores resultados nos experimentos com processamento em lote (Seção 4.3.1).

5 Sistema Híbrido de Recomendação de Testes de Software baseado nas Descrições Textuais de Falhas e Testes

Este capítulo apresenta o método proposto para o Sistema Híbrido de Recomendação de Testes de Software baseado nas descrições textuais de Falhas e Testes. O Sistema de Recomendação é proposto como uma forma de maximizar o processo de testagem manual de softwares, fornecendo ao testador apenas os testes relevantes a serem executados, baseado na descrição de uma falha já reportada ou uma nova falha descrita.

Para que o Sistema de Recomendação possa ser criado, é necessário que minimamente, os seguintes dados estejam disponíveis:

- Documentação das Falhas: Registro histórico das falhas ocorridas no sistema com descrição e data de criação.
- Documentação dos Testes: Passo a passo para execução do teste manual.
- Registro temporal de execução: Uma linha do tempo dos testes executados e das possíveis falhas geradas.
- Informações complementares: Informações sobre grupos de testes, grupos de falhas e descritivos que possam relatar a conexão existente entre testes e falhas.

A construção do Sistema de Recomendação pode ser dividida em quatro etapas, sendo elas: Coleta e Preparo dos Dados; Codificação de Dados; Treinamento do Modelo; e Avaliação e Atualização do Modelo.

A Figura 11 apresenta as principais etapas do método utilizado para criação do Sistema Híbrido de Recomendação de testes manuais baseado nas descrições textuais de Falhas e Testes. A primeira etapa é a coleta e preparo dos dados, que consiste basicamente em coletar informações e identificar ligações entre as informações utilizando de técnicas de processamento textual e manipulação de dados, a primeira etapa é descrita na Seção 5.1. Em seguida é realizada a codificação dos dados, processo esse que transforma os dados textuais em dados numéricos e prepara a base para a etapa de treinamento (Seção 5.2). Já na etapa de treinamento, os dados já codificados são utilizados para gerar o modelo que será a base para o Sistema Híbrido de Recomendação, pois será ele que irá gerar a lista de recomendações de testes para uma determinada falha apresentada na entrada (Seção 5.3). A última etapa chamada de Avaliação e Atualização do Modelo (Seção

5.4) é a etapa em que o Sistema de Recomendação é Testado e Atualizado, fazendo que assim seu aprendizado seja incremental. A última etapa está em um ciclo de execução, ou seja, o processo de atualização e avaliação é constantemente realizado, melhorando a recomendação e gerando novas estatísticas.

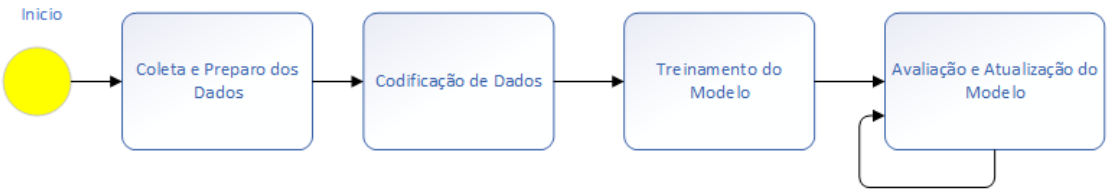


Figura 11 – Sequência de Etapas para a construção do Sistema Híbrido de Recomendação.

Fonte: Figura do autor

5.1 Coleta e preparo dos dados

Nesta primeira etapa ocorrem os procedimentos descritos nas Seções 4.1 e 4.2, onde é realizado o processamento dos dados coletados da documentação.

A Figura 12 apresenta o fluxo de execução de tarefas pertinentes a coleta e preparo dos dados. Em primeiro lugar é realizada a coleta de dados documentais, onde são coletadas as documentações dos Testes e das Falhas ocorridas, juntamente com seus registros temporais, gerando assim três arquivos: Documentação dos Testes, a qual contém em formato textual o passo a passo para execução do teste em linguagem natural técnica, o identificador do teste, time responsável, componente ao qual o teste pertence e demais informações extraídas da documentação; Documentação das Falhas, a qual contém a descrição da Falha em linguagem natural técnica, o identificador da falha, time responsável por tratar a falha, componente a qual a falha tem ligação e demais informações extraídas da documentação; e por último o Registro de Execução de Testes, o qual apresenta em ordem cronológica, os testes executados e as falhas geradas pelos mesmos, além de informações extraídas da Documentação.

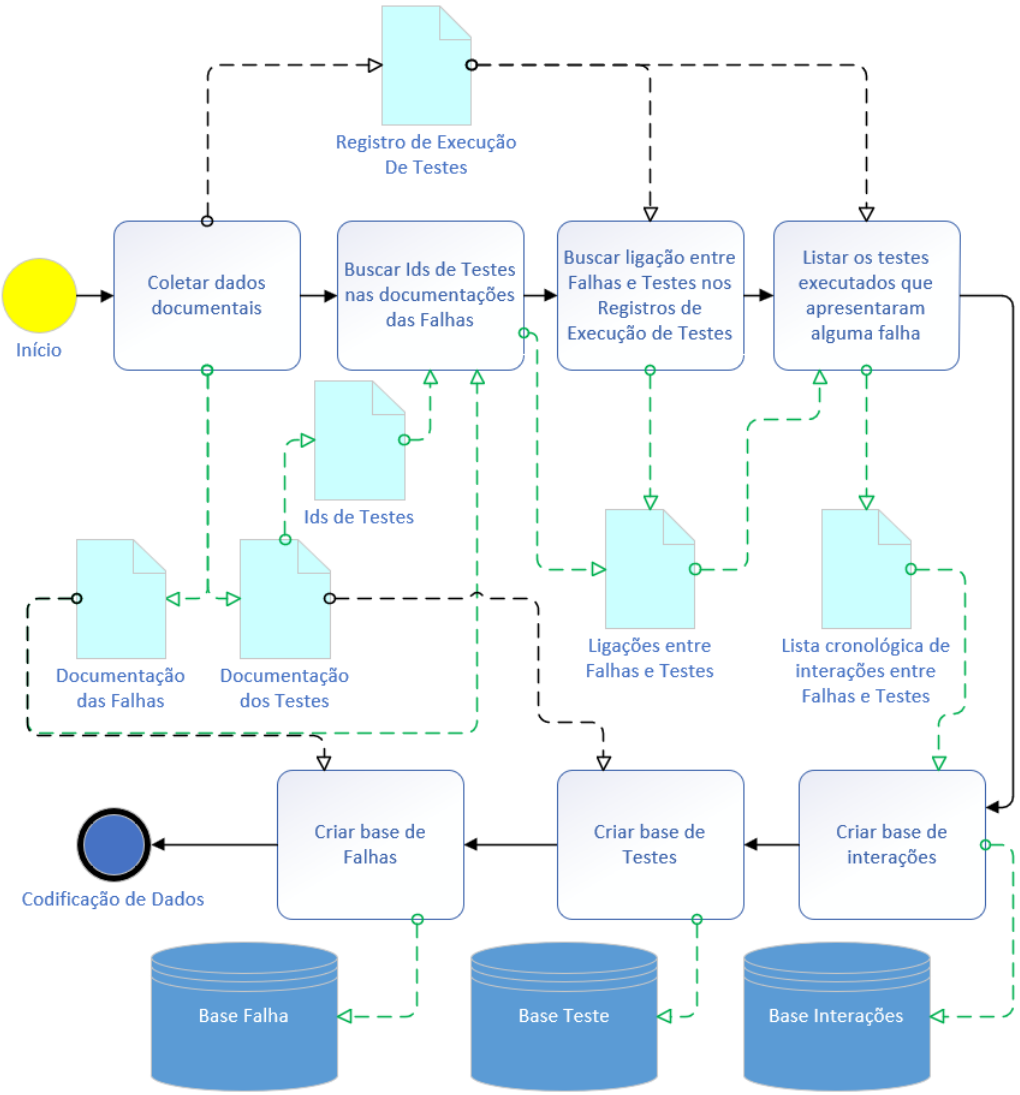


Figura 12 – Diagrama de Coleta e Preparo dos Dados.

Fonte: Figura do autor

Após a coleta de dados documentais, é realizada uma busca textual na Documentação das Falhas, procurando por possíveis identificadores de testes, caso um identificador de teste seja encontrado na Documentação da Falha, um registro de Ligação entre Falha e Teste é gerado. O processo é realizado para cada falha e cada identificador de teste. Após isso, são analisados os Registros de Execução de Testes para identificar possíveis ligações adicionais entre falhas e testes. O resultado desses dois processos, irá gerar uma lista de Ligações entre falhas e testes, o qual é utilizado para filtrar as informações contidas no Registro de Execução de Testes. Como a lista é gerada buscando menções de aos identificadores de falhas e testes, não é necessário uma validação, pois nenhuma suposição é realizada. A filtragem seguinte gera uma lista cronológica de interações entre Falhas e Testes, de forma que todo teste dessa lista tenha interagido com pelo menos uma falha, e cada falha tenha interagido com pelo menos um teste (ligação de muitos para muitos).

Com a listagem de ligações prontas, são realizadas as criações das bases de interação, testes e falhas, as quais serão utilizadas na etapa de Codificação de Dados. A base de interações nesse momento irá armazenar com base na Lista cronológica de interações entre testes e falhas as seguintes informações: Identificador da Falha, identificador do Teste e *timestamp* (marcação de data e hora), e será nomeada nesse momento como Base Interações. A partir da Documentação dos testes coletados no processo de Coleta de dados documentais, é gerada a Base Teste, a qual irá apresentar apenas as informações de: Identificação do Teste, Time responsável pelo Teste, Componente ao qual o teste é ligado e os dados textuais do passo a passo para execução daquele teste. E por fim, utilizando a Documentação das Falhas, é gerada a Base Falha, a qual irá conter as informações de: Identificação da Falha, Time responsável por resolver a Falha, Componente ao qual a falha está presente e Descrição textual da falha. Utilizando essa filtragem de informações, os dados que inicialmente foram coletados da documentação são reduzidos e separados por tipo de informação.

5.2 Codificação de Dados

O processo de codificação de dados consiste em, além de transformar as informações em dados numéricos, organizá-los de forma que as informações de cada ocorrência da base estejam padronizadas com o restante da base. Como os dados utilizados são dados textuais de tamanho variado, é necessário que os dados textuais sejam transformados em vetores de tamanho fixo para que as demais etapas sejam possíveis. Para isso será utilizado o *Sentence-BERT* (descrito na Subseção 2.2.3), o qual irá converter as sentenças de textos de um determinado teste ou falha em um vetor com 512 posições, representando numericamente, a informação semântica dos dados.

A Figura 13 apresenta o passo a passo de como serão tratadas as bases geradas pela etapa anterior (Base Falha, Base Teste e Base Interações). O primeiro passo descrito no diagrama é a Indexação de informações complementares, tal procedimento visa transformar os identificadores, times e componentes presentes na Base Teste e na Base Falha, em informações numéricas. O procedimento realizado é simplesmente inserir um elemento em uma lista e retornar a posição do mesmo na lista, caso o elemento já esteja na lista, é retornada a posição daquele elemento sem realizar novas inserções. As listas geradas nessa etapa são salvas como forma de possível recuperação de informações e para utilização na tratativa da Base Interações.

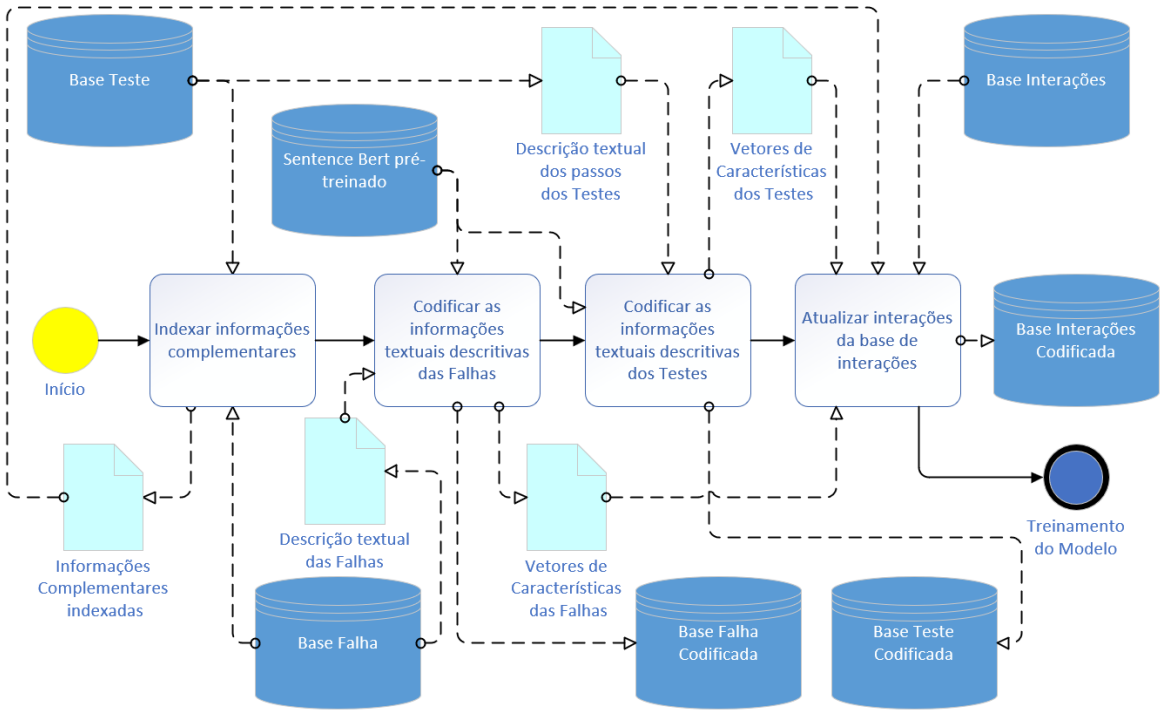


Figura 13 – Diagrama de Codificação de Dados e geração de bases codificadas.

Fonte: Figura do autor

A Etapa seguinte é realizar a codificação das informações textuais descritivas das Falhas, ou seja, os dados textuais da falha são utilizados como entrada para o modelo pré-treinado do *Sentence-BERT* (descrito na Subseção 2.2.3) e o mesmo retorna para cada entrada, um vetor de 512 posições contendo apenas dados numéricos com casas decimais. Para cada Falha na base, é armazenada uma linha na Base Falha Codificada, a qual em cada linha terá 515 números, sendo eles o *index* do identificador, o *index* do time responsável, o *index* do componente, e em seguida 512 números com casas decimais que representam a descrição da falha. As informações inseridas na Base Falha Codificada não são normalizadas.

Em seguida é realizado um processo semelhante ao anterior, porém codificando as informações textuais que descrevem o passo a passo a ser executado durante um teste. O passo a passo de cada teste é utilizado como entrada do modelo pré-treinado *Sentence-BERT*, o qual assim como no passo anterior, irá retornar um vetor numérico com 512 posições, as quais irão ser armazenadas na Base Teste Codificada, onde cada linha terá também 515 números, sendo eles o *index* do identificador, o *index* do time responsável, o *index* do componente, e em seguida 512 números com casas decimais que representam os passos a serem executados para realizar o teste. Assim como na Base Falha Codificada, as informações inseridas na Base Teste Codificada não são normalizadas.

O próximo e último passo da Codificação de Dados é a atualização da Base Interações, substituindo os identificadores de testes e falhas por seus respectivos *index*.

Após todos os passos da etapa de codificação de dados, serão obtidas três novas bases de dados, Base Falha Codificada, Base Teste Codificada e Base Interações Codificada, que serão utilizadas no processo de treinamento do modelo.

5.3 Treinamento do Modelo

Após as bases codificadas serem geradas, é iniciado o processo de treinamento do modelo, o qual é apresentado na Figura 14 e inicia com o processo de inserção de elementos no modelo em forma de objetos. O processo de inserção de elementos no modelo carrega a Base Falha Codificada e a Base Teste Codificada e armazena as informações na memória de forma separada as informações de *index* de identificação, *index* de time, *index* de componentes e vetor textual transformado. O vetor com 512 valores é armazenado como um vetor de *embedding*, e irá ser utilizado para futuros processos.

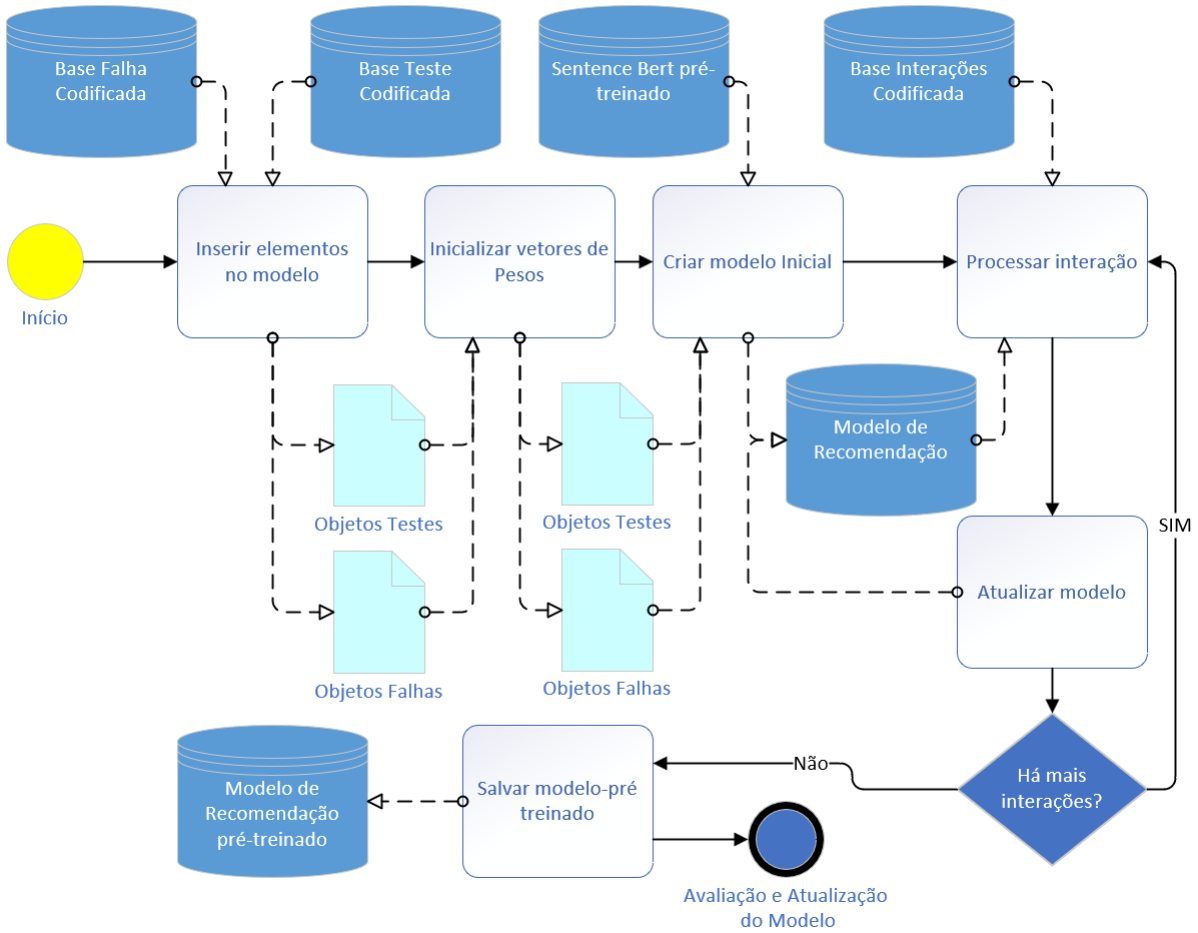


Figura 14 – Diagrama de Treinamento do Modelo e geração do Modelo de Recomendação Pré-treinado.

Fonte: Figura do autor

Com as bases carregadas, é realizado o processo de inicialização dos vetores de pesos, onde conforme descrito na Seção 4.3, onde é realizada a inicialização dos pesos

de três formas, a primeira é gerando um vetor com 512 números com casas decimais aleatórios entre -1 e 1 e atribuindo a cada Teste e cada Falha da base esse vetor de pesos, fazendo com que cada Falha e Teste da base apresente um vetor de pesos único e aleatório, o qual podemos denominar como *pesos_rand*. A segunda inicialização é utilizando o vetor de *embedding*, o qual contém as informações textuais transformadas com auxílio do *Sentence-BERT* e que será clonado para o vetor de pesos que podemos denominar como *pesos_embedding*. A terceira e última forma de inicialização é a média aritmética, onde basicamente é realizada uma média de valores por posição de todos os vetores *embedding* dos Testes, gerando assim um vetor com 512 posições que contém o valor médio de cada posição para os dados de Testes, em seguida cada Teste do modelo receberá um vetor de pesos idêntico que podemos denominar de *pesos_avg*. Os vetores de pesos não são estáticos, e serão atualizados nas etapas seguintes.

O processo seguinte é a criação do modelo chamado de modelo inicial, que é basicamente um modelo sem treinamento, contendo além das informações das Falhas e dos Testes, o modelo pré-treinado *Sentence-BERT*. Isso garante que as futuras transformações textuais em vetores numéricos sejam realizadas com a mesma versão do transformador. O modelo é salvo e denominado de Modelo de Recomendação.

Utilizando o modelo salvo, é possível agora processar cada uma das interações, ou seja, cada interação armazenada na Base Interações Codificadas será utilizada para treinar o modelo. Para o processo de treinamento do modelo, serão necessárias informações de Falha, Teste Positivo e Testes Negativos, os quais serão utilizados nos cálculos da Classificação Bayesiana Personalizada para Matriz de Fatoração - BPR-MF (Subseção 2.2.4.5), com o intuito de aproximar os vetores de pesos da Falha e do Teste Positivo, e distanciar ambos do vetor de pesos do Teste Negativo. O vetor de pesos da Falha e do Teste Positivo são coletados da base de Interações, onde consideramos que qualquer Teste que teve interação com uma falha, é um Teste Positivo para aquela Falha, já os testes Negativos não podemos identificar a partir dos dados, pois um teste não apresentar falhas não significa que ele seja Negativo. Para contornar esse problema, foram utilizados como Testes Negativos, os testes que possuem componentes e times diferentes da falha e do Teste Positivo, fazendo com que esses testes sejam distanciados. Para cada ligação entre Falha e Teste Positivo, são selecionados vinte Testes Negativos.

Após serem selecionados os Testes Negativos, é iniciada o processo de atualização do modelo, o qual para cada Teste Negativo, realiza a atualização dos pesos do Teste Negativo em questão, da falha e do Teste Positivo. O processo de atualização de pesos ocorre para cada tipo de vetor de pesos (Aleatório, *Embedding* e Média) e é realizado por meio do BPR-MF com taxa de aprendizagem igual a 0.001 e taxa de regularização igual a 0.01 seguindo alguns exemplos apresentados em (MNIH; SALAKHUTDINOV, 2007). A função de perda utilizada foi a *Sigmoid* com base no trabalho que apresenta o BPR-MF

(RENDLE, 2012).

O processo de atualização de pesos é repetido para os vinte Testes Negativos, fazendo com que o a Falha e o Teste Positivo se aproximem vinte vezes mais rápido do que se distanciam de cada Teste Negativo. Além disso, o processo é executado para cada tipo de vetor de pesos e para cada item na base de Interações. Após todas as interações da Base Interações Codificada serem processados, o Modelo de Recomendação é salvo e renomeado para Modelo de Recomendação Pré-treinado, o qual será utilizado no processo de avaliação e atualização, e que contém todas as informações necessárias para realizar as recomendações. Vale ressaltar que os vetores de pesos atualizados não tem ligação com os vetores de *embedding* das Falhas e Testes, ou seja, são vetores diferentes que são usados de formas diferentes.

5.4 Avaliação e Atualização do Modelo

A etapa de Avaliação e Atualização do modelo (Figura 15) se inicia com o carregamento do modelo pré-treinado, o qual pode ser armazenado em arquivo na máquina, fazendo com que essa etapa possa ser realizada e repetida em vários momentos. Com o modelo carregado, é realizada a coleta de informações da Falha, podendo ser a descrição textual da falha ou até mesmo o identificador da Falha, os quais são utilizados pelo modelo para determinar se aquela Falha é uma falha já existente no modelo ou é uma falha nova. Caso seja informada apenas a descrição textual da Falha, o Modelo irá transformar essa descrição em um vetor numérico de 512 posições e irá realizar o cálculo de Similaridade entre os cossenos entre o vetor numérico da falha informada e os vetores de *embedding* das Falhas da base.

Caso a Falha não seja uma falha nova, o objeto referente a falha informada é selecionado e a lista de testes recomendados é gerada com base no vetor de pesos. Caso a falha informada seja uma nova falha, ou seja, não é uma falha presente na base, é criado um objeto de Falha a partir das informações fornecidas. Nesse momento é realizada a transformação das descrições textuais em dados numéricos utilizando o *Sentence-BERT* que está salvo no modelo, e em seguida é realizada a inicialização dos pesos. Diferente da inicialização de pesos realizada na criação do modelo de Recomendação, a inicialização de pesos aqui é realizada com base nos dados presentes na base, seguindo algumas premissas: Caso a nova falha tenha similaridade superior a 50% (valor médio entre totalmente idêntico, 100% e totalmente diferente, 0%) com outra falha do modelo, o vetor de pesos da nova falha irá copiar o vetor de pesos da falha com maior similaridade; caso contrário, o vetor de pesos irá ser inicializado com a média dos vetores de pesos das demais falhas com mesmo time e componente. Após a inicialização dos pesos, uma lista de testes recomendados é gerada com base no vetor de pesos, utilizando o método de recomendação que insere no

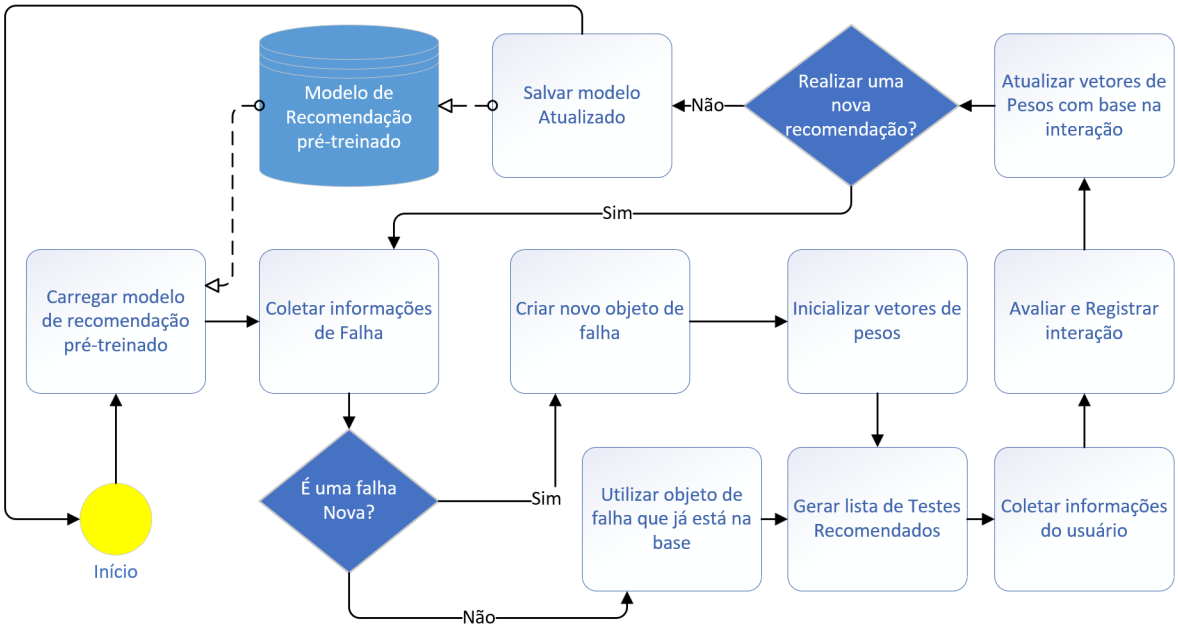


Figura 15 – Diagrama de Avaliação e Atualização do Modelo.

Fonte: Figura do autor

início da fila, os testes que já tiveram interação com aquela falha.

A similaridade entre os cossenos foi utilizada por ser referenciada na documentação do SBERT como técnica *default* para comparação entre vetores de *embedding* e sua fórmula é apresentada na Equação 5.1, onde X e Y são os vetores de entrada da função.

$$SimilaridadeCossenos(X, Y) =< X, Y > /(||X|| * ||Y||)) \tag{5.1}$$

A lista com os testes recomendados é apresentada ao usuário e ele irá selecionar os testes que são relevantes para aquela Falha (Testes Positivos), os demais testes não selecionados serão considerados como irrelevantes (Testes Negativos). Cada Teste Positivo irá gerar uma nova interação na base, e uma atualização de pesos é realizada utilizando o BPR-MF, onde os testes irrelevantes são considerados como Testes Negativos.

Após os vetores de pesos serem atualizados, o modelo é salvo caso o usuário não deseje solicitar mais recomendações e o processo de Avaliação e Atualização do Modelo pode ser inicializado novamente em algum momento futuro. Caso o usuário deseja solicitar mais recomendações, o processo volta para a etapa de coleta de informações da falha, e segue o fluxo de execução aqui descrito. A avaliação do modelo se dá por meio das interações com o usuário por meio das métricas Recall@K e NDCG@K.

6 Resultados

Os resultados da pesquisa visam além de provar as hipóteses descritas no Capítulo 1, apresentar os resultados dos experimentos realizados por meio da combinação de diferentes tipos de inicialização, formas de recomendação e métodos de avaliação. Assim como, mencionado no Capítulo 4, os experimentos foram realizados a partir de duas abordagens, a primeira é a utilização dos dados coletados (Seção 4.3.1) e a segunda é por meio da interação com o usuário (Seção 4.3.2).

6.1 Resultados com os Dados Coletados

Os resultados com dados coletados são obtidos a partir da execução da aplicação das etapas mencionadas na Seção 4.3.1. A coleta de dados do Jira foi realizada e um total de 83.139 *issues* do tipo Falha e 11.352 *issues* do tipo Teste foram coletadas, todas as *issues* são processadas e carregadas no modelo. Para as *issues* do tipo Execução de testes foram coletadas 3.738, as quais em média registram a execução de vinte testes por *issue*, totalizando assim 72.557 registros de execução de testes. Quando um teste é executado e encontra uma falha, é registrada uma interação entre o teste executado e a falha encontrada, além disso, é realizada uma busca textual na documentação das falhas com o intuito de encontrar possíveis identificadores de testes, registrando assim interações adicionais. Como nem todos os testes encontram falhas, para o treinamento do modelo foram utilizadas 3.054 interações, que é a quantidade total de interações registradas no Jira. As interações representam a ligação entre falhas e testes que são registradas no Jira. Baseado na quantidade de Testes e de Falhas em relação a quantidade de interações, podemos dizer que a base utilizada possui uma predominância de interações faltantes, gerando assim um ambiente propício a gerar problemas com *ColdStart*.

Com o objetivo de apresentar uma possível tratativa para o problema de *ColdStart*, foram realizados testes utilizando três tipos de inicialização, sendo elas: a inicialização por *Embedding*, que é a inicialização utilizando o vetor numérico gerado pelo *Sentence-BERT*; a inicialização por Média, que realiza a média aritmética entre todos os vetores de *Embedding*; e a inicialização aleatória, que gera um vetor com dados numéricos aleatórios entre -1 e 1 para cada teste e falha do modelo.

Além dos inicializadores, são apresentados comparativos entre cinco possíveis formas de geração de recomendação, sendo elas: “Base Completa com Positivos”, a qual basicamente ordena o vetor com 11.352 testes por meio da Similaridade entre os cossenos de cada teste e da Falha que está sendo verificada; “Base Completa sem Positivos”, a qual é uma abordagem geralmente utilizada por *e-commerce* ao qual remove da lista de

recomendações aqueles testes aos quais a falha já teve interação, considerando assim com acerto apenas novos testes; "Base Modelo com Positivos", a qual se enquadra melhor as necessidades da aplicação, realizando a recomendação dos Testes Positivos (testes que já tiveram interação com a falha) seguidos de uma lista de testes aos quais a falha nunca teve interação; "Base Aleatório com positivos", a qual gera um vetor de testes com ordenação aleatória contendo todos os 11.352 testes da base; e por último a "Base TOPPOP com Positivos" que gera um vetor ordenado com base na popularidade do teste, ou seja, testes que tem mais ligações com falhas, estarão no início da lista, já testes que não tem ligação com falhas, são colocados no final da lista.

Como métrica de avaliação, foram utilizadas as métricas Recall@K (*Recall@10*, *Recall@20*, *Recall@30*, *Recall@40* e *Recall@50*) e NDCG@K (*NDCG@10*, *NDCG@20*, *NDCG@30*, *NDCG@40* e *NDCG@50*). O Recall@K e o NDCG@K são descritos na Subseção 4.3.1 e suas fórmulas matemáticas são descritas por meio das Equações 4.1 e 4.2 respectivamente. Ambas as métricas consideram os K primeiros testes em um conjunto de 11.352 testes. Vale ressaltar que cada falha possui uma quantidade desconhecida de testes relevantes, podendo essa quantidade variar de acordo com a falha encontrada. Para a realização do treinamento do modelo, foi utilizada a técnica de validação cruzada com dez *folds*, onde cada *fold* é testado com base no treinamento dos demais nove *folds*, e o resultado obtido é a média aritmética entre os dez resultados. Além da média, é apresentado o valor de desvio padrão representado pelo símbolo σ.

As Tabelas 2, 4, 6, 8 e 10 apresentam os resultados dos testes utilizando apenas os dados coletados por meio da métrica Recall@K, para treinamento utilizando 10, 20, 40, 80 e 100 épocas respectivamente. Já as Tabelas 3, 5, 7, 9 e 11 apresentam os resultados dos testes utilizando apenas os dados coletados por meio da métrica NDCG@K, para treinamento utilizando 10, 20, 40, 80 e 100 épocas respectivamente.

Modelos de Recomendação	**Recall@K**	**Inicialização por *Embedding***		**Inicialização por Média**		**Inicialização por Aleatório**	
		média	σ	média	σ	média	σ
Base Completa com positivos	Recall@10	0,366835	0,071097	0,023909	0,014152	0,000656	0,001967
	Recall@20	0,443114	0,076627	0,035372	0,009275	0,003607	0,003423
	Recall@30	0,474884	0,080683	0,047154	0,021111	0,001963	0,002172
	Recall@40	0,499440	0,075390	0,060933	0,016604	0,002951	0,002295
	Recall@50	0,519095	0,073464	0,078310	0,017917	0,007537	0,005499
Base Completa sem positivos	Recall@10	0,176552	0,064894	0,009159	0,004557	0,000328	0,000984
	Recall@20	0,225975	0,074366	0,015384	0,006712	0,001311	0,001606
	Recall@30	0,252175	0,076387	0,016040	0,010091	0,001307	0,001601
	Recall@40	0,271174	0,075382	0,022925	0,008042	0,001311	0,001606
	Recall@50	0,286239	0,075462	0,031774	0,011758	0,003602	0,002296
Base Modelo com positivos	Recall@10	0,554185	0,064786	0,412967	0,097005	0,377625	0,099563
	Recall@20	0,608190	0,061651	0,464017	0,087488	0,400891	0,091239
	Recall@30	0,637349	0,062522	0,475522	0,087493	0,436242	0,087968
	Recall@40	0,657665	0,068727	0,508263	0,088481	0,440840	0,088489
	Recall@50	0,675340	0,070273	0,521386	0,078464	0,453278	0,082865
Base Aleatório com positivos	Recall@10	0,000651	0,001303	0,001635	0,001635	0,000656	0,001311
	Recall@20	0,001639	0,002199	0,002610	0,003184	0,001311	0,001606
	Recall@30	0,003607	0,004002	0,002951	0,001766	0,003274	0,002540
	Recall@40	0,004582	0,003339	0,005889	0,004338	0,002619	0,002857
	Recall@50	0,004254	0,003601	0,005233	0,003326	0,003930	0,003213
Base TOPPOP com positivos	Recall@10	0,054035	0,022216	0,054035	0,022216	0,054035	0,022216
	Recall@20	0,097233	0,029843	0,097233	0,029843	0,097233	0,029843
	Recall@30	0,118838	0,038462	0,118838	0,038462	0,118838	0,038462
	Recall@40	0,136824	0,045769	0,136824	0,045769	0,136824	0,045769
	Recall@50	0,148938	0,044860	0,148938	0,044860	0,148938	0,044860

Tabela 2 – Resultados de Recall@K obtidos a partir do treinamento utilizando 10 épocas.

Modelos de Recomendação	**NDCG@K**	**Inicialização por *Embedding***		**Inicialização por Média**		**Inicialização por Aleatório**	
		média	σ	média	σ	média	σ
Base Completa com positivos	NDCG@10	0,266075	0,055292	0,011516	0,007353	0,000226	0,000678
	NDCG@20	0,285456	0,056636	0,011199	0,003063	0,001061	0,001054
	NDCG@30	0,292013	0,057438	0,015222	0,006000	0,000503	0,000611
	NDCG@40	0,297004	0,056275	0,018835	0,008605	0,001118	0,001261
	NDCG@50	0,300181	0,055653	0,022999	0,007729	0,002270	0,002248
Base Completa sem positivos	NDCG@10	0,123301	0,049557	0,005218	0,003309	0,000109	0,000328
	NDCG@20	0,135872	0,051573	0,004946	0,002612	0,000480	0,000676
	NDCG@30	0,141539	0,052069	0,004897	0,003412	0,000306	0,000378
	NDCG@40	0,145237	0,051697	0,006243	0,002620	0,000470	0,000670
	NDCG@50	0,147871	0,051258	0,009087	0,005291	0,001354	0,001808
Base Modelo com positivos	NDCG@10	0,432689	0,062882	0,328767	0,078566	0,312053	0,082374
	NDCG@20	0,446502	0,061783	0,346092	0,075779	0,316862	0,075917
	NDCG@30	0,453232	0,061789	0,341898	0,075954	0,326834	0,080222
	NDCG@40	0,456890	0,062831	0,348884	0,076945	0,325774	0,079778
	NDCG@50	0,459700	0,062736	0,354561	0,076303	0,328111	0,076899
Base Aleatório com positivos	NDCG@10	0,000201	0,000402	0,000751	0,000792	0,000212	0,000426
	NDCG@20	0,000511	0,000670	0,001074	0,001589	0,000556	0,000749
	NDCG@30	0,000985	0,001094	0,000890	0,000597	0,000805	0,000657
	NDCG@40	0,001271	0,000957	0,001436	0,001091	0,000582	0,000651
	NDCG@50	0,001151	0,001043	0,001380	0,000841	0,000842	0,000649
Base TOPPOP com positivos	NDCG@10	0,026735	0,011304	0,026735	0,011304	0,026735	0,011304
	NDCG@20	0,037461	0,012449	0,037461	0,012449	0,037461	0,012449
	NDCG@30	0,042075	0,013381	0,042075	0,013381	0,042075	0,013381
	NDCG@40	0,045545	0,014473	0,045545	0,014473	0,045545	0,014473
	NDCG@50	0,047741	0,014287	0,047741	0,014287	0,047741	0,014287

Tabela 3 – Resultados de NDCG@K obtidos a partir do treinamento utilizando 10 épocas.

Modelos de Recomendação	**Recall@K**	**Inicialização por *Embedding***		**Inicialização por Média**		**Inicialização por Aleatório**	
		média	σ	média	σ	média	σ
Base Completa com positivos	Recall@10	0,406448	0,067439	0,008512	0,004444	0,000656	0,001311
	Recall@20	0,475850	0,070822	0,024577	0,013068	0,000984	0,001502
	Recall@30	0,506640	0,074634	0,044539	0,008731	0,002295	0,002951
	Recall@40	0,530234	0,069853	0,055035	0,016238	0,002938	0,003398
	Recall@50	0,552180	0,063865	0,059579	0,021687	0,002947	0,002723
Base Completa sem positivos	Recall@10	0,187358	0,068238	0,002942	0,002712	0,000656	0,001311
	Recall@20	0,233499	0,070005	0,012451	0,007309	0,000656	0,001311
	Recall@30	0,258724	0,074409	0,015393	0,005695	0,000984	0,001502
	Recall@40	0,278711	0,075296	0,024237	0,010706	0,001959	0,002608
	Recall@50	0,295087	0,077042	0,021597	0,009476	0,002291	0,002559
Base Modelo com positivos	Recall@10	0,561390	0,063270	0,406107	0,094019	0,378285	0,095355
	Recall@20	0,613108	0,061086	0,458175	0,096798	0,402859	0,098464
	Recall@30	0,642915	0,063649	0,467687	0,103974	0,420223	0,097257
	Recall@40	0,663549	0,068234	0,505976	0,091307	0,432635	0,092565
	Recall@50	0,679261	0,070055	0,517451	0,092557	0,450992	0,086485
Base Aleatório com positivos	Recall@10	0,000000	0,000000	0,000656	0,001311	0,000984	0,001502
	Recall@20	0,001639	0,002199	0,000979	0,001496	0,001967	0,002175
	Recall@30	0,003274	0,002540	0,001639	0,002199	0,002295	0,003893
	Recall@40	0,003270	0,002529	0,004905	0,003341	0,002947	0,002723
	Recall@50	0,003598	0,002284	0,004262	0,003295	0,004254	0,004644
Base TOPPOP com positivos	Recall@10	0,054035	0,022216	0,054035	0,022216	0,054035	0,022216
	Recall@20	0,097233	0,029843	0,097233	0,029843	0,097233	0,029843
	Recall@30	0,118838	0,038462	0,118838	0,038462	0,118838	0,038462
	Recall@40	0,136824	0,045769	0,136824	0,045769	0,136824	0,045769
	Recall@50	0,148938	0,044860	0,148938	0,044860	0,148938	0,044860

Tabela 4 – Resultados de Recall@K obtidos a partir do treinamento utilizando 20 épocas.

Modelos de Recomendação	**NDCG@K**	**Inicialização por *Embedding***		**Inicialização por Média**		**Inicialização por Aleatório**	
		média	σ	média	σ	média	σ
Base Completa com positivos	NDCG@10	0,296512	0,056696	0,004020	0,002858	0,000334	0,000691
	NDCG@20	0,313660	0,057420	0,009996	0,006393	0,000517	0,000834
	NDCG@30	0,320606	0,058501	0,012694	0,003740	0,000516	0,000662
	NDCG@40	0,324488	0,057827	0,014779	0,005478	0,000681	0,000802
	NDCG@50	0,328511	0,056720	0,015361	0,004584	0,000632	0,000580
Base Completa sem positivos	NDCG@10	0,129368	0,050985	0,001096	0,001054	0,000334	0,000691
	NDCG@20	0,140648	0,050541	0,004038	0,002676	0,000310	0,000662
	NDCG@30	0,146248	0,051784	0,004181	0,001586	0,000222	0,000341
	NDCG@40	0,149747	0,051940	0,006436	0,003540	0,000428	0,000584
	NDCG@50	0,152965	0,051987	0,005439	0,002785	0,000451	0,000495
Base Modelo com positivos	NDCG@10	0,437197	0,062277	0,325925	0,076744	0,312772	0,083199
	NDCG@20	0,450180	0,061462	0,338643	0,079071	0,319577	0,082606
	NDCG@30	0,456880	0,061932	0,339015	0,081302	0,322044	0,082610
	NDCG@40	0,460456	0,062672	0,347900	0,079166	0,323656	0,082826
	NDCG@50	0,463493	0,062665	0,347371	0,079051	0,327945	0,080844
Base Aleatório com positivos	NDCG@10	0,000000	0,000000	0,000236	0,000483	0,000447	0,000725
	NDCG@20	0,000808	0,001211	0,000254	0,000390	0,000839	0,001041
	NDCG@30	0,000880	0,000677	0,000502	0,000659	0,000661	0,001298
	NDCG@40	0,000846	0,000764	0,001214	0,000899	0,000794	0,000833
	NDCG@50	0,000957	0,000774	0,000973	0,000770	0,001005	0,001163
Base TOPPOP com positivos	NDCG@10	0,026735	0,011304	0,026735	0,011304	0,026735	0,011304
	NDCG@20	0,037461	0,012449	0,037461	0,012449	0,037461	0,012449
	NDCG@30	0,042075	0,013381	0,042075	0,013381	0,042075	0,013381
	NDCG@40	0,045545	0,014473	0,045545	0,014473	0,045545	0,014473
	NDCG@50	0,047741	0,014287	0,047741	0,014287	0,047741	0,014287

Tabela 5 – Resultados de NDCG@K obtidos a partir do treinamento utilizando 20 épocas.

Modelos de Recomendação	**Recall@K**	**Inicialização por *Embedding***		**Inicialização por Média**		**Inicialização por Aleatório**	
		média	σ	média	σ	média	σ
Base Completa com positivos	Recall@10	0,450340	0,064717	0,006868	0,005358	0,000000	0,000000
	Recall@20	0,516148	0,062227	0,020643	0,008941	0,001631	0,002180
	Recall@30	0,553828	0,062074	0,028461	0,015576	0,001311	0,001606
	Recall@40	0,577737	0,059103	0,038948	0,019597	0,005233	0,003326
	Recall@50	0,601646	0,056706	0,045203	0,019906	0,005553	0,004359
Base Completa sem positivos	Recall@10	0,197829	0,062897	0,004250	0,005076	0,000000	0,000000
	Recall@20	0,250531	0,072356	0,009168	0,005640	0,001307	0,001601
	Recall@30	0,274444	0,077538	0,016036	0,012805	0,000656	0,001311
	Recall@40	0,293439	0,076477	0,013762	0,011153	0,003926	0,003525
	Recall@50	0,312107	0,081054	0,019310	0,009529	0,003594	0,002694
Base Modelo com positivos	Recall@10	0,567943	0,064635	0,411366	0,092957	0,377314	0,097307
	Recall@20	0,618669	0,063235	0,438545	0,101716	0,398928	0,092871
	Recall@30	0,649792	0,066171	0,479150	0,104015	0,424119	0,096671
	Recall@40	0,668778	0,067762	0,499776	0,081606	0,437553	0,092169
	Recall@50	0,686462	0,068966	0,513521	0,087016	0,454922	0,082919
Base Aleatório com positivos	Recall@10	0,000656	0,001967	0,000324	0,000971	0,001639	0,002199
	Recall@20	0,001307	0,001601	0,002295	0,002951	0,001631	0,002180
	Recall@30	0,002619	0,002857	0,002619	0,002452	0,002619	0,002857
	Recall@40	0,003279	0,002933	0,004918	0,004212	0,003270	0,002923
	Recall@50	0,003607	0,003423	0,003926	0,003818	0,004245	0,003855
Base TOPPOP com positivos	Recall@10	0,054035	0,022216	0,054035	0,022216	0,054035	0,022216
	Recall@20	0,097233	0,029843	0,097233	0,029843	0,097233	0,029843
	Recall@30	0,118838	0,038462	0,118838	0,038462	0,118838	0,038462
	Recall@40	0,136824	0,045769	0,136824	0,045769	0,136824	0,045769
	Recall@50	0,148938	0,044860	0,148938	0,044860	0,148938	0,044860

Tabela 6 – Resultados de Recall@K obtidos a partir do treinamento utilizando 40 épocas.

Modelos de Recomendação	**NDCG@K**	**Inicialização por *Embedding***		**Inicialização por Média**		**Inicialização por Aleatório**	
		média	σ	média	σ	média	σ
Base Completa com positivos	NDCG@10	0,340864	0,058447	0,003774	0,003485	0,000000	0,000000
	NDCG@20	0,358405	0,057076	0,006641	0,003112	0,000397	0,000535
	NDCG@30	0,365110	0,056900	0,009113	0,005227	0,000501	0,000695
	NDCG@40	0,370385	0,056497	0,009703	0,004945	0,002159	0,002027
	NDCG@50	0,374860	0,055596	0,012793	0,006212	0,001626	0,001279
Base Completa sem positivos	NDCG@10	0,137557	0,049284	0,002624	0,003448	0,000000	0,000000
	NDCG@20	0,151417	0,051979	0,002967	0,001876	0,000318	0,000390
	NDCG@30	0,155674	0,052556	0,005164	0,004097	0,000276	0,000632
	NDCG@40	0,159890	0,052426	0,003553	0,002850	0,001359	0,001488
	NDCG@50	0,162925	0,053144	0,005345	0,003297	0,001213	0,001051
Base Modelo com positivos	NDCG@10	0,443878	0,061249	0,324938	0,076412	0,313946	0,083029
	NDCG@20	0,457370	0,061153	0,333197	0,081030	0,319681	0,082547
	NDCG@30	0,463155	0,061475	0,338625	0,082147	0,321886	0,082937
	NDCG@40	0,467390	0,061367	0,345618	0,072386	0,326455	0,082918
	NDCG@50	0,470200	0,061625	0,345691	0,080011	0,329249	0,079198
Base Aleatório com positivos	NDCG@10	0,000305	0,000915	0,000125	0,000376	0,000902	0,001264
	NDCG@20	0,000396	0,000507	0,000952	0,001328	0,000551	0,000750
	NDCG@30	0,000940	0,001258	0,000690	0,000767	0,000695	0,000771
	NDCG@40	0,000979	0,000857	0,001382	0,001289	0,000969	0,001134
	NDCG@50	0,001115	0,001258	0,001572	0,001547	0,001150	0,001009
Base TOPPOP com positivos	NDCG@10	0,026735	0,011304	0,026735	0,011304	0,026735	0,011304
	NDCG@20	0,037461	0,012449	0,037461	0,012449	0,037461	0,012449
	NDCG@30	0,042075	0,013381	0,042075	0,013381	0,042075	0,013381
	NDCG@40	0,045545	0,014473	0,045545	0,014473	0,045545	0,014473
	NDCG@50	0,047741	0,014287	0,047741	0,014287	0,047741	0,014287

Tabela 7 – Resultados de NDCG@K obtidos a partir do treinamento utilizando 40 épocas.

Modelos de Recomendação	**Recall@K**	**Inicialização por *Embedding***		**Inicialização por Média**		**Inicialização por Aleatório**	
		média	σ	média	σ	média	σ
Base Completa com positivos	Recall@10	0,513900	0,058420	0,003926	0,003525	0,001311	0,001606
	Recall@20	0,575778	0,055023	0,012097	0,007125	0,002282	0,003869
	Recall@30	0,607211	0,058907	0,015397	0,007050	0,002951	0,005763
	Recall@40	0,632751	0,054991	0,027503	0,010278	0,007209	0,005644
	Recall@50	0,647816	0,052897	0,025186	0,011063	0,007869	0,006425
Base Completa sem positivos	Recall@10	0,222066	0,064975	0,002942	0,003414	0,001311	0,001606
	Recall@20	0,267887	0,075089	0,006864	0,005729	0,001954	0,003909
	Recall@30	0,295394	0,077240	0,007865	0,007363	0,002951	0,005763
	Recall@40	0,317004	0,074122	0,014724	0,006396	0,001963	0,002172
	Recall@50	0,329118	0,073650	0,017339	0,008502	0,000656	0,001311
Base Modelo com positivos	Recall@10	0,573513	0,068196	0,398579	0,096638	0,379260	0,096486
	Recall@20	0,629792	0,062835	0,433270	0,093669	0,403834	0,102503
	Recall@30	0,658623	0,067910	0,464375	0,100050	0,430008	0,088713
	Recall@40	0,678934	0,069804	0,478490	0,085474	0,441828	0,095125
	Recall@50	0,695966	0,073681	0,510890	0,089076	0,450681	0,087945
Base Aleatório com positivos	Recall@10	0,000984	0,001502	0,000984	0,002099	0,000984	0,002099
	Recall@20	0,001967	0,003005	0,001311	0,001606	0,002291	0,002559
	Recall@30	0,003274	0,002933	0,001967	0,001606	0,002623	0,001967
	Recall@40	0,003590	0,004472	0,002623	0,004095	0,002619	0,003212
	Recall@50	0,003930	0,003213	0,005246	0,003651	0,002623	0,003212
Base TOPPOP com positivos	Recall@10	0,054035	0,022216	0,054035	0,022216	0,054035	0,022216
	Recall@20	0,097233	0,029843	0,097233	0,029843	0,097233	0,029843
	Recall@30	0,118838	0,038462	0,118838	0,038462	0,118838	0,038462
	Recall@40	0,136824	0,045769	0,136824	0,045769	0,136824	0,045769
	Recall@50	0,148938	0,044860	0,148938	0,044860	0,148938	0,044860

Tabela 8 – Resultados de Recall@K obtidos a partir do treinamento utilizando 80 épocas.

Modelos de Recomendação	**NDCG@K**	**Inicialização por *Embedding***		**Inicialização por Média**		**Inicialização por Aleatório**	
		média	σ	média	σ	média	σ
Base Completa com positivos	NDCG@10	0,392683	0,059253	0,002244	0,002484	0,000586	0,000728
	NDCG@20	0,408268	0,057104	0,004698	0,003468	0,001419	0,002999
	NDCG@30	0,414911	0,058146	0,005242	0,003265	0,000707	0,001427
	NDCG@40	0,420730	0,056619	0,007286	0,003055	0,001548	0,001165
	NDCG@50	0,423030	0,056902	0,006884	0,004816	0,002374	0,002464
Base Completa sem positivos	NDCG@10	0,153498	0,049484	0,001884	0,002474	0,000586	0,000728
	NDCG@20	0,165517	0,051993	0,002296	0,001883	0,001340	0,003025
	NDCG@30	0,171237	0,051847	0,002342	0,002520	0,000707	0,001427
	NDCG@40	0,176239	0,052090	0,004039	0,001964	0,000516	0,000594
	NDCG@50	0,177907	0,052174	0,005165	0,004822	0,000148	0,000297
Base Modelo com positivos	NDCG@10	0,450277	0,064700	0,315757	0,079658	0,313623	0,084174
	NDCG@20	0,464434	0,062782	0,327639	0,078000	0,322424	0,083717
	NDCG@30	0,470561	0,063702	0,335274	0,077210	0,323979	0,080254
	NDCG@40	0,475292	0,063598	0,339360	0,076611	0,327934	0,083633
	NDCG@50	0,478383	0,063974	0,343505	0,074919	0,328958	0,082993
Base Aleatório com positivos	NDCG@10	0,000316	0,000483	0,000325	0,000702	0,000457	0,000995
	NDCG@20	0,000583	0,000955	0,000420	0,000560	0,000775	0,000903
	NDCG@30	0,001300	0,001189	0,000574	0,000510	0,000672	0,000557
	NDCG@40	0,001034	0,001378	0,000662	0,000991	0,000560	0,000698
	NDCG@50	0,000944	0,000753	0,001365	0,001068	0,000577	0,000717
Base TOPPOP com positivos	NDCG@10	0,026735	0,011304	0,026735	0,011304	0,026735	0,011304
	NDCG@20	0,037461	0,012449	0,037461	0,012449	0,037461	0,012449
	NDCG@30	0,042075	0,013381	0,042075	0,013381	0,042075	0,013381
	NDCG@40	0,045545	0,014473	0,045545	0,014473	0,045545	0,014473
	NDCG@50	0,047741	0,014287	0,047741	0,014287	0,047741	0,014287

Tabela 9 – Resultados de NDCG@K obtidos a partir do treinamento utilizando 80 épocas.

Modelos de Recomendação	**Recall@K**	**Inicialização por *Embedding***		**Inicialização por Média**		**Inicialização por Aleatório**	
		média	σ	média	σ	média	σ
Base Completa com positivos	Recall@10	0,513900	0,058420	0,003926	0,003525	0,001311	0,001606
	Recall@20	0,575778	0,055023	0,012097	0,007125	0,002282	0,003869
	Recall@30	0,607211	0,058907	0,015397	0,007050	0,002951	0,005763
	Recall@40	0,632751	0,054991	0,027503	0,010278	0,007209	0,005644
	Recall@50	0,647816	0,052897	0,025186	0,011063	0,007869	0,006425
Base Completa sem positivos	Recall@10	0,222066	0,064975	0,002942	0,003414	0,001311	0,001606
	Recall@20	0,267887	0,075089	0,006864	0,005729	0,001954	0,003909
	Recall@30	0,295394	0,077240	0,007865	0,007363	0,002951	0,005763
	Recall@40	0,317004	0,074122	0,014724	0,006396	0,001963	0,002172
	Recall@50	0,329118	0,073650	0,017339	0,008502	0,000656	0,001311
Base Modelo com positivos	Recall@10	0,573513	0,068196	0,398579	0,096638	0,379260	0,096486
	Recall@20	0,629792	0,062835	0,433270	0,093669	0,403834	0,102503
	Recall@30	0,658623	0,067910	0,464375	0,100050	0,430008	0,088713
	Recall@40	0,678934	0,069804	0,478490	0,085474	0,441828	0,095125
	Recall@50	0,695966	0,073681	0,510890	0,089076	0,450681	0,087945
Base Aleatório com positivos	Recall@10	0,000984	0,001502	0,000984	0,002099	0,000984	0,002099
	Recall@20	0,001967	0,003005	0,001311	0,001606	0,002291	0,002559
	Recall@30	0,003274	0,002933	0,001967	0,001606	0,002623	0,001967
	Recall@40	0,003590	0,004472	0,002623	0,004095	0,002619	0,003212
	Recall@50	0,003930	0,003213	0,005246	0,003651	0,002623	0,003212
Base TOPPOP com positivos	Recall@10	0,054035	0,022216	0,054035	0,022216	0,054035	0,022216
	Recall@20	0,097233	0,029843	0,097233	0,029843	0,097233	0,029843
	Recall@30	0,118838	0,038462	0,118838	0,038462	0,118838	0,038462
	Recall@40	0,136824	0,045769	0,136824	0,045769	0,136824	0,045769
	Recall@50	0,148938	0,044860	0,148938	0,044860	0,148938	0,044860

Tabela 10 – Resultados de Recall@K obtidos a partir do treinamento utilizando 100 épocas.

Modelos de Recomendação	**NDCG@K**	**Inicialização por *Embedding***		**Inicialização por Média**		**Inicialização por Aleatório**	
		média	σ	média	σ	média	σ
Base Completa com positivos	NDCG@10	0,407082	0,056772	0,004936	0,003179	0,000207	0,000621
	NDCG@20	0,421619	0,056762	0,002849	0,001793	0,001078	0,001651
	NDCG@30	0,427576	0,056704	0,005182	0,004295	0,001211	0,001189
	NDCG@40	0,433039	0,055584	0,008142	0,006116	0,001449	0,001987
	NDCG@50	0,435475	0,054788	0,006551	0,002555	0,004235	0,002563
Base Completa sem positivos	NDCG@10	0,157800	0,050703	0,003092	0,002318	0,000207	0,000621
	NDCG@20	0,169975	0,052564	0,001875	0,001639	0,001004	0,001681
	NDCG@30	0,175317	0,051909	0,002881	0,003675	0,000704	0,001050
	NDCG@40	0,180215	0,052543	0,003449	0,002190	0,000390	0,000506
	NDCG@50	0,181846	0,051544	0,003211	0,002498	0,000934	0,001153
Base Modelo com positivos	NDCG@10	0,453585	0,064779	0,324283	0,077110	0,317678	0,079804
	NDCG@20	0,467026	0,063377	0,327248	0,080567	0,322149	0,082520
	NDCG@30	0,472263	0,064387	0,334775	0,077125	0,325120	0,082975
	NDCG@40	0,476695	0,063859	0,338990	0,083512	0,328856	0,082343
	NDCG@50	0,479325	0,063633	0,338829	0,083849	0,331608	0,079970
Base Aleatório com positivos	NDCG@10	0,000117	0,000350	0,000408	0,000681	0,000339	0,000530
	NDCG@20	0,000451	0,000651	0,000274	0,000421	0,000813	0,002189
	NDCG@30	0,000909	0,000566	0,002024	0,001777	0,001281	0,002006
	NDCG@40	0,000698	0,000850	0,000891	0,000732	0,000287	0,000358
	NDCG@50	0,000720	0,000685	0,000674	0,000669	0,001084	0,001033
Base TOPPOP com positivos	NDCG@10	0,026735	0,011304	0,026735	0,011304	0,026735	0,011304
	NDCG@20	0,037461	0,012449	0,037461	0,012449	0,037461	0,012449
	NDCG@30	0,042075	0,013381	0,042075	0,013381	0,042075	0,013381
	NDCG@40	0,045545	0,014473	0,045545	0,014473	0,045545	0,014473
	NDCG@50	0,047741	0,014287	0,047741	0,014287	0,047741	0,014287

Tabela 11 – Resultados de NDCG@K obtidos a partir do treinamento utilizando 100 épocas.

Podemos notar que os melhores resultados das dez tabelas, se encontram na inicialização utilizando *Embedding*, superando as demais abordagem de inicialização de pesos utilizadas. Isso se deve ao fato de que a inicialização utilizando os dados textuais faz com que falhas e testes que abordam os mesmos assuntos em suas descrições e passo a passo, irão apresentar vetores numéricos mais próximos devido a análise de contexto gerada pelo *Sentence-BERT*. Analisando esses resultados podemos responder a questão de pesquisa 1, elencando a inicialização de pesos com os *embeddings* como o método de inicialização que apresenta os melhores resultados.

Quando analisamos as formas de recomendação, podemos ver um distanciamento maior da "Base Completa sem Positivos", isso se deve ao fato de que para o conjunto de dados utilizados nessa pesquisa, a remoção dos testes positivos não é interessante, pois uma determinada falha tende a ser relacionada a um mesmo teste até que ela seja solucionada, já as formas de recomendação que mantém os Testes Positivos são mais adequadas para a natureza da base. Entre as formas que utilizam os Testes Positivos na recomendação, podemos notar que a diferença entre a abordagem que utiliza "Base Completa com Positivos" e a abordagem que utiliza "Base Modelo com Positivos" é maior quando utilizado o treinamento com 10 épocas (Tabelas 2 e 3), porém ele se aproxima quando a quantidade de épocas é 20 ou mais. Com essa informação podemos responder a questão de pesquisa 2, elencando o método "Base Modelo com Positivos" como a melhor forma de recomendação a ser utilizada quando a base não possui treinamento, e os métodos que utilizam Testes Positivos como melhores formas de recomendação para seleção de testes.

Analisando o quadro de inicialização por *Embedding* "Base Modelo com Positivos", podemos notar que os resultados obtidos por meio métrica Recall@10 é próximo aos resultados obtidos por meio da métrica Recall@50. Isso se repete quando analisamos os resultados obtidos por meio da métrica NDCG@10 e NDCG@50, mostrando que grande parte dos acertos ocorre dentro dos 10 primeiros elementos.

Podemos notar também que não há grande diferença entre o uso de 10 ou 100 épocas no treinamento, possivelmente pelo fato de que há poucos dados de entrada para o treinamento (interações), fazendo com que grande parte da base não possua qualquer treinamento independente da quantidade de épocas. Com isso, podemos concluir que a quantidade de épocas de treinamento não está relacionada a melhora de resultados, principalmente em bases em que a quantidade de interações é proporcionalmente pequena em relação a quantidade de dados.

Para confirmar as afirmações sobre os resultados, foram realizados testes estatísticos nos resultados, os quais são apresentados na Subseção 6.1.1.

6.1.1 Testes Estatísticos

Os testes estatísticos foram realizados utilizando os resultados obtidos em cada validação realizada na validação cruzada, ou seja, cada combinação de de média e desvio padrão anteriormente apresentados, remete a um vetor com dez resultados oriundos da execução de uma validação cruzada de 10 *folds*. Os testes estatísticos realizados são o Teste de *Friedman* e o Teste de *Nemenyi*. O teste de *Friedman* foi executado através da biblioteca *Python scipy*[1], já a o Teste de *Nemenyi* foi realizado através da biblioteca *Python Orange*[2]. Os testes estatísticos foram realizados para identificar possíveis diferenças entre os métodos de inicialização (Tabela 12), diferenças entre os modelos de recomendação (Tabela 13), e diferenças entre as épocas de treinamento (Tabela 14).

A Tabela 12 apresenta os *pvalue* resultantes do Teste de *Friendman*, onde um *pvalue* menor que 0,05 (*pvalue*<0,05) indica que há diferença significativa entre os valores testados. Cada valor na tabela foi calculado através dos vetores de resultados dos testes utilizando os métodos de inicialização *Embedding*, Média e Aleatório. Podemos verificar que para os modelos de recomendação "Base Completa com positivos", "Base Completa sem positivos" e "Base Modelo com positivos", independente da quantidade de épocas ou da métrica utilizada, os resultados foram inferiores a 0,05, o que representa uma diferença significativa entre os resultados. Os resultados para o modelo de recomendação "Base Aleatório com positivos" foram superiores a 0,05, o que significa que não há diferença entre os resultados, isso se deve ao fato de que esse modelo não se baseia na inicialização dos pesos, ele apenas escolhe um teste aleatório da base. Os resultados para o modelo de recomendação "Base TOPPOP com positivos" apresentaram resultados NaN (*Not a Number*) devido ao fato de que todos os resultados listados foram idênticos, pois o modelo sempre irá recomendar os testes mais populares, com isso podemos concluir que não há diferença alguma entre os resultados para esse modelo.

A Figura 16 apresenta o complemento da Tabela 12, onde podemos notar que para os modelos que apresentaram diferença estatística no teste de *Friedman*, os melhores resultados estão sempre apresentados na inicialização por *Embedding* (apresentados na Figura nos itens "a", "b", "c", "d", "e" e "f"). Para os demais modelos, não há diferença estatística para os métodos de inicialização, pois não se baseiam nos vetores de pesos para gerar as recomendações.

[1] https://scipy.org/
[2] https://orangedatamining.com/

Modelos de Recomendação	Métrica	10 épocas	20 épocas	40 épocas	80 épocas	100 épocas
Base Completa com positivos	Recall@10	4,540E-05	6,917E-05	6,917E-05	3,623E-04	5,718E-05
	Recall@20	4,540E-05	4,540E-05	4,540E-05	2,249E-04	2,249E-04
	Recall@30	4,540E-05	4,540E-05	1,117E-04	1,117E-04	6,917E-05
	Recall@40	4,540E-05	4,540E-05	5,718E-05	1,117E-04	2,249E-04
	Recall@50	4,540E-05	4,540E-05	4,540E-05	1,299E-04	4,448E-04
	NDCG@10	4,540E-05	1,299E-04	6,917E-05	3,623E-04	5,718E-05
	NDCG@20	4,540E-05	4,540E-05	4,540E-05	2,249E-04	2,249E-04
	NDCG@30	4,540E-05	4,540E-05	1,117E-04	4,540E-05	4,540E-05
	NDCG@40	4,540E-05	4,540E-05	1,117E-04	4,540E-05	2,249E-04
	NDCG@50	4,540E-05	4,540E-05	4,540E-05	2,249E-04	3,707E-04
Base Completa sem positivos	Recall@10	4,540E-05	2,446E-04	8,843E-05	3,355E-04	5,718E-05
	Recall@20	4,540E-05	5,718E-05	1,117E-04	1,532E-04	2,361E-04
	Recall@30	5,718E-05	4,540E-05	1,299E-04	1,299E-04	1,445E-04
	Recall@40	4,540E-05	4,540E-05	1,445E-04	4,540E-05	4,540E-05
	Recall@50	4,540E-05	4,540E-05	1,117E-04	4,540E-05	1,299E-04
	NDCG@10	4,540E-05	2,446E-04	8,843E-05	3,355E-04	5,718E-05
	NDCG@20	4,540E-05	1,117E-04	1,117E-04	1,532E-04	3,355E-04
	NDCG@30	5,718E-05	4,540E-05	1,117E-04	1,299E-04	2,249E-04
	NDCG@40	4,540E-05	4,540E-05	3,707E-04	4,540E-05	4,540E-05
	NDCG@50	4,540E-05	4,540E-05	1,117E-04	4,540E-05	1,299E-04
Base Modelo com positivos	Recall@10	5,718E-05	5,718E-05	1,117E-04	1,117E-04	2,249E-04
	Recall@20	4,540E-05	4,540E-05	4,540E-05	4,540E-05	4,540E-05
	Recall@30	4,540E-05	4,540E-05	4,540E-05	1,117E-04	4,540E-05
	Recall@40	4,540E-05	4,540E-05	4,540E-05	1,117E-04	1,117E-04
	Recall@50	4,540E-05	4,540E-05	4,540E-05	4,540E-05	2,249E-04
	NDCG@10	4,540E-05	1,117E-04	2,249E-04	3,707E-04	3,707E-04
	NDCG@20	1,117E-04	1,117E-04	1,117E-04	3,707E-04	2,249E-04
	NDCG@30	1,117E-04	1,117E-04	4,540E-05	1,117E-04	2,249E-04
	NDCG@40	4,540E-05	4,540E-05	1,117E-04	1,117E-04	1,117E-04
	NDCG@50	4,540E-05	4,540E-05	4,540E-05	1,117E-04	3,707E-04
Base Aleatório com positivos	Recall@10	2,231E-01	1,738E-01	2,231E-01	9,487E-01	5,134E-01
	Recall@20	6,625E-01	5,945E-01	8,338E-01	7,979E-01	8,338E-01
	Recall@30	9,692E-01	3,406E-01	8,521E-01	6,286E-01	9,048E-01
	Recall@40	3,434E-01	3,406E-01	5,623E-01	9,636E-01	2,058E-01
	Recall@50	3,558E-01	8,910E-01	9,017E-01	3,259E-01	8,035E-01
	NDCG@10	9,072E-02	1,462E-01	2,231E-01	9,487E-01	6,918E-01
	NDCG@20	8,869E-01	5,073E-01	7,716E-01	8,089E-01	8,338E-01
	NDCG@30	8,975E-01	3,147E-01	9,078E-01	2,008E-01	4,018E-01
	NDCG@40	4,389E-01	8,948E-01	9,001E-01	9,649E-01	3,281E-01
	NDCG@50	1,465E-01	7,733E-01	4,692E-01	3,679E-01	7,037E-01
Base TOPPOP com positivos	Recall@10	NaN	NaN	NaN	NaN	NaN
	Recall@20	NaN	NaN	NaN	NaN	NaN
	Recall@30	NaN	NaN	NaN	NaN	NaN
	Recall@40	NaN	NaN	NaN	NaN	NaN
	Recall@50	NaN	NaN	NaN	NaN	NaN
	NDCG@10	NaN	NaN	NaN	NaN	NaN
	NDCG@20	NaN	NaN	NaN	NaN	NaN
	NDCG@30	NaN	NaN	NaN	NaN	NaN
	NDCG@40	NaN	NaN	NaN	NaN	NaN
	NDCG@50	NaN	NaN	NaN	NaN	NaN

Tabela 12 – Valores de *pvalue* para o teste de *Friedman* comparando os métodos de inicialização. *pvalue* = NaN representa que os vetores testados são idênticos.

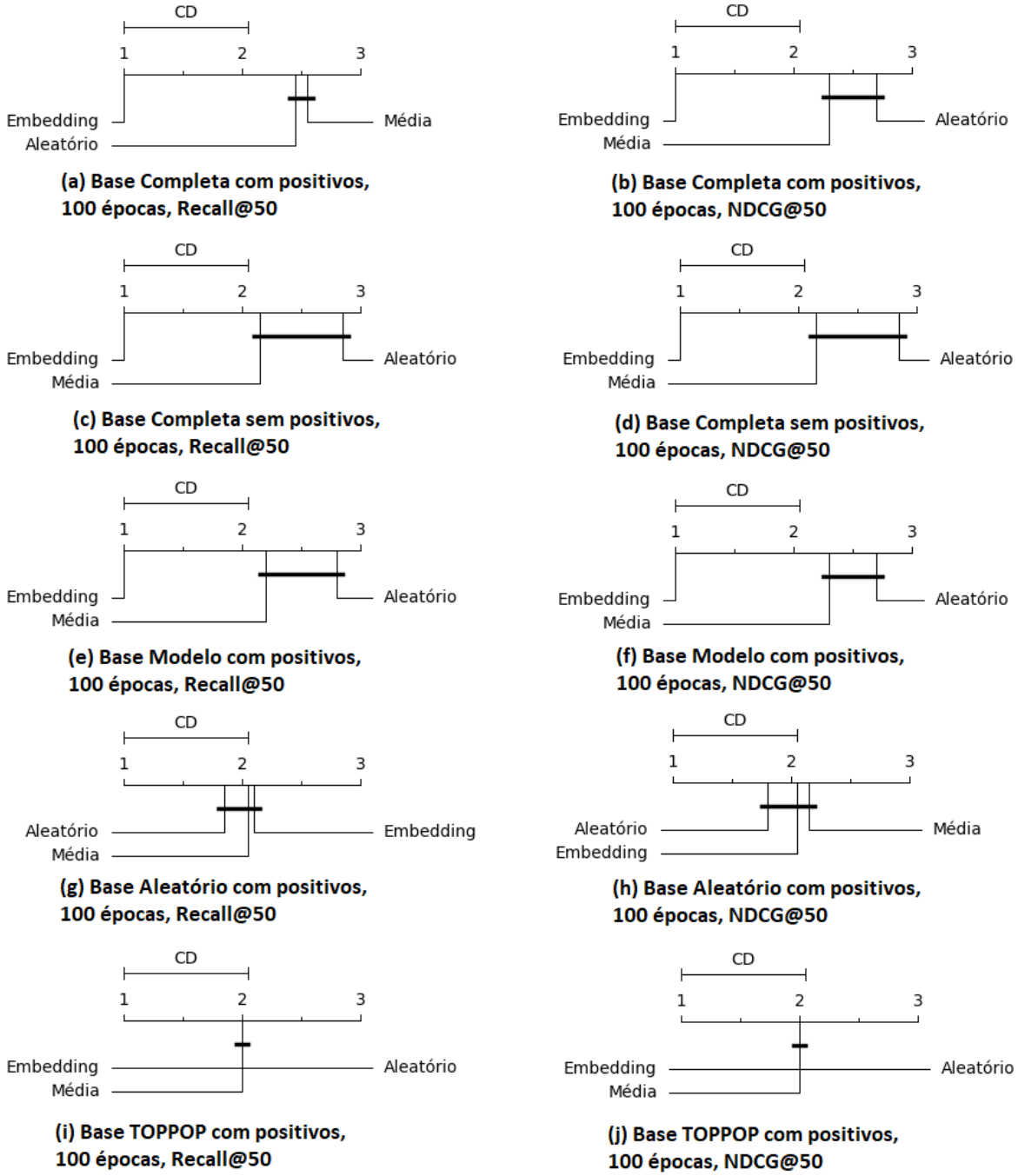


Figura 16 – Resultados para o teste de Nemenyi utilizando 100 épocas e k=50 para cada modelo de recomendação e métrica, com objetivo de comparar os métodos de inicialização.

Fonte: Figura do autor

A Tabela 13 apresenta os *pvalue* resultantes do Teste de *Friendman*, onde cada valor na tabela foi calculado através dos vetores de resultados dos testes utilizando os modelos de Recomendação "Base Completa com positivos" (BCCP), "Base Completa sem positivos" (BCSP), "Base Modelo com positivos" (BMCP), "Base Aleatório com positivos" (BACP) e "Base TOPPOP com positivos"(BTCP). Podemos verificar que para todas as combinações de métodos de inicialização e quantidade de épocas, os valores de *pvalue* obtidos através do Teste de *Friedman* apresentaram valores inferiores a 0,05, o que resulta em diferença significativa em todas as combinações da tabela.

Para identificar onde essa diferença é apresentada, a Figura 17 apresenta os gráficos gerados pelo teste de *Nemenyi*, que servem de complemento para a Tabela 13. Podemos notar que o modelo de recomendação BMCP sempre está em primeiro, porém com resultados estatisticamente iguais a outros modelos.

Métodos de Inicialização	**Métrica**	**10 épocas**	**20 épocas**	**40 épocas**	**80 épocas**	**100 épocas**
Embedding	Recall@10	6,098E-08	4,328E-08	4,328E-08	7,959E-08	7,959E-08
	Recall@20	6,098E-08	6,098E-08	7,959E-08	7,959E-08	7,959E-08
	Recall@30	6,098E-08	6,098E-08	6,098E-08	7,959E-08	7,959E-08
	Recall@40	6,098E-08	6,098E-08	8,588E-08	7,959E-08	7,959E-08
	Recall@50	8,588E-08	6,657E-08	6,098E-08	7,959E-08	7,959E-08
	NDCG@10	4,328E-08	4,328E-08	4,328E-08	4,328E-08	6,098E-08
	NDCG@20	4,328E-08	4,328E-08	4,328E-08	7,959E-08	7,959E-08
	NDCG@30	4,328E-08	4,328E-08	4,328E-08	7,959E-08	7,959E-08
	NDCG@40	4,328E-08	4,328E-08	4,328E-08	7,959E-08	7,959E-08
	NDCG@50	4,328E-08	4,328E-08	4,328E-08	7,959E-08	7,959E-08
Média	Recall@10	6,657E-08	9,853E-08	1,013E-07	3,625E-07	7,158E-08
	Recall@20	4,328E-08	5,196E-08	7,508E-08	9,811E-08	9,588E-08
	Recall@30	5,049E-08	4,328E-08	1,078E-07	9,039E-08	3,008E-07
	Recall@40	6,098E-08	4,328E-08	2,531E-07	4,328E-08	9,580E-08
	Recall@50	6,098E-08	5,196E-08	5,619E-08	6,586E-08	6,140E-08
	NDCG@10	6,098E-08	1,171E-07	1,013E-07	3,625E-07	7,158E-08
	NDCG@20	4,328E-08	4,718E-08	9,039E-08	9,580E-08	8,903E-08
	NDCG@30	6,407E-08	4,328E-08	1,558E-07	9,039E-08	5,024E-07
	NDCG@40	4,328E-08	4,328E-08	3,005E-07	4,328E-08	5,049E-08
	NDCG@50	6,098E-08	4,718E-08	7,205E-08	1,324E-07	5,619E-08
Aleatório	Recall@10	8,320E-08	1,152E-07	9,600E-08	1,152E-07	1,429E-07
	Recall@20	1,635E-07	2,364E-07	2,137E-07	2,863E-07	1,152E-07
	Recall@30	1,758E-07	4,693E-07	3,555E-07	2,601E-07	3,112E-07
	Recall@40	2,660E-07	3,148E-07	4,226E-07	4,068E-07	2,865E-07
	Recall@50	4,329E-07	4,358E-07	1,960E-07	1,910E-07	1,728E-07
	NDCG@10	8,874E-08	1,429E-07	9,600E-08	1,429E-07	1,429E-07
	NDCG@20	3,209E-07	3,567E-07	2,833E-07	2,863E-07	1,477E-07
	NDCG@30	5,238E-07	4,927E-07	3,555E-07	4,551E-07	4,610E-07
	NDCG@40	6,194E-07	3,148E-07	5,367E-07	5,315E-07	2,865E-07
	NDCG@50	6,767E-07	6,446E-07	6,145E-07	2,389E-07	3,277E-07

Tabela 13 – Valores de *pvalue* para o teste de *Friedman* comparando os modelos de recomendação.

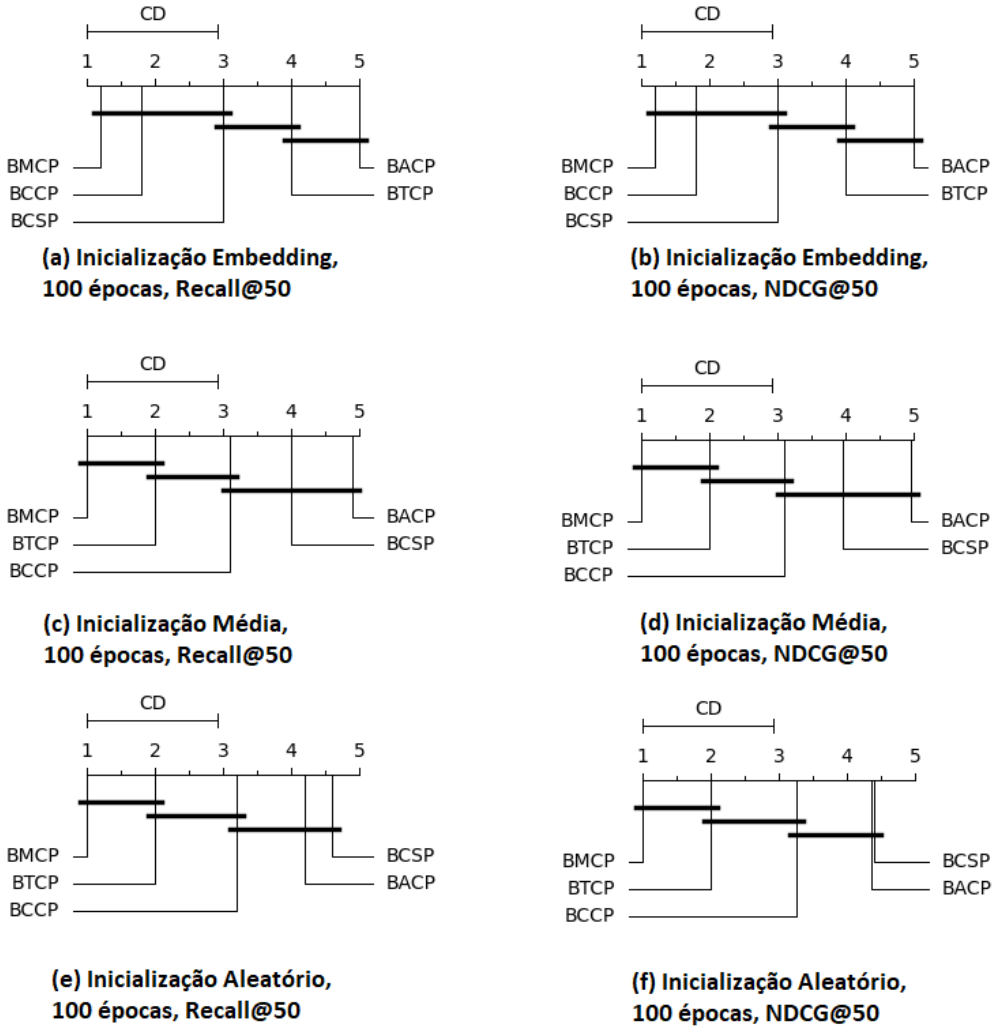


Figura 17 – Resultados para o teste de Nemenyi utilizando 100 épocas e k=50 para cada método de inicialização e métrica, com objetivo de comparar os modelos de recomendação.

Fonte: Figura do autor

A Tabela 14 apresenta os *pvalue* resultantes do Teste de *Friendman*, onde cada valor na tabela foi calculado através dos vetores de resultados dos testes utilizando o treinamento com 10 épocas, 20 épocas, 40 épocas, 80 épocas e 100 épocas. Podemos verificar primeiramente que todos os testes utilizando o modelo de recomendação "Base TOPPOP com positivos"(BTCP) resultaram em NaN (*Not a Number*), o que indica que todos os resultados foram idênticos independente do número de épocas utilizado. Os testes utilizando o modelo de recomendação "Base Aleatório com positivos"(BACP) resultaram em valores superiores a 0,05, o que indica que não há diferença significativa entre os resultados para o uso desse modelo de Recomendação independente da quantidade de épocas. Os demais modelos apresentaram resultados com e sem diferença significativa.

Para visualizar melhor essa diferença, a Figura 18 apresenta os gráficos gerados pelo teste de *Nemenyi*, que servem de complemento para a Tabela 14 apresentando alguns

resultados para a base BMCP. Podemos notar que em alguns casos, os resultados não apresentam diferença significativa, já nos demais, uma maior quantidade de épocas resulta em resultados melhores, porém sem diferença significativa entre os segundos colocados.

Métodos de Inicialização	**Métrica**	**BCCP**	**BCSP**	**BMCP**	**BACP**	**BTCP**
Embedding	Recall@10	4,3E-08	2,4E-07	5,6E-06	4,52E-01	NaN
	Recall@20	4,3E-08	7,6E-08	1,6E-07	9,97E-01	NaN
	Recall@30	8,0E-08	1,2E-07	3,7E-07	8,46E-01	NaN
	Recall@40	4,3E-08	7,8E-08	5,2E-07	8,96E-01	NaN
	Recall@50	4,3E-08	9,9E-08	7,4E-07	8,63E-01	NaN
	NDCG@10	4,3E-08	1,1E-07	8,0E-08	4,52E-01	NaN
	NDCG@20	4,3E-08	8,6E-08	4,3E-08	9,17E-01	NaN
	NDCG@30	4,3E-08	8,0E-08	1,6E-07	7,71E-01	NaN
	NDCG@40	4,3E-08	6,1E-08	1,1E-07	9,80E-01	NaN
	NDCG@50	4,3E-08	6,1E-08	1,5E-07	8,84E-01	NaN
Média	Recall@10	2,0E-04	2,2E-02	8,9E-03	3,0E-01	NaN
	Recall@20	2,5E-04	5,1E-02	3,7E-03	7,6E-01	NaN
	Recall@30	1,1E-03	1,5E-01	4,9E-01	2,8E-01	NaN
	Recall@40	7,9E-05	1,4E-02	1,6E-02	4,8E-01	NaN
	Recall@50	5,8E-05	1,8E-02	3,7E-01	8,1E-01	NaN
	NDCG@10	2,7E-02	6,3E-02	9,6E-03	4,0E-01	NaN
	NDCG@20	1,1E-03	5,4E-01	8,6E-04	8,6E-01	NaN
	NDCG@30	5,2E-04	5,2E-02	2,0E-01	1,1E-01	NaN
	NDCG@40	6,9E-04	1,1E-01	6,8E-02	3,3E-01	NaN
	NDCG@50	1,8E-03	1,6E-01	1,1E-02	5,7E-01	NaN
Aleatório	Recall@10	1,8E-01	1,6E-01	8,1E-01	8,3E-01	NaN
	Recall@20	2,1E-01	8,4E-01	4,3E-01	6,9E-01	NaN
	Recall@30	1,8E-01	6,1E-01	9,7E-02	6,0E-01	NaN
	Recall@40	2,9E-01	3,4E-01	4,9E-01	6,5E-01	NaN
	Recall@50	4,3E-03	5,5E-02	2,8E-01	9,5E-01	NaN
	NDCG@10	2,5E-01	2,0E-01	2,1E-01	7,2E-01	NaN
	NDCG@20	3,6E-01	9,6E-01	1,1E-01	7,8E-01	NaN
	NDCG@30	1,8E-01	5,5E-01	4,0E-02	6,2E-01	NaN
	NDCG@40	3,3E-01	4,5E-01	3,8E-01	6,1E-01	NaN
	NDCG@50	6,0E-03	3,8E-02	7,4E-01	5,9E-01	NaN

Tabela 14 – Valores de *pvalue* para o teste de *Friedman* comparando o uso de diferentes épocas para treinamento. *pvalue* = NaN representa que os vetores testados são idênticos. BCCP representa a Base Completa com positivos, BCSP representa a Base Completa sem positivos, BMCP representa a Base Modelo com positivos, BACP representa a Base Aleatório com positivos, BTCP representa a Base TOPPOP com positivos.

Para cada *pvalue* calculado através do teste de *Friedman*, foi gerado um gráfico do teste de *Nemenyi*, o qual é apresentado no Apêndice A.

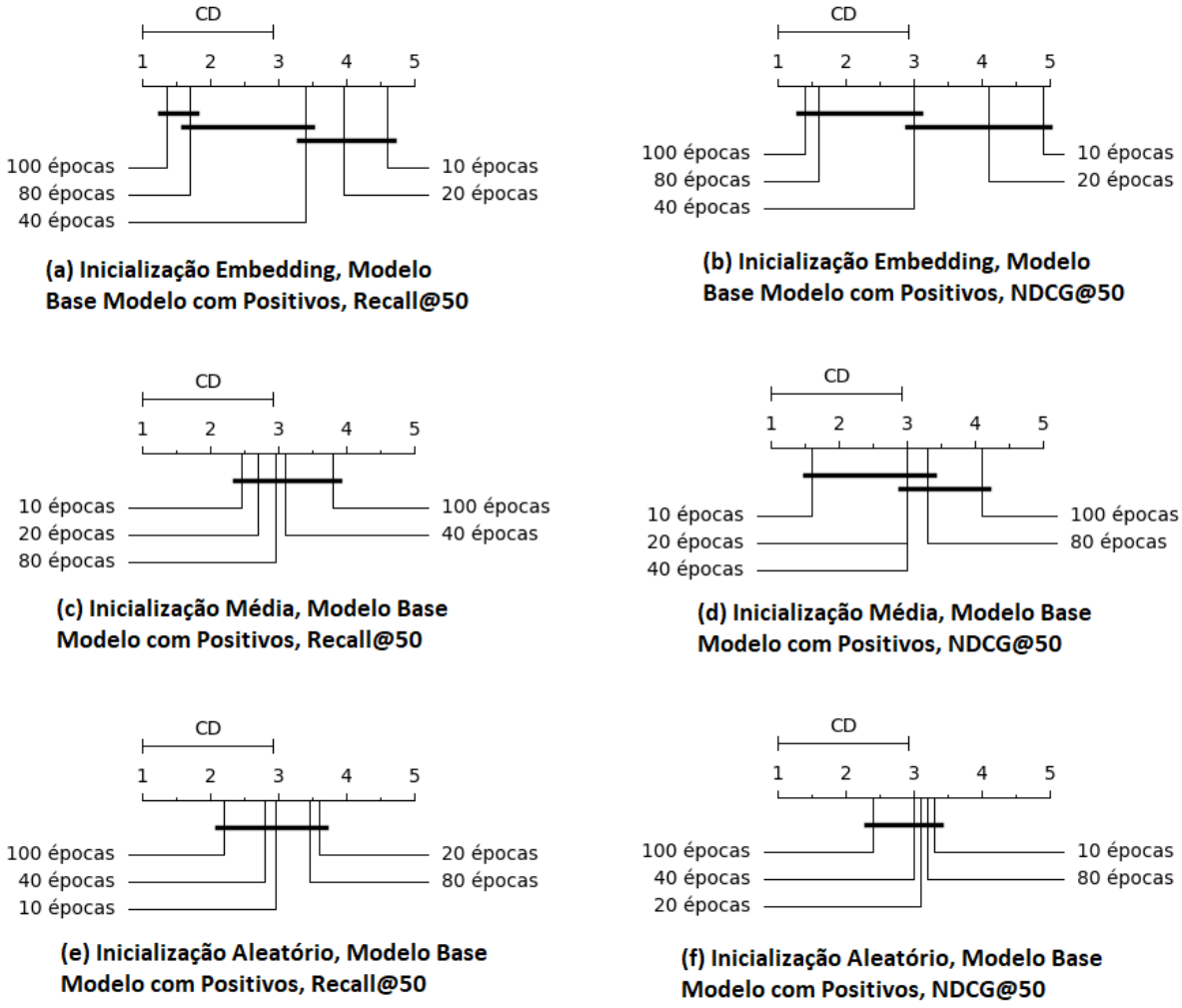


Figura 18 – Resultados para o teste de Nemenyi utilizando o Modelo de Recomendação Base Modelo com Positivos e k=50 para cada método de inicialização e métrica, com objetivo de comparar as épocas de treinamento.

Fonte: Figura do autor

6.2 Resultados com Interação do Usuário

Os resultados obtidos a partir da interação com o usuário foram possíveis por meio da criação de uma aplicação com interface gráfica, o qual é executado na máquina do usuário e que interage com o modelo pré-treinado. O modelo foi treinado utilizando todos os dados descritos na Seção anterior, porém com diferença de que não foi realizada uma validação cruzada, pois o objetivo aqui era apenas treinar e não avaliar o modelo em questão. A aplicação foi treinada utilizando a inicialização por *Embeddings* e utiliza o modelo de recomendação "Base Modelo com Positivos", os quais apresentaram melhores resultados nos experimentos realizados na Seção 6.1, o treinamento foi realizado com 100 épocas e são apresentados ao usuário sempre 20 itens (K = 20). A aplicação foi criada com textos em língua inglesa, devido ao possível compartilhamento dela com usuários de outras nacionalidades. Além da aplicação, foi desenvolvido um guia de utilização em

língua inglesa que explica passo a passo como utilizar a aplicação para obter uma lista de possíveis testes.

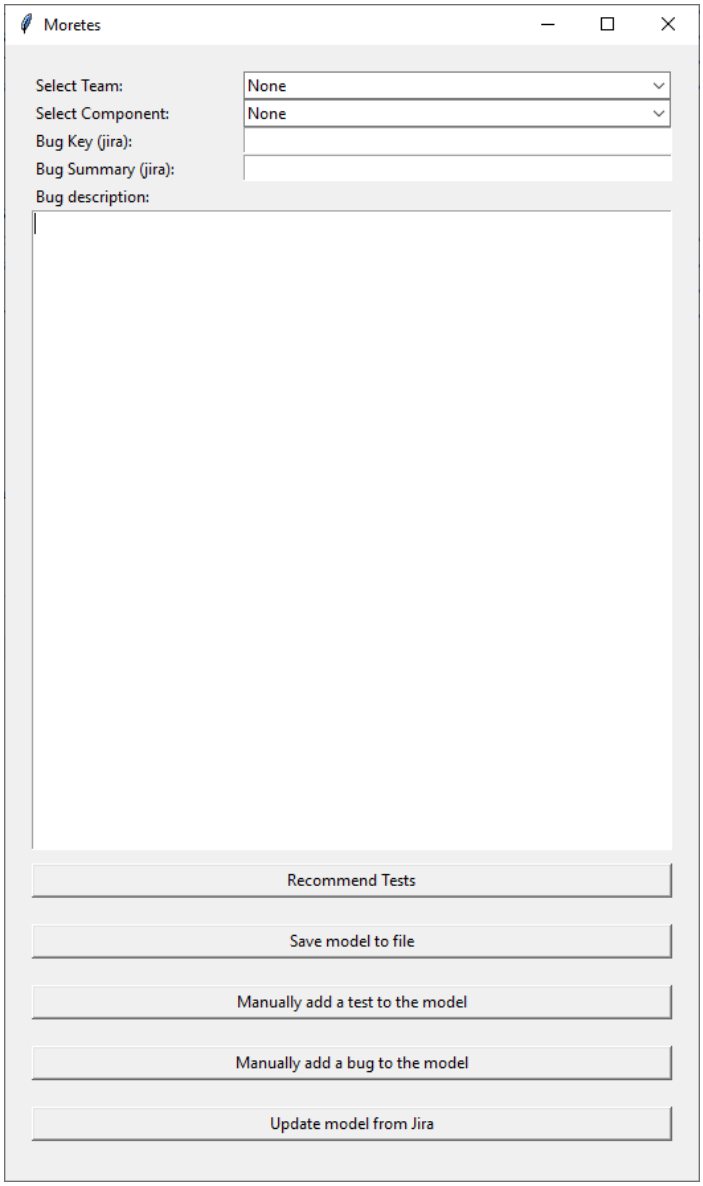


Figura 19 – Tela inicial da aplicação Moretes.

Fonte: Figura do autor

Falando um pouco da aplicação, a Figura 19 apresenta a tela inicial da aplicação denominada "Moretes", onde o usuário pode selecionar a partir de uma lista o Time, selecionar a partir de uma lista o Componente, informar a partir de texto o "*Bug Key*"(identificação da Falha no software Jira), informar o "*Bug Summary*" (breve descrição da falha, algo como título da falha) e informar a "*Bug description*" (descrição da falha). Além disso, o usuário pode escolher cinco possíveis opções de utilização da aplicação, sendo elas: "*Recommend Tests*" (Recomendar teste), "*Save model to file*" (salvar modelo em arquivo), "*Manually add a test to the model*" (adicionar manualmente um teste ao modelo), "*Manually add a bug to the model*" (adicionar manualmente uma falha ao modelo)

e "*Update model from Jira*" (atualizar o modelo a partir do Jira).

Ao clicar no botão "*Recommend Tests*", em um primeiro momento será verificado se o "*Bug Key*" informado pelo usuário representa uma falha que já está no modelo, caso a falha esteja no modelo, será gerada uma lista com vinte testes para que o usuário possa selecionar os Testes Positivos (testes relacionados com a falha), caso a falha não esteja no modelo, ou a informação sobre identificador da falha não seja fornecida, é realizada uma verificação da informação de "*Bug Description*", a qual será transformada em um vetor numérico de 512 posições e será comparada por meio da Similaridade entre os cossenos com todas as demais falhas armazenadas no modelo, a fim de encontrar possíveis similaridades. Caso a descrição informada possua mais que 95% de similaridade com outra falha da base, é informado ao usuário que o modelo já possui uma falha com aquela descrição, e apresenta o identificador da falha encontrada para que o usuário verifique. Caso o usuário informe que é uma nova falha, ou a similaridade encontrada seja inferior ou igual a 95% com alguma outra falha do modelo, o sistema irá criar uma falha seguindo os passos descritos na Seção 5.4, se tornando necessário assim que todas as informações sejam preenchidas para a criação da mesma. Após a nova falha ser criada ou uma falha existente ser selecionada, é gerada uma lista de vinte testes recomendados para aquela falha. A Figura 20 apresenta a tela de recomendação de testes.

Podemos verificar que na Figura 20 é apresentado o identificador do teste e uma breve descrição ou título do mesmo (*Summary*), isso irá auxiliar o usuário a identificar se aquele teste é realmente relacionado a falha informada. O usuário seleciona os testes relevantes (itens selecionados ficam marcados em azul) e clica no botão "*Save*" (salvar). Ao clicar para salvar, o modelo irá realizar a atualização dos pesos dos itens baseado nos passos descritos Seção 5.4.

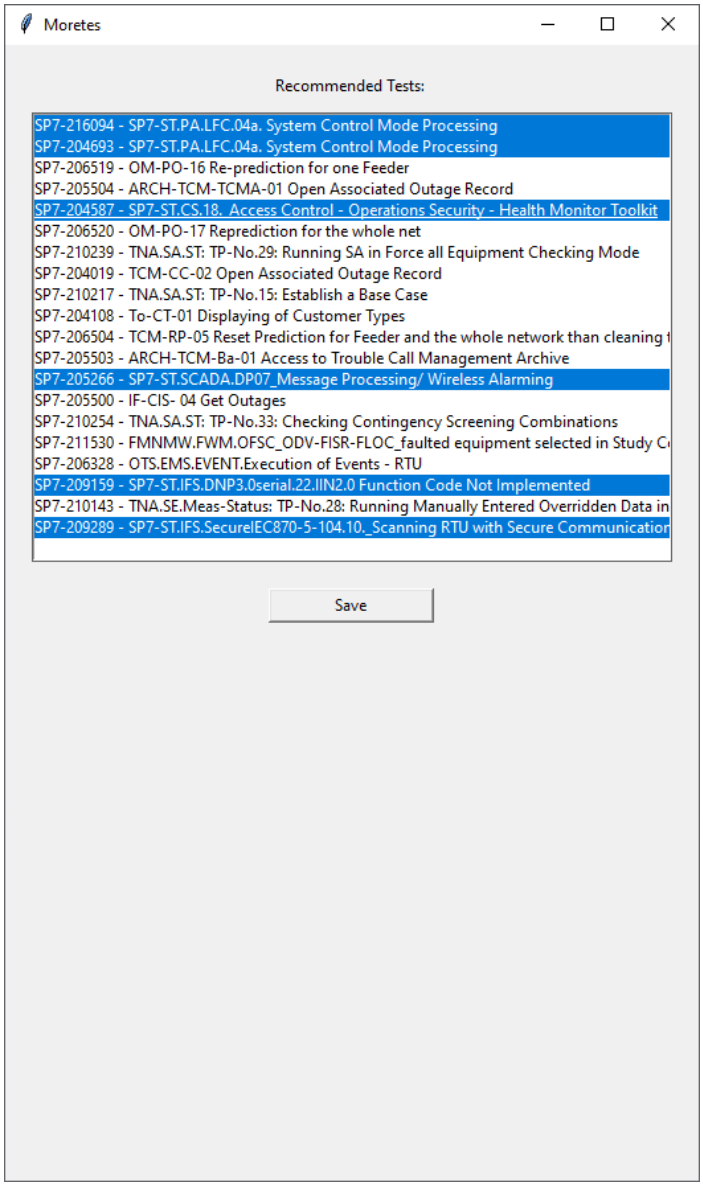


Figura 20 – Tela de testes recomendados.

Fonte: Figura do autor

Após realizar alguma interação com a aplicação, é necessário na tela inicial, clicar no botão de "*Save model to file*", para que o modelo atualizado seja armazenado e possa ser utilizado em futuras utilizações. Para evitar que o modelo seja salvo com interações incorretas, ou até mesmo apresentações da aplicação, o modelo só irá ser atualizado em arquivo quando o usuário clicar nesse botão, caso ele não clique, o modelo não irá ser alterado, ou seja, não irá ser incrementado.

Para que a aplicação não fique defasada, é necessário que os dados de falhas e testes do modelo sejam sempre os mais atualizados possíveis. Para que isso ocorra, foi criada uma tela para atualizar o modelo com as informações do Jira de forma automática. Ao clicar na tela inicial no botão "*Update model from Jira*", você será direcionado a tela de atualização do modelo (Figura 21) a qual irá apresentar os últimos identificadores de

Falhas e Testes que a base possui e o link padrão para o servidor Jira, dados esses que podem ser editados. Além disso, é necessário que o usuário informe sua *API Key*, para que a aplicação tenha autorização para conectar ao servidor do jira com o acesso do usuário. Após a confirmação (botão "*Confirm*"), a aplicação irá executar a conexão com o servidor, processo esse que pode demorar alguns minutos, após os dados serem coletados, eles são inseridos no modelo da mesma forma que é realizada a inserção de novas Falhas, garantindo assim que o modelo possa sempre ter as informações mais atualizadas dos testes e falhas.

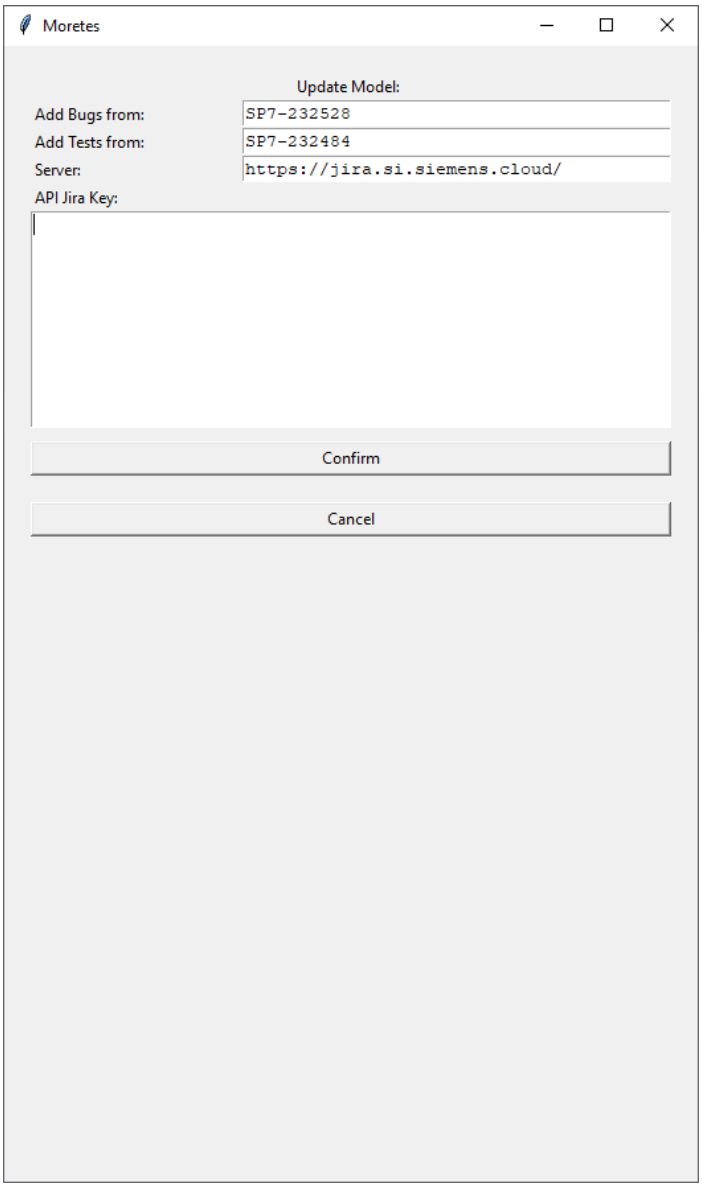


Figura 21 – Tela de atualização de modelo.

Fonte: Figura do autor

Além da atualização automática, podem ser realizadas inserções de Falhas e Testes de forma manual, informando a aplicação as informações necessárias para se criar um objeto no modelo. A Figura 22 apresenta a tela de inserção manual de falhas, onde é necessário informar o time, componente, identificador, descrição breve e descrição completa

da falha que está sendo inserida. Utilizando essas informações uma nova falha é inserida no modelo.

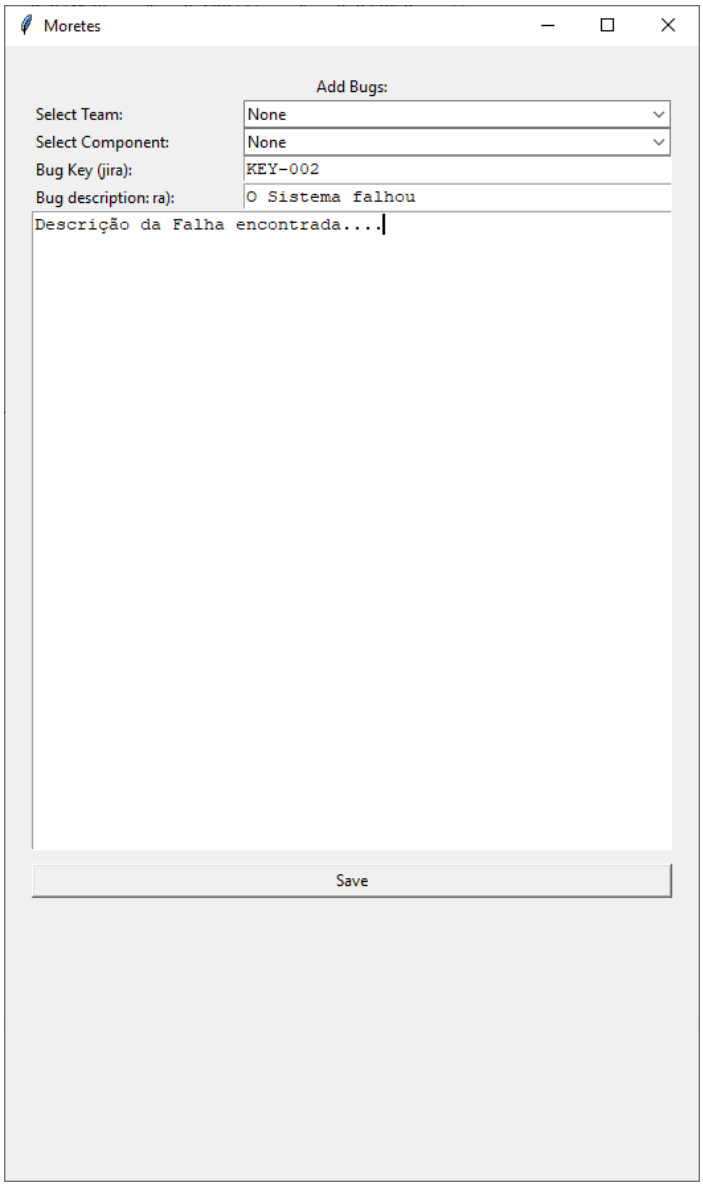


Figura 22 – Tela de inserção manual de falha.

Fonte: Figura do autor

Já na Figura 23 é apresentada a tela de inserção manual de testes, onde é necessário informar o time, componente, identificador, descrição breve do teste que está sendo inserido. Além disso, é necessário inserir os passos para que o teste seja executado, clicando no botão "Add Step" você é direcionado a tela apresentada na Figura 24 a qual é necessário informar os dados de Ação a ser realizada, Dados a serem fornecidos e Resultados esperados para um determinado passo do teste. É possível inserir vários passos para um mesmo teste, clicando sempre em *Add Step*. Após inserir todas as informações, basta salvar o Teste que ele será inserido no modelo. Vale lembrar que as informações são inseridas sempre no modelo carregado na memória, sendo sempre necessário salvar o modelo em disco para que

as informações possam ser utilizadas em outro momento, ou até mesmo compartilhada com outros usuários.

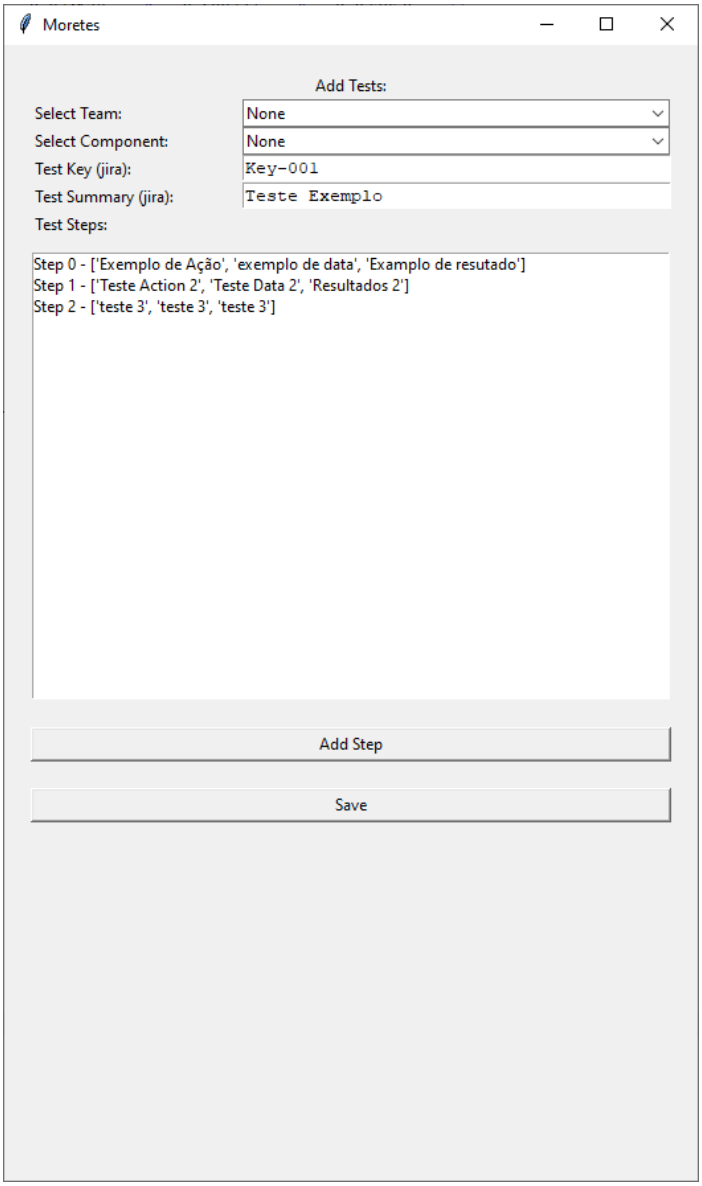


Figura 23 – Tela de inserção manual de testes.

Fonte: Figura do autor

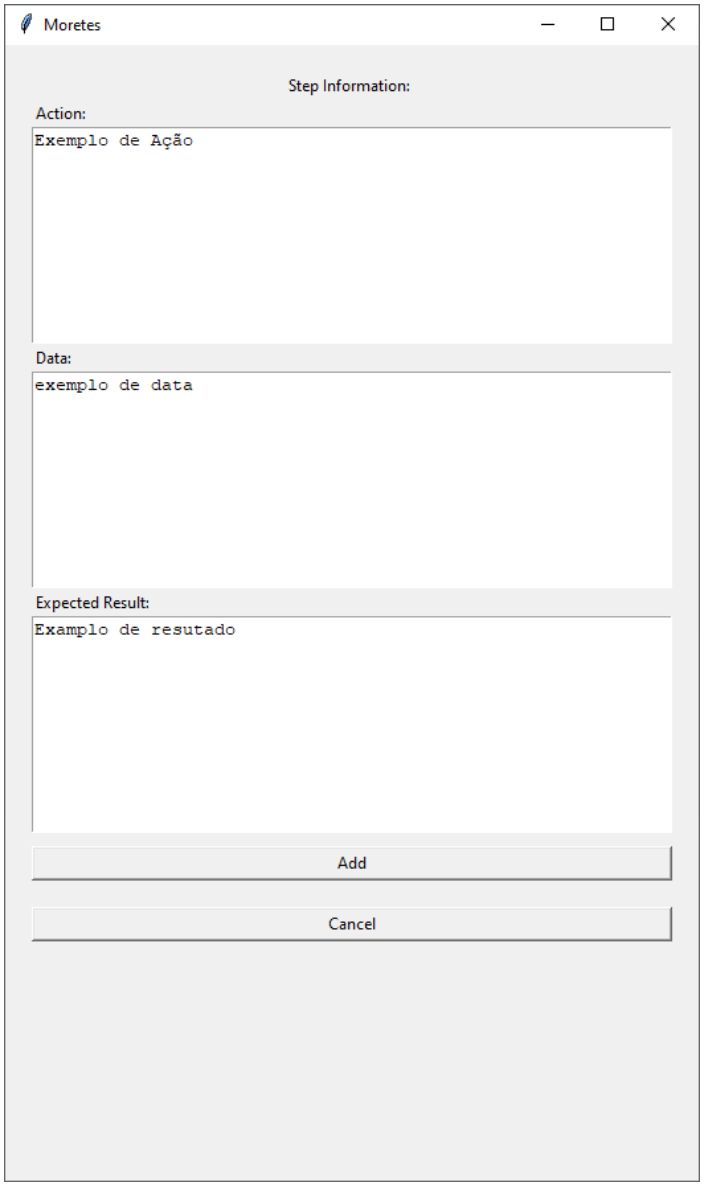


Figura 24 – Tela de inserção Passos de um teste.

Fonte: Figura do autor

A aplicação apresentada foi utilizada por três usuários (testadores indicados pela empresa), os quais realizaram a atualização automática da base por meio da conexão com o Jira e analisaram as recomendações geradas pela aplicação. A base inicial de cada testador não teve influência da interação de outros testadores, devido ao fato de que o modelo é armazenado localmente, com isso garantimos que todos os testadores possuem o mesmo modelo pré-treinado. Os testes foram realizados durante duas semanas em horários ociosos dos testadores, utilizando os dados atualizados do Jira (dados de treinamento acrescidos de dados coletados automaticamente pela ferramenta). Os resultados foram coletados em forma de vetores de interação, onde são apresentados os identificadores das falhas juntamente com duas listas de itens, sendo elas os itens Positivos (selecionados pelo usuário) e os itens Negativos (itens não selecionados). As interações são armazenadas na aplicação

e podem ser extraídas em formato de arquivo de texto (.txt) para compartilhamento.

A Tabela 15 apresenta os resultados obtidos a partir da utilização do usuário final do sistema. Podemos verificar que o *Usuário 2* apresentou uma taxa de acerto inferior aos demais, isso se deve ao fato de que as falhas e testes aos quais o *Usuário 2* é responsável, não foram utilizados para o treinamento do modelo, sendo assim a recomendação gerada para o *Usuário 2* é uma recomendação de Testes que nunca tiveram interações com falhas por meio de uma falha que nunca teve interação com testes, que é o clássico problema de *ColdStart*. Já o *Usuário 3* obteve a melhor taxa de acerto, onde cada recomendação apresentou em média quatro testes relevantes para a falha informada. Isso se deve ao fato de que as falhas informadas pelo *Usuário 3* são falhas que já tiveram interação com itens na base, acelerando assim o acerto do modelo. Podemos citar também que a utilização de termos padronizados entre os passos do teste e a descrição da falha, ajuda o modelo a encontrar testes de forma mais rápida.

Usuário	**Recall@K**	**Média de Testes Recomendados Corretamente**	**Total de Recomendações com acertos**	**Resultado da Avaliação Recall@20**
Usuário 1	Recall@20	1,398268	133 de 231	0,575758
Usuário 2	Recall@20	0,431227	90 de 269	0,334572
Usuário 3	Recall@20	3,634328	236 de 268	0,880597

Tabela 15 – Resultados obtidos a partir da utilização da aplicação por usuários.

Os experimentos a partir da interação com os usuários nos possibilita responder a questão de pesquisa 3, onde podemos verificar que é possível realizar recomendações de testes que nunca interagiram com falhas anteriormente.

7 Conclusão

Por meio dos experimentos realizados, podemos concluir que a utilização de dados textuais descritivos das falhas melhora a taxa de acerto na recomendação de testes em um sistema de recomendação, refutando assim a Hipótese Nula 01 (H01), e aceitando a Hipótese Alternativa 01 (HA1), descrita na seção 1.5, como verdadeira. Os dados textuais descritivos das falhas e os dados textuais contendo o passo a passo para a execução de cada teste, foram utilizados como valores de inicialização dos pesos dos itens, pesos esses que foram gerados com o *Sentence-BERT*, provando por meio dos testes realizados, tanto com os dados coletados como com as interações com usuários, que o uso de *transformers* para inicialização dos pesos de um sistema de recomendação é uma possível alternativa para contornar problemas relacionados a *ColdStart*, refutando assim que a Hipótese Nula 03 (H03), e aceitando a Hipótese Alternativa 03 (HA3) como verdadeira.

Com base na tabela 15, podemos destacar que Usuários que tiveram as piores avaliações, são aqueles que solicitaram recomendações para falhas que não tinham nenhuma ligação com testes. Apesar da avaliação ser menor do que a avaliação dos demais usuários, podemos considerar que a Hipótese Nula 02 (H02) foi refutada, aceitando assim a Hipótese Alternativa 02 (HA2) como verdadeira.

Outro ponto importante que devemos ressaltar, é que nesse trabalho os dados utilizados durante a pesquisa são dados reais, de um software em desenvolvimento e aprimoramento contínuo, onde atualmente a seleção de casos de teste é realizada por testadores, sem o auxílio do modelo de recomendação, e que futuramente será utilizado o modelo de recomendação de testes durante o dia a dia do desenvolvimento e solução de falhas.

Os objetivos definidos na seção 1.2 foram alcançados. Na seção 5 é apresentado o método para seleção de casos de testes manuais utilizando o sistema de recomendação, o qual foi definido como objetivo principal do projeto. Os objetivos específicos gerar uma lista de recomendações que apresente testes relevantes para o testador, recomendar testes que não seriam recomendados em seleções baseadas em ocorrência de falha, criação de um modelo incremental que possa ser atualizado com novas informações e a geração de recomendações para qualquer falha independente de sua origem ou descrição foram alcançados por meio da aplicação que foi utilizada para os experimentos com o usuário final do sistema.

Como trabalhos futuros podemos listar a atualização da aplicação para que seja possível o usuário informar possíveis testes que são conhecidos como relevantes, para otimizar a primeira recomendação de uma determinada falha nova. Além disso, transformar

a aplicação que roda com dados locais em um sistema compartilhado por vários usuários, possibilitando assim que o aprendizado seja acelerado e as recomendações mais precisas. Além da atualização da aplicação, realizar testes com diferentes parâmetros, diferentes modelos base de recomendação e abordagens de inicialização, comparando os resultados para determinar qual a melhor combinação.

Referências

ABDELKARIM, M.; ELADAWI, R. Tcp-net: Test case prioritization using end-to-end deep neural networks. In: *15th IEEE International Conference on Software Testing, Verification and Validation Workshops ICST Workshops 2022, Valencia, Spain, April 4-13, 2022*. IEEE, 2022. p. 122–129. Disponível em: <https://doi.org/10.1109/ICSTW55395.2022.00034>. Citado 3 vezes nas páginas 22, 50 e 51.

ABELE, S.; GÖHNER, P. Improving proceeding test case prioritization with learning software agents. In: DUVAL, B.; HERIK, H. J. van den; LOISEAU, S.; FILIPE, J. (Ed.). *ICAART 2014 - Proceedings of the 6th International Conference on Agents and Artificial Intelligence, Volume 2, ESEO, Angers, Loire Valley, France, 6-8 March, 2014*. SciTePress, 2014. p. 293–298. Disponível em: <https://doi.org/10.5220/0004920002930298>. Citado 3 vezes nas páginas 21, 47 e 51.

ACHIAM, J.; ADLER, S.; AGARWAL, S.; AHMAD, L.; AKKAYA, I.; ALEMAN, F. L.; ALMEIDA, D.; ALTENSCHMIDT, J.; ALTMAN, S.; ANADKAT, S. et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. Citado na página 24.

ACM Digital Library. <https://dl.acm.org/>. Acesso: 30-12-2024. Citado na página 41.

AL-SABBAGH, K. W.; STARON, M.; HEBIG, R.; MEDING, W. Predicting test case verdicts using textual analysis of committed code churns. In: TARHAN, A. K.; COSKUNÇAY, A. (Ed.). *Joint Proceedings of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM Mensura 2019), Haarlem, The Netherlands, October 7-9, 2019*. CEUR-WS.org, 2019. (CEUR Workshop Proceedings, v. 2476), p. 138–153. Disponível em: <http://ceur-ws.org/Vol-2476/paper7.pdf>. Citado 3 vezes nas páginas 22, 48 e 51.

ASHWIN, R.; YIYANG, Z.; JIANXIN, J. A text mining-based recommendation system for customer decision making in online product customization. *IEEE*, 2006. Citado na página 52.

AZIZI, M. A tag-based recommender system for regression test case prioritization. In: IEEE. *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. [S.l.], 2021. p. 146–157. Citado na página 45.

AZIZI, M.; DO, H. A collaborative filtering recommender system for test case prioritization in web applications. In: *Proceedings of the 33rd annual ACM symposium on applied computing*. [S.l.: s.n.], 2018. p. 1560–1567. Citado na página 45.

BAGHERZADEH, M.; KAHANI, N.; BRIAND, L. C. Reinforcement learning for test case prioritization. *CoRR*, abs/2011.01834, 2021. Disponível em: <https://arxiv.org/abs/2011.01834>. Citado 3 vezes nas páginas 21, 49 e 51.

BUCHGEHER, G.; ERNSTBRUNNER, C.; RAMLER, R.; LUSSER, M. Towards tool-support for test case selection in manual regression testing. In: *Sixth IEEE International Conference on Software Testing, Verification and Validation, ICST 2013 Workshops Proceedings, Luxembourg, Luxembourg, March 18-22, 2013*. IEEE Computer

Society, 2013. p. 74–79. Disponível em: <https://doi.org/10.1109/ICSTW.2013.16>. Citado na página 22.

CHANG, P.; DRMANAC, D. G.; WANG, L. Online selection of effective functional test programs based on novelty detection. In: SCHEFFER, L.; PHILLIPS, J. R.; HU, A. J. (Ed.). *2010 International Conference on Computer-Aided Design, ICCAD 2010, San Jose, CA, USA, November 7-11, 2010*. IEEE, 2010. p. 762–769. Disponível em: <https://doi.org/10.1109/ICCAD.2010.5653868>. Citado 2 vezes nas páginas 46 e 51.

CHEN, S.; CHEN, Z.; ZHAO, Z.; XU, B.; FENG, Y. Using semi-supervised clustering to improve regression test selection techniques. In: *Fourth IEEE International Conference on Software Testing, Verification and Validation, ICST 2011, Berlin, Germany, March 21-25, 2011*. IEEE Computer Society, 2011. p. 1–10. Disponível em: <https://doi.org/10.1109/ICST.2011.38>. Citado 3 vezes nas páginas 22, 46 e 51.

COTRONEO, D.; PIETRANTUONO, R.; RUSSO, S. A learning-based method for combining testing techniques. In: NOTKIN, D.; CHENG, B. H. C.; POHL, K. (Ed.). *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*. IEEE Computer Society, 2013. p. 142–151. Disponível em: <https://doi.org/10.1109/ICSE.2013.6606560>. Citado 3 vezes nas páginas 22, 47 e 51.

DASH, B. B.; DE, U. C.; BEHERA, T. M.; SAMANT, T.; BANERJEE, S. Recommendation system for e-commerce apparel stores based on text-semantics. In: IEEE. *2023 2nd International Conference for Innovation in Technology (INOCON)*. [S.l.], 2023. p. 1–5. Citado na página 52.

DEVLIN, J.; CHANG, M.; LEE, K.; TOUTANOVA, K. BERT: pre-training of deep bidirectional transformers for language understanding. In: BURSTEIN, J.; DORAN, C.; SOLORIO, T. (Ed.). *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019. p. 4171–4186. Disponível em: <https://doi.org/10.18653/v1/n19-1423>. Citado 2 vezes nas páginas 24 e 33.

ELAHI, M.; RICCI, F.; RUBENS, N. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*, Elsevier, v. 20, p. 29–50, 2016. Citado 3 vezes nas páginas 35, 36 e 37.

FOWLER, M. *Continuous integration*. 2006. Citado na página 31.

GLIGORIC, M.; NEGARA, S.; LEGUNSEN, O.; MARINOV, D. An empirical evaluation and comparison of manual and automated test selection. In: *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*. [S.l.: s.n.], 2014. p. 361–372. Citado na página 23.

Google Shared Dataset of Test Suite Results. <https://bitbucket.org/HelgeS/atcs-data/src/master/>. Acesso: 30-08-2024. Citado na página 48.

HASNAIN, M.; JEONG, S. R.; PASHA, M. F.; GHANI, I. An ontology based test case prioritization approach in regression testing. *Computers, Materials and Continua*, Tech Science Press, v. 67, n. 1, p. 1051–1068, 2021. Citado 2 vezes nas páginas 49 e 51.

HELM, J. M.; SWIERGOSZ, A. M.; HAEBERLE, H. S.; KARNUTA, J. M.; SCHAFFER, J. L.; KREBS, V. E.; SPITZER, A. I.; RAMKUMAR, P. N. Machine learning and artificial intelligence: definitions, applications, and future directions. *Current reviews in musculoskeletal medicine*, Springer, v. 13, p. 69–76, 2020. Citado na página 31.

HEMMATI, H.; BRIAND, L. C.; ARCURI, A.; ALI, S. An enhanced test case selection approach for model-based testing: an industrial case study. In: ROMAN, G.; HOEK, A. van der (Ed.). *Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2010, Santa Fe, NM, USA, November 7-11, 2010*. ACM, 2010. p. 267–276. Disponível em: <https://doi.org/10.1145/1882291.1882331>. Citado 3 vezes nas páginas 22, 46 e 51.

HUANG, Y.; SHU, T.; DING, Z. A learn-to-rank method for model-based regression test case prioritization. *IEEE Access*, v. 9, p. 16365–16382, 2021. Disponível em: <https://doi.org/10.1109/ACCESS.2021.3053163>. Citado 2 vezes nas páginas 49 e 51.

IEEE. IEEE standard for software test documentation. *IEEE Std 829-1983*, p. 1–48, 1983. Citado 4 vezes nas páginas 21, 29, 30 e 31.

IEEE Xplore. <https://ieeexplore.ieee.org/Xplore/home.jsp>. Acesso: 30-12-2024. Citado na página 41.

ISINKAYE, F. O.; FOLAJIMI, Y. O.; OJOKOH, B. A. Recommendation systems: Principles, methods and evaluation. *Egyptian informatics journal*, Elsevier, v. 16, n. 3, p. 261–273, 2015. Citado na página 34.

JAHAN, H.; FENG, Z.; MAHMUD, S. M. H.; DONG, P. Version specific test case prioritization approach based on artificial neural network. *J. Intell. Fuzzy Syst.*, v. 36, n. 6, p. 6181–6194, 2019. Disponível em: <https://doi.org/10.3233/JIFS-181998>. Citado 3 vezes nas páginas 22, 48 e 51.

JIRA. <https://www.atlassian.com/software/jira>. Acesso: 30-12-2024. Citado na página 56.

KLAMMER, C.; RAMLER, R. A journey from manual testing to automated test generation in an industry project. In: *2017 IEEE International Conference on Software Quality, Reliability and Security Companion, QRS-C 2017, Prague, Czech Republic, July 25-29, 2017*. IEEE, 2017. p. 591–592. Disponível em: <https://doi.org/10.1109/QRS-C.2017.108>. Citado na página 23.

LACHMANN, R.; SCHULZE, S.; NIEKE, M.; SEIDL, C.; SCHAEFER, I. System-level test case prioritization using machine learning. In: *15th IEEE International Conference on Machine Learning and Applications, ICMLA 2016, Anaheim, CA, USA, December 18-20, 2016*. IEEE Computer Society, 2016. p. 361–368. Disponível em: <https://doi.org/10.1109/ICMLA.2016.0065>. Citado 3 vezes nas páginas 22, 47 e 51.

LEVI, A.; MOKRYN, O.; DIOT, C.; TAFT, N. Finding a needle in a haystack of reviews: cold start context-based hotel recommender system. In: *Proceedings of the sixth ACM conference on Recommender systems*. [S.l.: s.n.], 2012. p. 115–122. Citado na página 53.

LIN, L.-J. *Reinforcement learning for robots using neural networks*. [S.l.]: Carnegie Mellon University, 1992. Citado na página 49.

LIU, J.; WEI, Z.; LI, Z.; MAO, X.; WANG, J.; WEI, Z.; ZHANG, Q. Sam: A self-adaptive attention module for context-aware recommendation system. In: IEEE. *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2024. p. 7775–7779. Citado na página 52.

LOPS, P.; GEMMIS, M. D.; SEMERARO, G. Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, Springer, p. 73–105, 2011. Citado na página 36.

LOU, Y.; CHEN, J.; ZHANG, L.; HAO, D. Chapter one - A survey on regression test-case prioritization. *Adv. Comput.*, v. 113, p. 1–46, 2019. Disponível em: <https://doi.org/10.1016/bs.adcom.2018.10.001>. Citado 2 vezes nas páginas 21 e 22.

LOUSADA, J.; RIBEIRO, M. Reinforcement learning for test case prioritization. *CoRR*, abs/2012.11364, 2020. Disponível em: <https://arxiv.org/abs/2012.11364>. Citado 2 vezes nas páginas 48 e 51.

LUO, G. Research on user profile multi model fusion recommendation system based on spark. In: IEEE. *2024 7th International Conference on Artificial Intelligence and Big Data (ICAIBD)*. [S.l.], 2024. p. 186–193. Citado na página 53.

LUO, L. Software testing techniques. *Institute for software research international Carnegie mellon university Pittsburgh, PA*, v. 15232, n. 1-19, p. 19, 2001. Citado na página 30.

MNIH, A.; SALAKHUTDINOV, R. R. Probabilistic matrix factorization. *Advances in neural information processing systems*, v. 20, 2007. Citado na página 69.

MYERS, G. J. *The art of software testing (2. ed.)*. Wiley, 2004. ISBN 978-0-471-46912-4. Disponível em: <http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0471469122.html>. Citado na página 29.

MYERS, G. J.; SANDLER, C.; BADGETT, T. *The art of software testing*. [S.l.]: John Wiley & Sons, 2011. Citado na página 21.

OLSTHOORN, M.; PANICHELLA, A. Multi-objective test case selection through linkage learning-based crossover. In: O'REILLY, U.; DEVROEY, X. (Ed.). *Search-Based Software Engineering - 13th International Symposium, SSBSE 2021, Bari, Italy, October 11-12, 2021, Proceedings*. Springer, 2021. (Lecture Notes in Computer Science, v. 12914), p. 87–102. Disponível em: <https://doi.org/10.1007/978-3-030-88106-1_7>. Citado 2 vezes nas páginas 49 e 51.

OMRI, S.; SINZ, C. Learning to rank for test case prioritization. In: *15th IEEE/ACM International Workshop on Search-Based Software Testing, SBST@ICSE 2022, Pittsburgh, PA, USA, May 9, 2022*. IEEE, 2022. p. 16–24. Disponível em: <https://doi.org/10.1145/3526072.3527525>. Citado 2 vezes nas páginas 50 e 51.

PLASMA - most MIPS I(TM) opcodes. <https://opencores.org/projects/plasma>. Acesso: 30-08-2024. Citado na página 46.

REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2019. Disponível em: <https://arxiv.org/abs/1908.10084>. Citado na página 34.

RENDLE, S.; FREUDENTHALER, C.; GANTNER, Z.; SCHMIDT-THIEME, L. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012. Citado 2 vezes nas páginas 39 e 70.

RICCI, F.; ROKACH, L.; SHAPIRA, B. Recommender systems: Techniques, applications, and challenges. *Recommender systems handbook*, Springer, p. 1–35, 2021. Citado na página 34.

ROSENBAUER, L.; STEIN, A.; MAIER, R.; PÄTZEL, D.; HÄHNER, J. XCS as a reinforcement learning approach to automatic test case prioritization. In: COELLO, C. A. C. (Ed.). *GECCO '20: Genetic and Evolutionary Computation Conference, Companion Volume, Cancún, Mexico, July 8-12, 2020*. ACM, 2020. p. 1798–1806. Disponível em: <https://doi.org/10.1145/3377929.3398128>. Citado 2 vezes nas páginas 49 e 51.

ROSENBAUER, L.; STEIN, A.; PÄTZEL, D.; HÄHNER, J. XCSF for automatic test case prioritization. In: GUERVÓS, J. J. M.; GARIBALDI, J. M.; WAGNER, C.; BÄCK, T.; MADANI, K.; WARWICK, K. (Ed.). *Proceedings of the 12th International Joint Conference on Computational Intelligence, IJCCI 2020, Budapest, Hungary, November 2-4, 2020*. SCITEPRESS, 2020. p. 49–58. Disponível em: <https://doi.org/10.5220/0010105700490058>. Citado 2 vezes nas páginas 49 e 51.

ROSENBAUER, L.; STEIN, A.; PÄTZEL, D.; HÄHNER, J. XCSF with experience replay for automatic test case prioritization. In: *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020, Canberra, Australia, December 1-4, 2020*. IEEE, 2020. p. 1307–1314. Disponível em: <https://doi.org/10.1109/SSCI47803.2020.9308379>. Citado 2 vezes nas páginas 49 e 51.

ROSERO, R. H.; GÓMEZ, O. S.; RAFAEL, G. D. R. Regression testing of database applications under an incremental software development setting. *IEEE Access*, v. 5, p. 18419–18428, 2017. Disponível em: <https://doi.org/10.1109/ACCESS.2017.2749502>. Citado 4 vezes nas páginas 21, 22, 47 e 51.

ROZA, E. A. da; LIMA, J. A. P.; SILVA, R. C.; VERGILIO, S. R. Machine learning regression techniques for test case prioritization in continuous integration environment. In: *IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2022, Honolulu, HI, USA, March 15-18, 2022*. IEEE, 2022. p. 196–206. Disponível em: <https://doi.org/10.1109/SANER53432.2022.00034>. Citado 2 vezes nas páginas 50 e 51.

SAAKI, M.; HOSSEINI, S.; RAHMANI, S.; KANGAVARI, M. R.; HUA, W.; ZHOU, X. Value-wise convnet for transformer models: an infinite time-aware recommender system. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 35, n. 10, p. 9932–9945, 2022. Citado na página 52.

SCHAFER, J. B.; FRANKOWSKI, D.; HERLOCKER, J.; SEN, S. Collaborative filtering recommender systems. In: *The adaptive web: methods and strategies of web personalization*. [S.l.]: Springer, 2007. p. 291–324. Citado na página 35.

SCHWARZER, M.; BREITINGER, C.; SCHUBOTZ, M.; MEUSCHKE, N.; GIPP, B. Citolytics: A link-based recommender system for wikipedia. In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. [S.l.: s.n.], 2017. p. 360–361. Citado na página 52.

SCIENCEDIRECT.COM. <https://www.sciencedirect.com/>. Acesso: 30-12-2024. Citado na página 41.

SCOPUS preview. <https://www.scopus.com/>. Acesso: 30-08-2024. Citado na página 42.

SHARIF, A.; MARIJAN, D.; LIAAEN, M. Deeporder: Deep learning for test case prioritization in continuous integration testing. In: *IEEE International Conference on Software Maintenance and Evolution, ICSME 2021, Luxembourg, September 27 - October 1, 2021*. IEEE, 2021. p. 525–534. Disponível em: <https://doi.org/10.1109/ICSME52107.2021.00053>. Citado 2 vezes nas páginas 50 e 51.

SHARMA, S.; SHAKYA, H. K. Recommendation system for movies using improved version of som with hybrid filtering methods. In: IEEE. *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*. [S.l.], 2023. p. 1–7. Citado na página 53.

SIEMENS Brasil. <https://new.siemens.com/br/pt.html>. Acesso: 30-12-2024. Citado na página 55.

SIMONS, C.; PARAISO, E. C. Regression test cases prioritization using failure pursuit sampling. In: *10th International Conference on Intelligent Systems Design and Applications, ISDA 2010, November 29 - December 1, 2010, Cairo, Egypt*. IEEE, 2010. p. 923–928. Disponível em: <https://doi.org/10.1109/ISDA.2010.5687069>. Citado na página 22.

Software-artifact Infrastructure Repository. <https://sir.csc.ncsu.edu/portal/index.php>. Acesso: 30-08-2024. Citado na página 47.

SPIEKER, H.; GOTLIEB, A.; MARIJAN, D.; MOSSIGE, M. Reinforcement learning for automatic test case prioritization and selection in continuous integration. In: BULTAN, T.; SEN, K. (Ed.). *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, Santa Barbara, CA, USA, July 10 - 14, 2017*. ACM, 2017. p. 12–22. Disponível em: <https://doi.org/10.1145/3092703.3092709>. Citado 4 vezes nas páginas 22, 48, 49 e 51.

TANG, M.; CUI, S.; JIN, Z.; LIANG, S.-n.; LI, C.; ZOU, L. Sequential recommendation by reprogramming pretrained transformer. *Information Processing & Management*, Elsevier, v. 62, n. 1, p. 103938, 2024. Citado na página 51.

TONELLA, P.; AVESANI, P.; SUSI, A. Using the case-based ranking methodology for test case prioritization. In: *22nd IEEE International Conference on Software Maintenance (ICSM 2006), 24-27 September 2006, Philadelphia, Pennsylvania, USA*. IEEE Computer Society, 2006. p. 123–133. Disponível em: <https://doi.org/10.1109/ICSM.2006.74>. Citado 4 vezes nas páginas 22, 23, 46 e 51.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. *CoRR*, abs/1706.03762, 2017. Disponível em: <http://arxiv.org/abs/1706.03762>. Citado 2 vezes nas páginas 24 e 32.

VINISKI, A. D.; BARDDAL, J. P.; JR, A. de S. B.; ENEMBRECK, F.; CAMPOS, H. V. A. de. A case study of batch and incremental recommender systems in supermarket data under concept drifts and cold start. *Expert Systems with Applications*, Elsevier, v. 176, p. 114890, 2021. Citado 2 vezes nas páginas 22 e 53.

WANG, F.; YANG, S.-C.; YANG, Y.-L. Regression testing based on neural networks and program slicing techniques. In: *Practical Applications of Intelligent Systems*. [S.l.]: Springer, 2011. p. 409–418. Citado 4 vezes nas páginas 21, 22, 46 e 51.

WANG, H.; AMAGATA, D.; MAKEAWA, T.; HARA, T.; HAO, N.; YONEKAWA, K.; KUROKAWA, M. A dnn-based cross-domain recommender system for alleviating cold-start problem in e-commerce. *IEEE Open Journal of the Industrial Electronics Society*, IEEE, v. 1, p. 194–206, 2020. Citado na página 53.

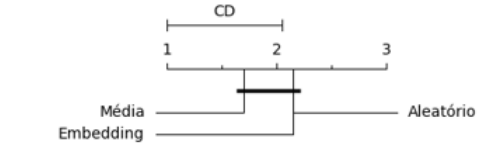
WANG, K.; ZHU, C.; ÇELIK, A.; KIM, J.; BATORY, D. S.; GLIGORIC, M. Towards refactoring-aware regression test selection. In: CHAUDRON, M.; CRNKOVIC, I.; CHECHIK, M.; HARMAN, M. (Ed.). *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*. ACM, 2018. p. 233–244. Disponível em: <https://doi.org/10.1145/3180155.3180254>. Citado 2 vezes nas páginas 48 e 51.

YARAGHI, A. S.; BAGHERZADEH, M.; KAHANI, N.; BRIAND, L. C. Scalable and accurate test case prioritization in continuous integration contexts. *IEEE Trans. Software Eng.*, v. 49, n. 4, p. 1615–1639, 2023. Disponível em: <https://doi.org/10.1109/TSE.2022.3184842>. Citado 2 vezes nas páginas 50 e 51.

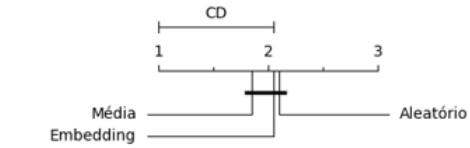
YU, T.; WANG, T. A study of regression test selection in continuous integration environments. In: IEEE. *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. [S.l.], 2018. p. 135–143. Citado 2 vezes nas páginas 22 e 23.

Apêndices

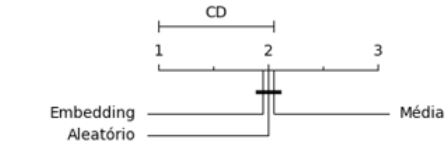
APÊNDICE A – Testes Estatísticos de *Friedman* e *Nemenyi*



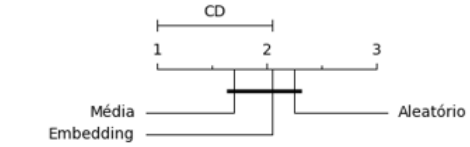
Modelo: Base Aleatório com positivos
Épocas:10
Métrica: Recall@10
Friedman pvalue = 0,223130160148428



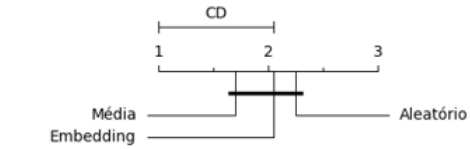
Modelo: Base Aleatório com positivos
Épocas:10
Métrica: Recall @20
Friedman pvalue = 0,66248013539392



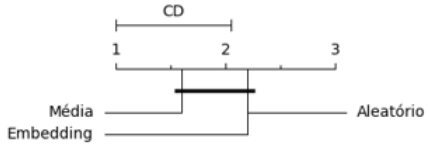
Modelo: Base Aleatório com positivos
Épocas:10
Métrica: Recall @30
Friedman pvalue = 0,969233234476337



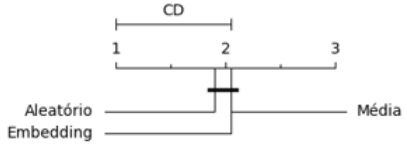
Modelo: Base Aleatório com positivos
Épocas:10
Métrica: Recall @40
Friedman pvalue = 0,343363537415406



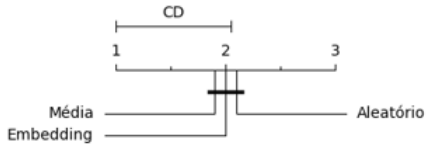
Modelo: Base Aleatório com positivos
Épocas:10
Métrica: Recall @50
Friedman pvalue = 0,355818918537339



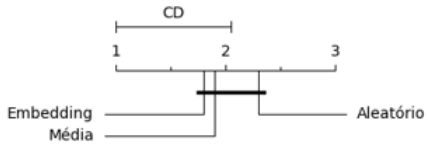
Modelo: Base Aleatório com positivos
Épocas:10
Métrica: NDCG@10
Friedman pvalue = 0,090717953289412



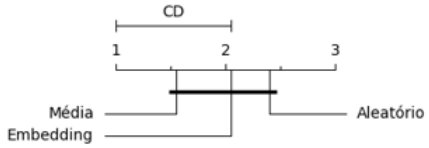
Modelo: Base Aleatório com positivos
Épocas:10
Métrica: NDCG@20
Friedman pvalue = 0,886920436717153



Modelo: Base Aleatório com positivos
Épocas:10
Métrica: NDCG@30
Friedman pvalue = 0,897530560854459



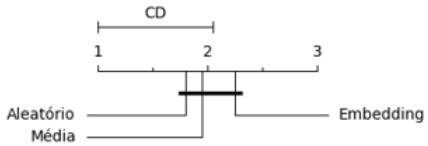
Modelo: Base Aleatório com positivos
Épocas:10
Métrica: NDCG@40
Friedman pvalue = 0,438879929791553



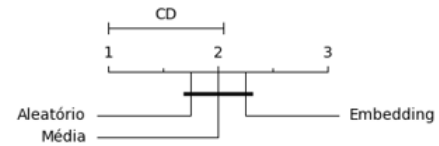
Modelo: Base Aleatório com positivos
Épocas:10
Métrica: NDCG@50
Friedman pvalue = 0,14645272020652

Figura 25 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação "Base Aleatório com Positivos", 10 Épocas

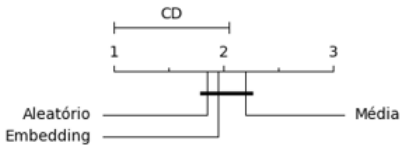
Fonte: Figura do autor



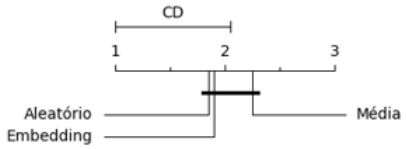
Modelo: Base Aleatório com positivos
Épocas:20
Métrica: Recall@10
Friedman pvalue = 0,173773943450442



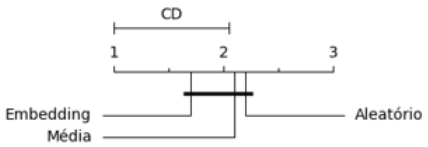
Modelo: Base Aleatório com positivos
Épocas:20
Métrica: NDCG@10
Friedman pvalue = 0,146156557071542



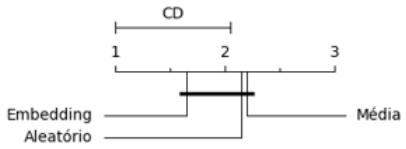
Modelo: Base Aleatório com positivos
Épocas:20
Métrica: Recall @20
Friedman pvalue = 0,594520547970192



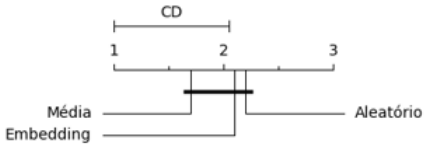
Modelo: Base Aleatório com positivos
Épocas:20
Métrica: NDCG@20
Friedman pvalue = 0,507341248128483



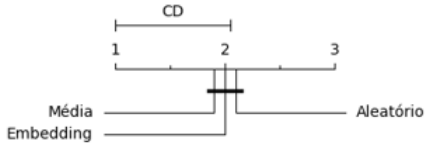
Modelo: Base Aleatório com positivos
Épocas:20
Métrica: Recall @30
Friedman pvalue = 0,34064204415744



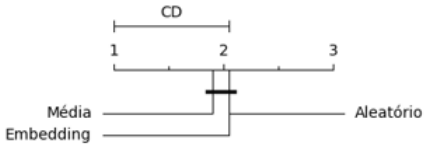
Modelo: Base Aleatório com positivos
Épocas:20
Métrica: NDCG@30
Friedman pvalue = 0,314663961018457



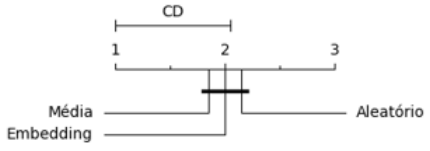
Modelo: Base Aleatório com positivos
Épocas:20
Métrica: Recall @40
Friedman pvalue = 0,34064204415744



Modelo: Base Aleatório com positivos
Épocas:20
Métrica: NDCG@40
Friedman pvalue = 0,894839316814368



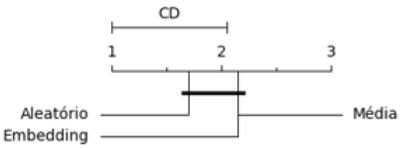
Modelo: Base Aleatório com positivos
Épocas:20
Métrica: Recall @50
Friedman pvalue = 0,891023376695552



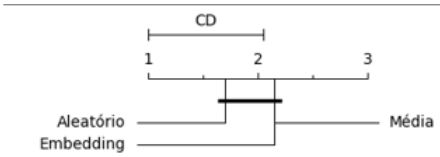
Modelo: Base Aleatório com positivos
Épocas:20
Métrica: NDCG@50
Friedman pvalue = 0,773257740483189

Figura 26 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação "Base Aleatório com Positivos", 20 Épocas

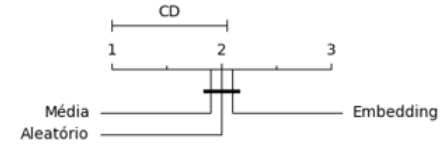
Fonte: Figura do autor



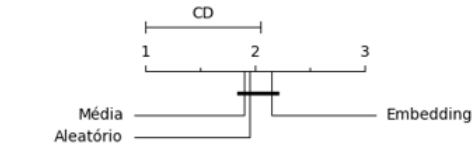
Modelo: Base Aleatório com positivos
Épocas:40
Métrica: Recall@10
Friedman pvalue = 0,223130160148428



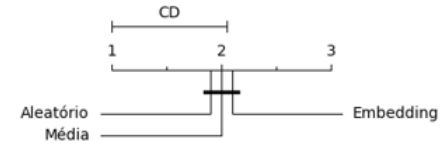
Modelo: Base Aleatório com positivos
Épocas:40
Métrica: NDCG@10
Friedman pvalue = 0,223130160148428



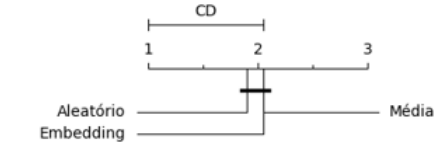
Modelo: Base Aleatório com positivos
Épocas:40
Métrica: Recall @20
Friedman pvalue = 0,833752918075178



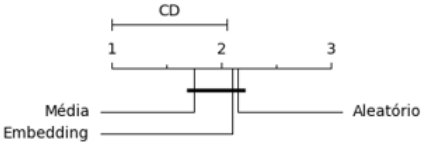
Modelo: Base Aleatório com positivos
Épocas:40
Métrica: NDCG@20
Friedman pvalue = 0,771622946715406



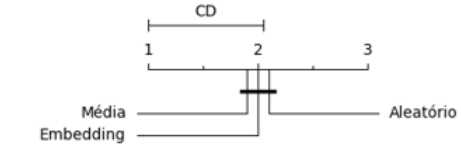
Modelo: Base Aleatório com positivos
Épocas:40
Métrica: Recall @30
Friedman pvalue = 0,852143788966209



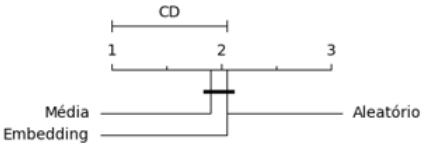
Modelo: Base Aleatório com positivos
Épocas:40
Métrica: NDCG@30
Friedman pvalue = 0,90776096127388



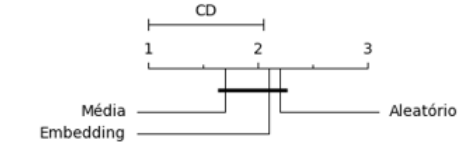
Modelo: Base Aleatório com positivos
Épocas:40
Métrica: Recall @40
Friedman pvalue = 0,562278738672987



Modelo: Base Aleatório com positivos
Épocas:40
Métrica: NDCG@40
Friedman pvalue = 0,900087626252258



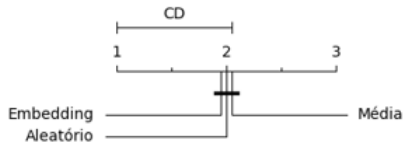
Modelo: Base Aleatório com positivos
Épocas:40
Métrica: Recall @50
Friedman pvalue = 0,901722662363008



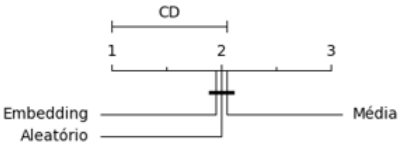
Modelo: Base Aleatório com positivos
Épocas:40
Métrica: NDCG@50
Friedman pvalue = 0,46918564525488

Figura 27 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação “Base Aleatório com Positivos”, 40 Épocas

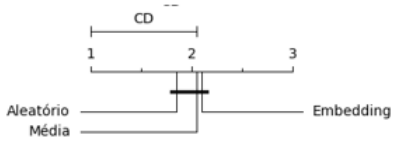
Fonte: Figura do autor



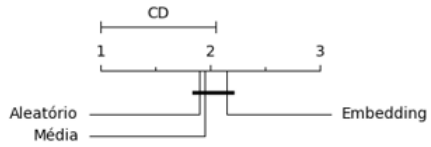
Modelo: Base Aleatório com positivos
Épocas:80
Métrica: Recall@10
Friedman pvalue = 0,948729480016426



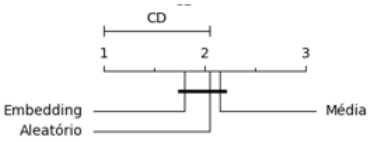
Modelo: Base Aleatório com positivos
Épocas:80
Métrica: NDCG@10
Friedman pvalue = 0,948729480016426



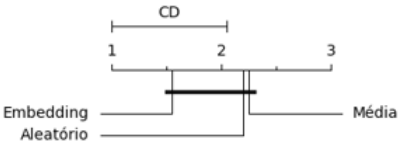
Modelo: Base Aleatório com positivos
Épocas:80
Métrica: Recall @20
Friedman pvalue = 0,797872513660219



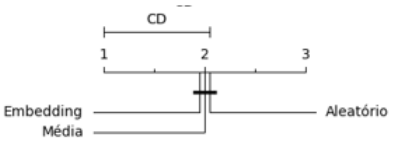
Modelo: Base Aleatório com positivos
Épocas:80
Métrica: NDCG@20
Friedman pvalue = 0,808866647182085



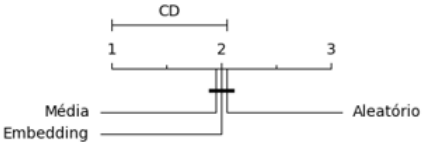
Modelo: Base Aleatório com positivos
Épocas:80
Métrica: Recall @30
Friedman pvalue = 0,628583933398623



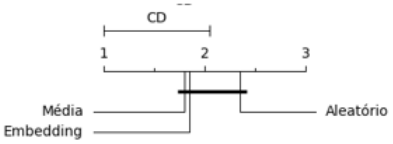
Modelo: Base Aleatório com positivos
Épocas:80
Métrica: NDCG@30
Friedman pvalue = 0,200836696193285



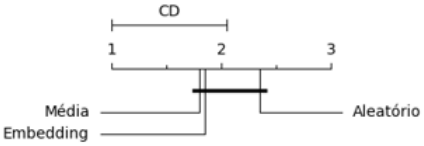
Modelo: Base Aleatório com positivos
Épocas:80
Métrica: Recall @40
Friedman pvalue = 0,963640444301278



Modelo: Base Aleatório com positivos
Épocas:80
Métrica: NDCG@40
Friedman pvalue = 0,964915944372362



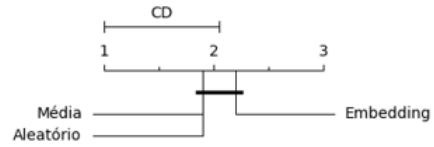
Modelo: Base Aleatório com positivos
Épocas:80
Métrica: Recall @50
Friedman pvalue = 0,325884543557382



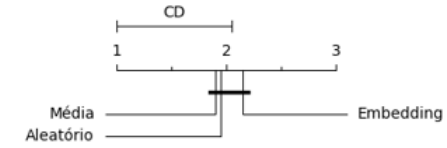
Modelo: Base Aleatório com positivos
Épocas:80
Métrica: NDCG@50
Friedman pvalue = 0,367879441171441

Figura 28 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação "Base Aleatório com Positivos", 80 Épocas

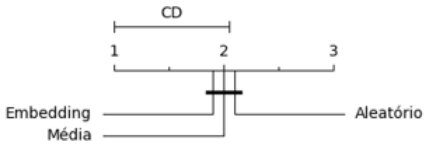
Fonte: Figura do autor



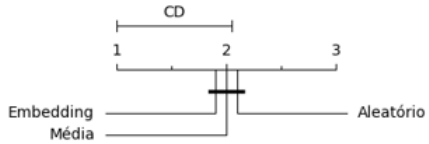
Modelo: Base Aleatório com positivos
Épocas:100
Métrica: Recall@10
Friedman pvalue = 0,513417119032587



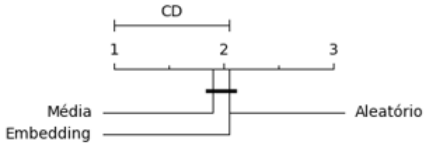
Modelo: Base Aleatório com positivos
Épocas:100
Métrica: NDCG@10
Friedman pvalue = 0,691825825270511



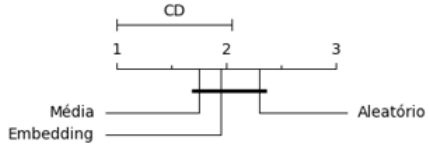
Modelo: Base Aleatório com positivos
Épocas:100
Métrica: Recall @20
Friedman pvalue = 0,833752918075178



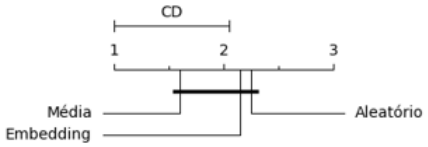
Modelo: Base Aleatório com positivos
Épocas:100
Métrica: NDCG@20
Friedman pvalue = 0,833752918075178



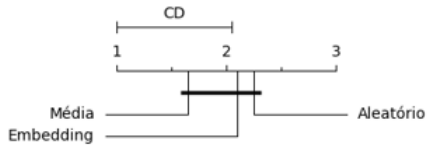
Modelo: Base Aleatório com positivos
Épocas:100
Métrica: Recall @30
Friedman pvalue = 0,904837418035956



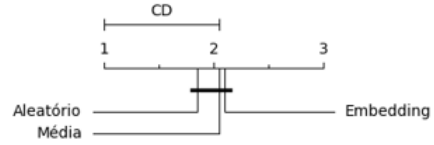
Modelo: Base Aleatório com positivos
Épocas:100
Métrica: NDCG@30
Friedman pvalue = 0,40181451356699



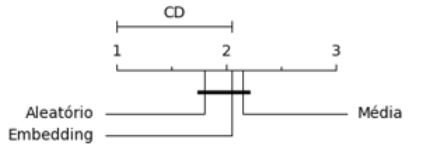
Modelo: Base Aleatório com positivos
Épocas:100
Métrica: Recall @40
Friedman pvalue = 0,20584225390236



Modelo: Base Aleatório com positivos
Épocas:100
Métrica: NDCG@40
Friedman pvalue = 0,328149587770579



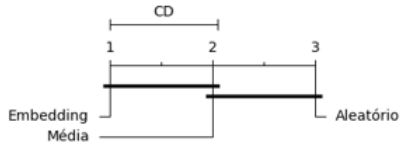
Modelo: Base Aleatório com positivos
Épocas:100
Métrica: Recall @50
Friedman pvalue = 0,803522573689057



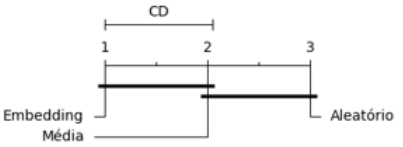
Modelo: Base Aleatório com positivos
Épocas:100
Métrica: NDCG@50
Friedman pvalue = 0,703736451659897

Figura 29 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação “Base Aleatório com Positivos”, 100 Épocas

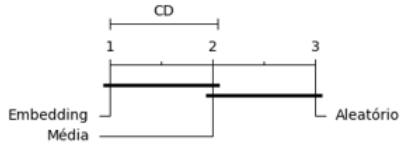
Fonte: Figura do autor



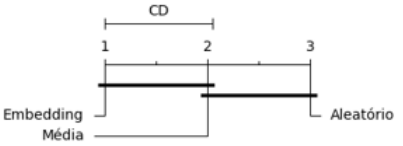
Modelo: Base Completa com positivos
Épocas:10
Métrica: Recall@10
Friedman pvalue = 0,0000453999297624849



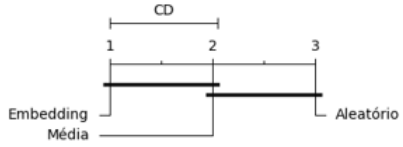
Modelo: Base Completa com positivos
Épocas:10
Métrica: NDCG@10
Friedman pvalue = 0,0000453999297624849



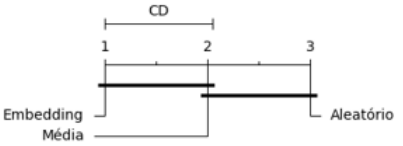
Modelo: Base Completa com positivos
Épocas:10
Métrica: Recall @20
Friedman pvalue = 0,0000453999297624849



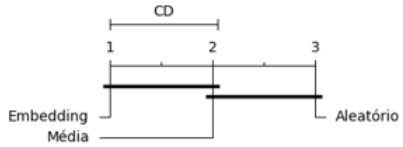
Modelo: Base Completa com positivos
Épocas:10
Métrica: NDCG@20
Friedman pvalue = 0,0000453999297624849



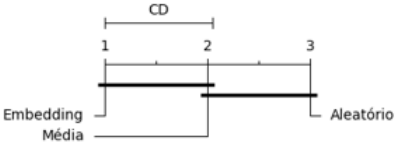
Modelo: Base Completa com positivos
Épocas:10
Métrica: Recall @30
Friedman pvalue = 0,0000453999297624849



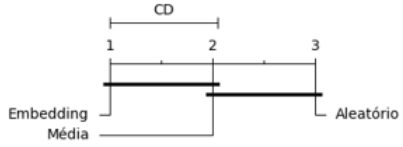
Modelo: Base Completa com positivos
Épocas:10
Métrica: NDCG@30
Friedman pvalue = 0,0000453999297624849



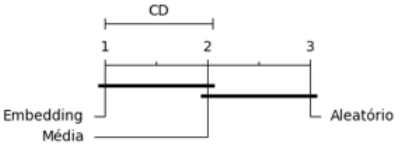
Modelo: Base Completa com positivos
Épocas:10
Métrica: Recall @40
Friedman pvalue = 0,0000453999297624849



Modelo: Base Completa com positivos
Épocas:10
Métrica: NDCG@40
Friedman pvalue = 0,0000453999297624849



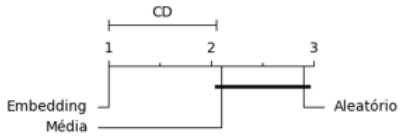
Modelo: Base Completa com positivos
Épocas:10
Métrica: Recall @50
Friedman pvalue = 0,0000453999297624849



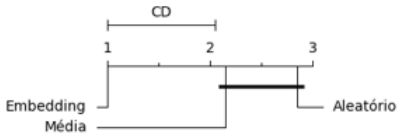
Modelo: Base Completa com positivos
Épocas:10
Métrica: NDCG@50
Friedman pvalue = 0,0000453999297624849

Figura 30 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação "Base Completa com Positivos", 10 Épocas

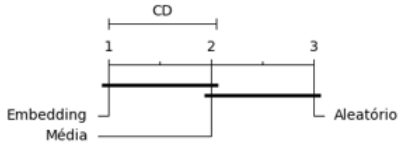
Fonte: Figura do autor



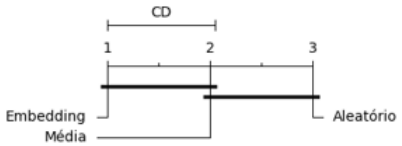
Modelo: Base Completa com positivos
Épocas:20
Métrica: Recall@10
Friedman pvalue = 0,0000691697196497659



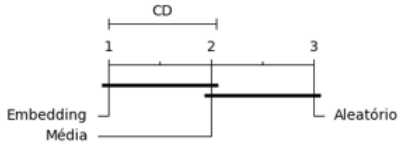
Modelo: Base Completa com positivos
Épocas:20
Métrica: NDCG@10
Friedman pvalue = 0,000129903596405296



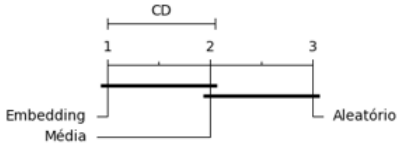
Modelo: Base Completa com positivos
Épocas:20
Métrica: Recall @20
Friedman pvalue = 0,0000453999297624849



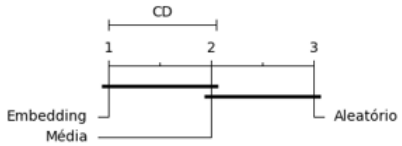
Modelo: Base Completa com positivos
Épocas:20
Métrica: NDCG@20
Friedman pvalue = 0,0000453999297624849



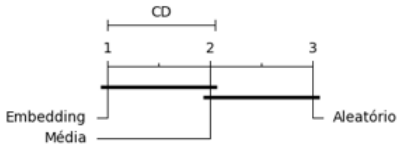
Modelo: Base Completa com positivos
Épocas:20
Métrica: Recall @30
Friedman pvalue = 0,0000453999297624849



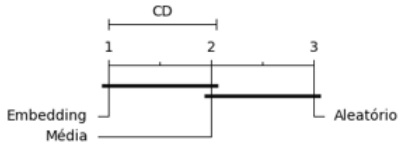
Modelo: Base Completa com positivos
Épocas:20
Métrica: NDCG@30
Friedman pvalue = 0,0000453999297624849



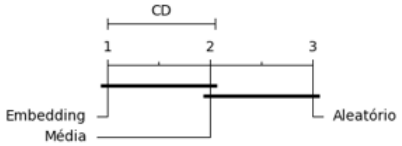
Modelo: Base Completa com positivos
Épocas:20
Métrica: Recall @40
Friedman pvalue = 0,0000453999297624849



Modelo: Base Completa com positivos
Épocas:20
Métrica: NDCG@40
Friedman pvalue = 0,0000453999297624849



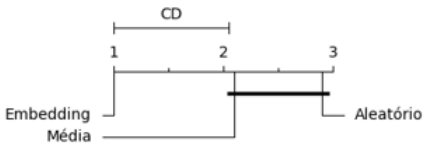
Modelo: Base Completa com positivos
Épocas:20
Métrica: Recall @50
Friedman pvalue = 0,0000453999297624849



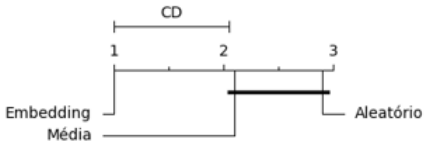
Modelo: Base Completa com positivos
Épocas:20
Métrica: NDCG@50
Friedman pvalue = 0,0000453999297624849

Figura 31 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação “Base Completa com Positivos”, 20 Épocas

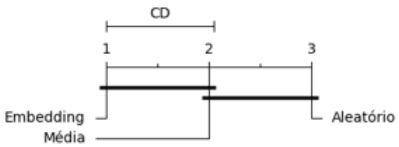
Fonte: Figura do autor



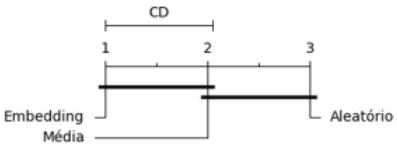
Modelo: Base Completa com positivos
Épocas:40
Métrica: Recall@10
Friedman pvalue = 0,0000691697196497659



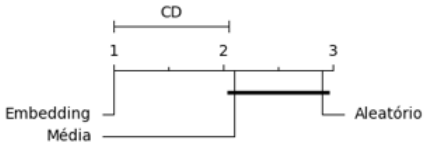
Modelo: Base Completa com positivos
Épocas:40
Métrica: NDCG@10
Friedman pvalue = 0,0000691697196497659



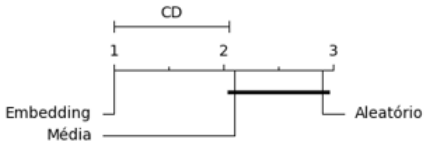
Modelo: Base Completa com positivos
Épocas:40
Métrica: Recall @20
Friedman pvalue = 0,0000453999297624849



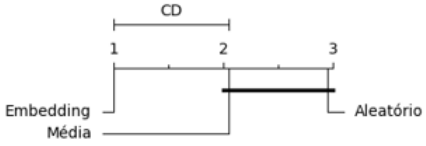
Modelo: Base Completa com positivos
Épocas:40
Métrica: NDCG@20
Friedman pvalue = 0,0000453999297624849



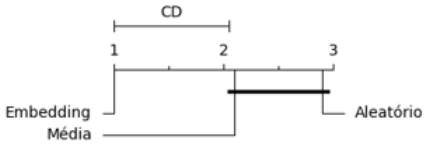
Modelo: Base Completa com positivos
Épocas:40
Métrica: Recall @30
Friedman pvalue = 0,000111665808490114



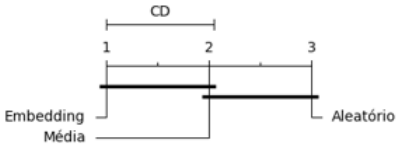
Modelo: Base Completa com positivos
Épocas:40
Métrica: NDCG@30
Friedman pvalue = 0,000111665808490114



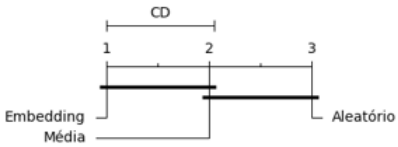
Modelo: Base Completa com positivos
Épocas:40
Métrica: Recall @40
Friedman pvalue = 0,0000571843230765873



Modelo: Base Completa com positivos
Épocas:40
Métrica: NDCG@40
Friedman pvalue = 0,000111665808490114



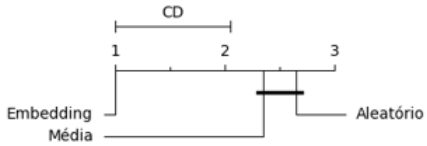
Modelo: Base Completa com positivos
Épocas:40
Métrica: Recall @50
Friedman pvalue = 0,0000453999297624849



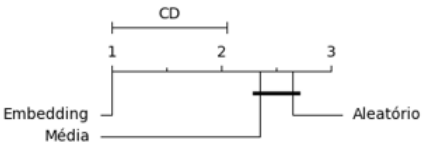
Modelo: Base Completa com positivos
Épocas:40
Métrica: NDCG@50
Friedman pvalue = 0,0000453999297624849

Figura 32 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação "Base Completa com Positivos", 40 Épocas

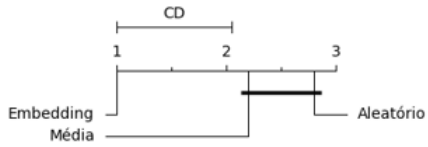
Fonte: Figura do autor



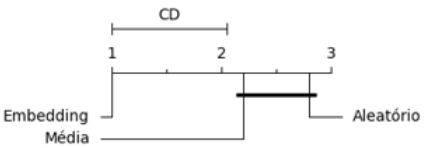
Modelo: Base Completa com positivos
Épocas:80
Métrica: Recall@10
Friedman pvalue = 0,000362285883975143



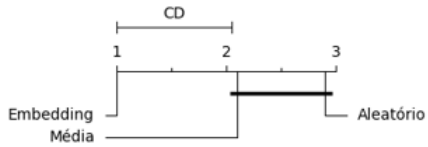
Modelo: Base Completa com positivos
Épocas:80
Métrica: NDCG@10
Friedman pvalue = 0,000362285883975143



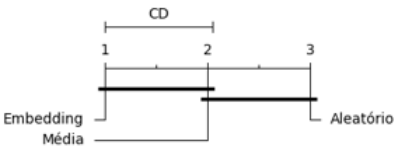
Modelo: Base Completa com positivos
Épocas:80
Métrica: Recall @20
Friedman pvalue = 0,000224867324178847



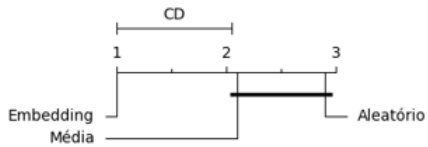
Modelo: Base Completa com positivos
Épocas:80
Métrica: NDCG@20
Friedman pvalue = 0,000224867324178847



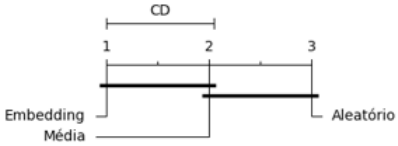
Modelo: Base Completa com positivos
Épocas:80
Métrica: Recall @30
Friedman pvalue = 0,000111665808490114



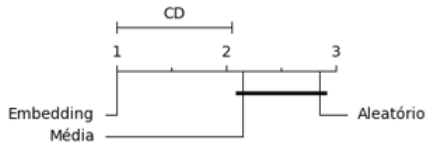
Modelo: Base Completa com positivos
Épocas:80
Métrica: NDCG@30
Friedman pvalue = 0,0000453999297624849



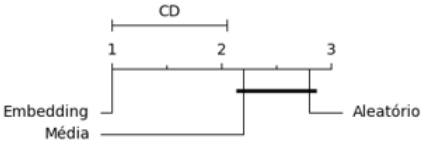
Modelo: Base Completa com positivos
Épocas:80
Métrica: Recall @40
Friedman pvalue = 0,000111665808490114



Modelo: Base Completa com positivos
Épocas:80
Métrica: NDCG@40
Friedman pvalue = 0,0000453999297624849



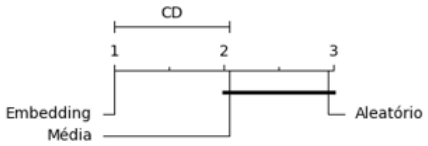
Modelo: Base Completa com positivos
Épocas:80
Métrica: Recall @50
Friedman pvalue = 0,000129903596405296



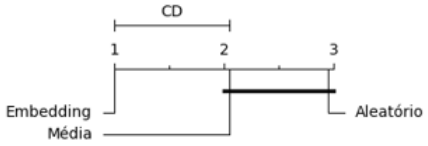
Modelo: Base Completa com positivos
Épocas:80
Métrica: NDCG@50
Friedman pvalue = 0,000224867324178847

Figura 33 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação “Base Completa com Positivos”, 80 Épocas

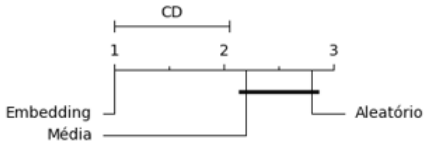
Fonte: Figura do autor



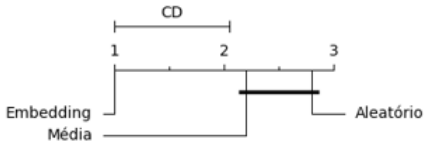
Modelo: Base Completa com positivos
Épocas:100
Métrica: Recall@10
Friedman pvalue = 0,0000571843230765873



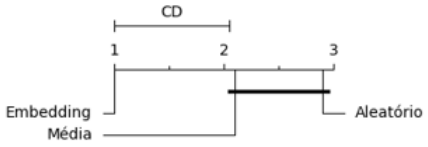
Modelo: Base Completa com positivos
Épocas:100
Métrica: NDCG@10
Friedman pvalue = 0,0000571843230765873



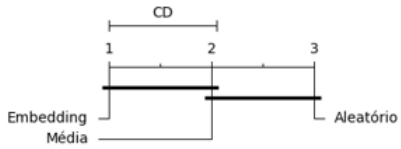
Modelo: Base Completa com positivos
Épocas:100
Métrica: Recall @20
Friedman pvalue = 0,000224867324178847



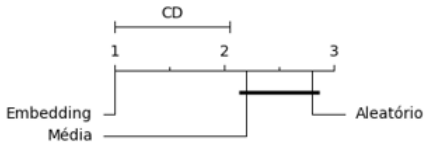
Modelo: Base Completa com positivos
Épocas:100
Métrica: NDCG@20
Friedman pvalue = 0,000224867324178847



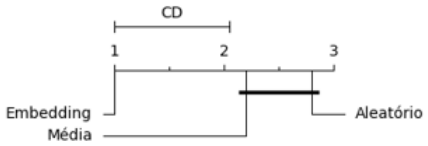
Modelo: Base Completa com positivos
Épocas:100
Métrica: Recall @30
Friedman pvalue = 0,0000691697196497659



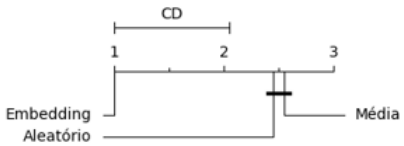
Modelo: Base Completa com positivos
Épocas:100
Métrica: NDCG@30
Friedman pvalue = 0,0000453999297624849



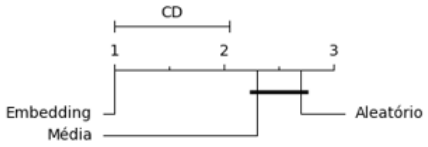
Modelo: Base Completa com positivos
Épocas:100
Métrica: Recall @40
Friedman pvalue = 0,000224867324178847



Modelo: Base Completa com positivos
Épocas:100
Métrica: NDCG@40
Friedman pvalue = 0,000224867324178847



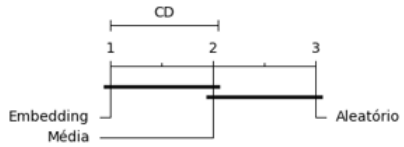
Modelo: Base Completa com positivos
Épocas:100
Métrica: Recall @50
Friedman pvalue = 0,000444772021660787



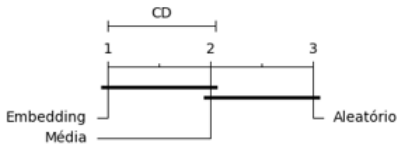
Modelo: Base Completa com positivos
Épocas:100
Métrica: NDCG@50
Friedman pvalue = 0,000370743540459086

Figura 34 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação "Base Completa com Positivos", 100 Épocas

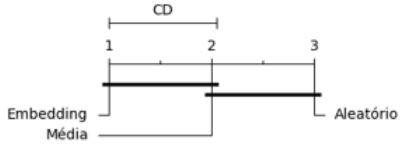
Fonte: Figura do autor



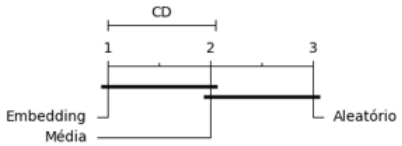
Modelo: Base Completa sem positivos
Épocas:10
Métrica: Recall@10
Friedman pvalue = 0,0000453999297624849



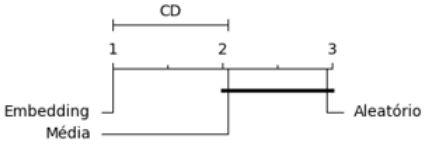
Modelo: Base Completa sem positivos
Épocas:10
Métrica: NDCG@10
Friedman pvalue = 0,0000453999297624849



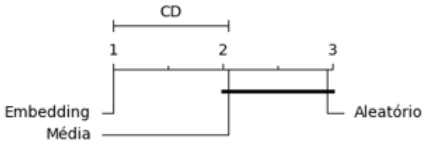
Modelo: Base Completa sem positivos
Épocas:10
Métrica: Recall @20
Friedman pvalue = 0,0000453999297624849



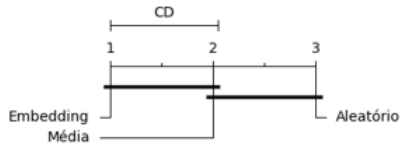
Modelo: Base Completa sem positivos
Épocas:10
Métrica: NDCG@20
Friedman pvalue = 0,0000453999297624849



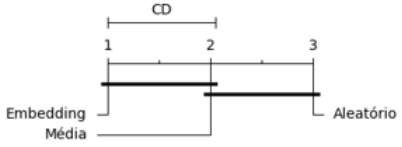
Modelo: Base Completa sem positivos
Épocas:10
Métrica: Recall @30
Friedman pvalue = 0,0000571843230765873



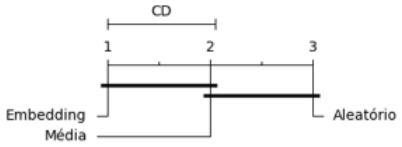
Modelo: Base Completa sem positivos
Épocas:10
Métrica: NDCG@30
Friedman pvalue = 0,0000571843230765873



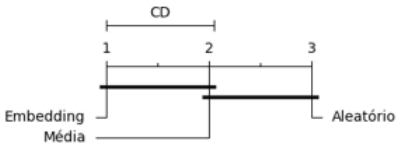
Modelo: Base Completa sem positivos
Épocas:10
Métrica: Recall @40
Friedman pvalue = 0,0000453999297624849



Modelo: Base Completa sem positivos
Épocas:10
Métrica: NDCG@40
Friedman pvalue = 0,0000453999297624849



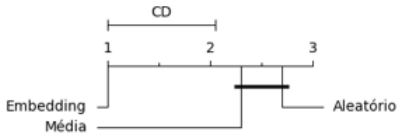
Modelo: Base Completa sem positivos
Épocas:10
Métrica: Recall @50
Friedman pvalue = 0,0000453999297624849



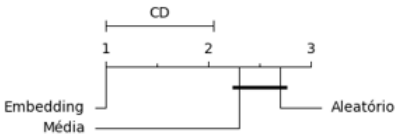
Modelo: Base Completa sem positivos
Épocas:10
Métrica: NDCG@50
Friedman pvalue = 0,0000453999297624849

Figura 35 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação “Base Completa sem Positivos”, 10 Épocas

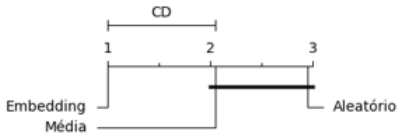
Fonte: Figura do autor



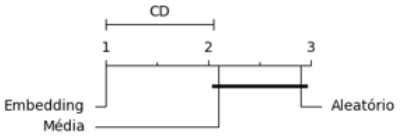
Modelo: Base Completa sem positivos
Épocas:20
Métrica: Recall@10
Friedman pvalue = 0,000244623693354663



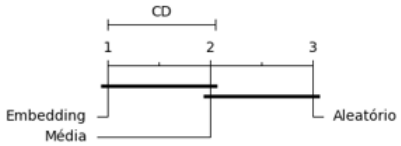
Modelo: Base Completa sem positivos
Épocas:20
Métrica: NDCG@10
Friedman pvalue = 0,000244623693354663



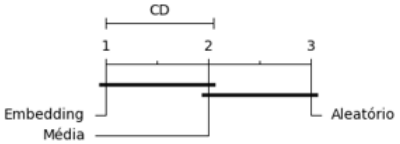
Modelo: Base Completa sem positivos
Épocas:20
Métrica: Recall @20
Friedman pvalue = 0,0000571843230765873



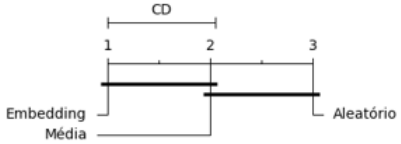
Modelo: Base Completa sem positivos
Épocas:20
Métrica: NDCG@20
Friedman pvalue = 0,000111665808490114



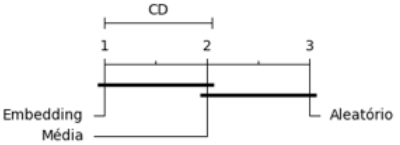
Modelo: Base Completa sem positivos
Épocas:20
Métrica: Recall @30
Friedman pvalue = 0,0000453999297624849



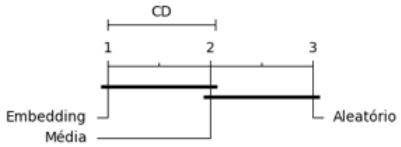
Modelo: Base Completa sem positivos
Épocas:20
Métrica: NDCG@30
Friedman pvalue = 0,0000453999297624849



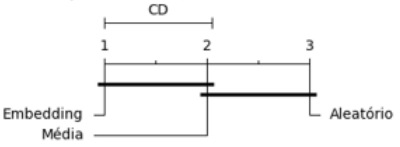
Modelo: Base Completa sem positivos
Épocas:20
Métrica: Recall @40
Friedman pvalue = 0,0000453999297624849



Modelo: Base Completa sem positivos
Épocas:20
Métrica: NDCG@40
Friedman pvalue = 0,0000453999297624849



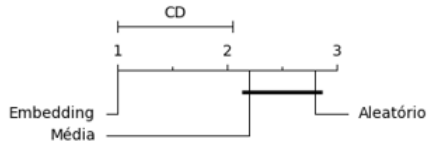
Modelo: Base Completa sem positivos
Épocas:20
Métrica: Recall @50
Friedman pvalue = 0,0000453999297624849



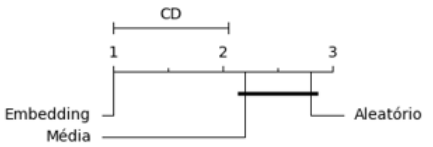
Modelo: Base Completa sem positivos
Épocas:20
Métrica: NDCG@50
Friedman pvalue = 0,0000453999297624849

Figura 36 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação "Base Completa sem Positivos", 20 Épocas

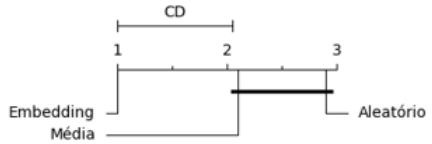
Fonte: Figura do autor



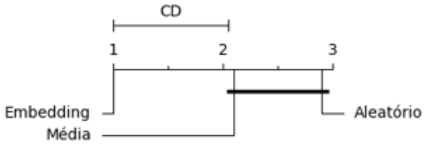
Modelo: Base Completa sem positivos
Épocas:40
Métrica: Recall@10
Friedman pvalue = 0,0000884269886598826



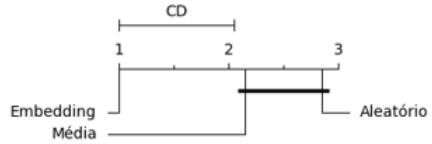
Modelo: Base Completa sem positivos
Épocas:40
Métrica: NDCG@10
Friedman pvalue = 0,0000884269886598826



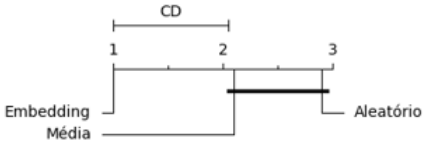
Modelo: Base Completa sem positivos
Épocas:40
Métrica: Recall @20
Friedman pvalue = 0,000111665808490114



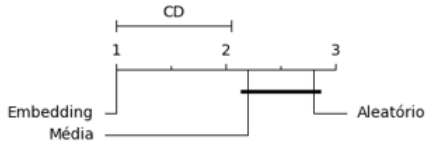
Modelo: Base Completa sem positivos
Épocas:40
Métrica: NDCG@20
Friedman pvalue = 0,000111665808490114



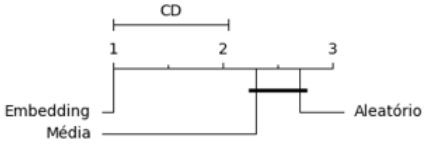
Modelo: Base Completa sem positivos
Épocas:40
Métrica: Recall @30
Friedman pvalue = 0,000129903596405296



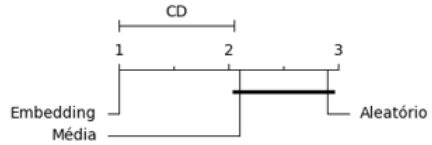
Modelo: Base Completa sem positivos
Épocas:40
Métrica: NDCG@30
Friedman pvalue = 0,000111665808490114



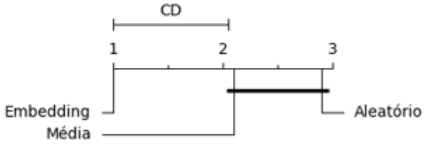
Modelo: Base Completa sem positivos
Épocas:40
Métrica: Recall @40
Friedman pvalue = 0,000144518177401405



Modelo: Base Completa sem positivos
Épocas:40
Métrica: NDCG@40
Friedman pvalue = 0,000370743540459086



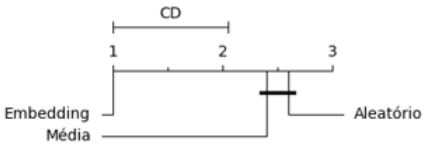
Modelo: Base Completa sem positivos
Épocas:40
Métrica: Recall @50
Friedman pvalue = 0,000111665808490114



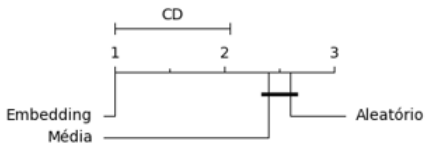
Modelo: Base Completa sem positivos
Épocas:40
Métrica: NDCG@50
Friedman pvalue = 0,000111665808490114

Figura 37 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação “Base Completa sem Positivos”, 40 Épocas

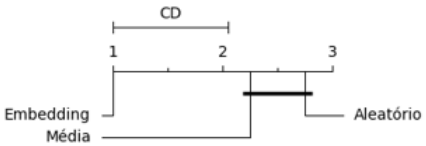
Fonte: Figura do autor



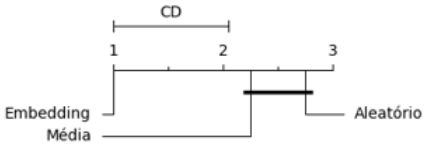
Modelo: Base Completa sem positivos
Épocas:80
Métrica: Recall@10
Friedman pvalue = 0,000335462627902509



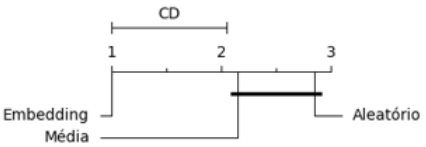
Modelo: Base Completa sem positivos
Épocas:80
Métrica: NDCG@10
Friedman pvalue = 0,000335462627902509



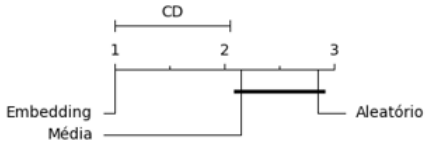
Modelo: Base Completa sem positivos
Épocas:80
Métrica: Recall @20
Friedman pvalue = 0,00015319732160851



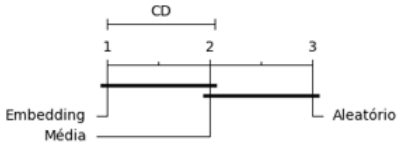
Modelo: Base Completa sem positivos
Épocas:80
Métrica: NDCG@20
Friedman pvalue = 0,00015319732160851



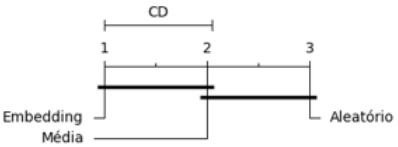
Modelo: Base Completa sem positivos
Épocas:80
Métrica: Recall @30
Friedman pvalue = 0,000129903596405296



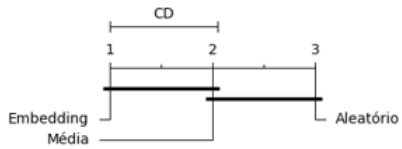
Modelo: Base Completa sem positivos
Épocas:80
Métrica: NDCG@30
Friedman pvalue = 0,000129903596405296



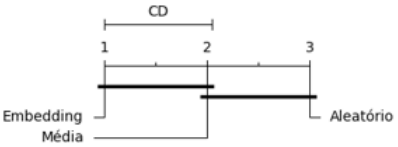
Modelo: Base Completa sem positivos
Épocas:80
Métrica: Recall @40
Friedman pvalue = 0,0000453999297624849



Modelo: Base Completa sem positivos
Épocas:80
Métrica: NDCG@40
Friedman pvalue = 0,0000453999297624849



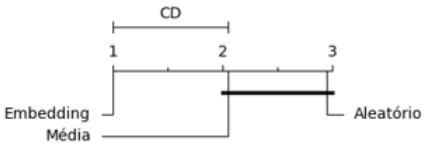
Modelo: Base Completa sem positivos
Épocas:80
Métrica: Recall @50
Friedman pvalue = 0,0000453999297624849



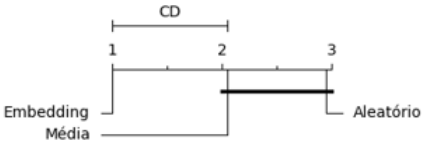
Modelo: Base Completa sem positivos
Épocas:80
Métrica: NDCG@50
Friedman pvalue = 0,0000453999297624849

Figura 38 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação "Base Completa sem Positivos", 80 Épocas

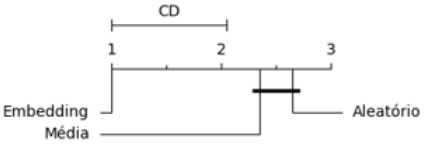
Fonte: Figura do autor



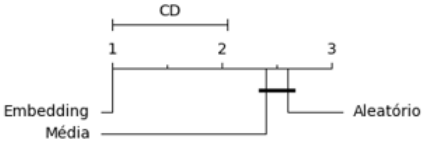
Modelo: Base Completa sem positivos
Épocas:100
Métrica: Recall@10
Friedman pvalue = 0,0000571843230765873



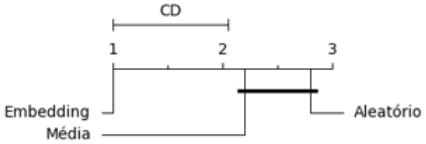
Modelo: Base Completa sem positivos
Épocas:100
Métrica: NDCG@10
Friedman pvalue = 0,0000571843230765873



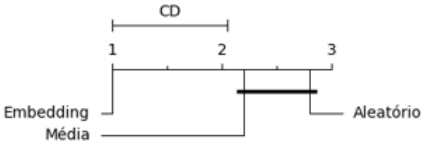
Modelo: Base Completa sem positivos
Épocas:100
Métrica: Recall @20
Friedman pvalue = 0,000236077279424617



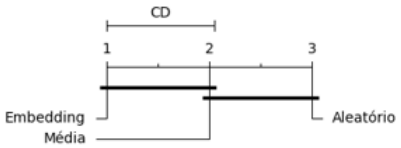
Modelo: Base Completa sem positivos
Épocas:100
Métrica: NDCG@20
Friedman pvalue = 0,000335462627902509



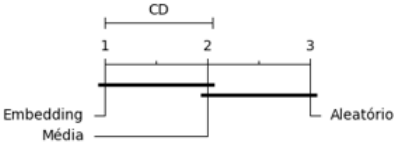
Modelo: Base Completa sem positivos
Épocas:100
Métrica: Recall @30
Friedman pvalue = 0,000144518177401405



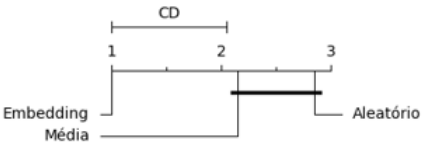
Modelo: Base Completa sem positivos
Épocas:100
Métrica: NDCG@30
Friedman pvalue = 0,000224867324178847



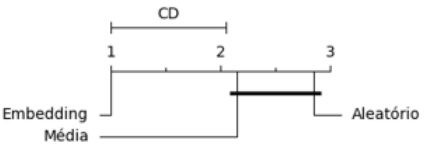
Modelo: Base Completa sem positivos
Épocas:100
Métrica: Recall @40
Friedman pvalue = 0,0000453999297624849



Modelo: Base Completa sem positivos
Épocas:100
Métrica: NDCG@40
Friedman pvalue = 0,0000453999297624849



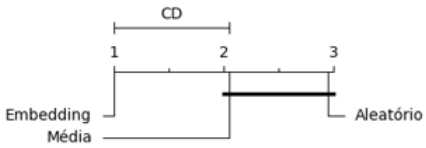
Modelo: Base Completa sem positivos
Épocas:100
Métrica: Recall @50
Friedman pvalue = 0,000129903596405296



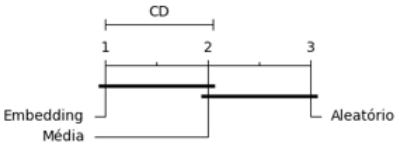
Modelo: Base Completa sem positivos
Épocas:100
Métrica: NDCG@50
Friedman pvalue = 0,000129903596405296

Figura 39 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação “Base Completa sem Positivos”, 100 Épocas

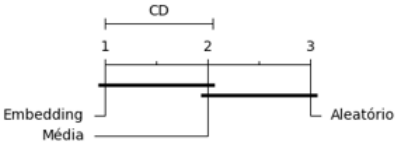
Fonte: Figura do autor



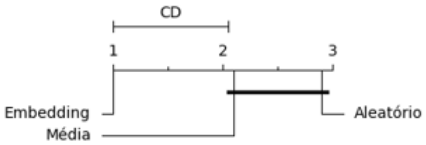
Modelo: Base Modelo com positivos
Épocas:10
Métrica: Recall@10
Friedman pvalue = 0,0000571843230765873



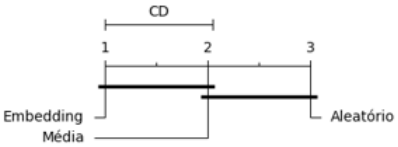
Modelo: Base Modelo com positivos
Épocas:10
Métrica: NDCG@10
Friedman pvalue = 0,0000453999297624849



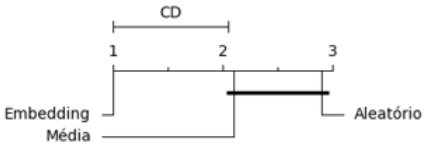
Modelo: Base Modelo com positivos
Épocas:10
Métrica: Recall @20
Friedman pvalue = 0,0000453999297624849



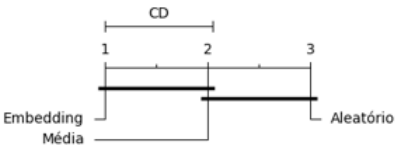
Modelo: Base Modelo com positivos
Épocas:10
Métrica: NDCG@20
Friedman pvalue = 0,000111665808490114



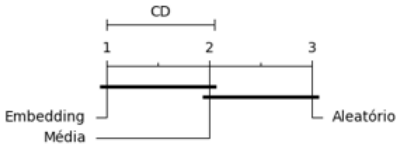
Modelo: Base Modelo com positivos
Épocas:10
Métrica: Recall @30
Friedman pvalue = 0,0000453999297624849



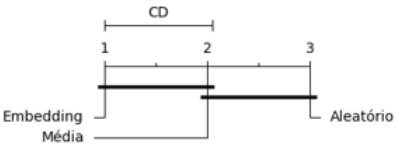
Modelo: Base Modelo com positivos
Épocas:10
Métrica: NDCG@30
Friedman pvalue = 0,000111665808490114



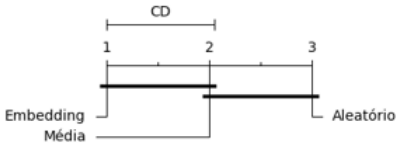
Modelo: Base Modelo com positivos
Épocas:10
Métrica: Recall @40
Friedman pvalue = 0,0000453999297624849



Modelo: Base Modelo com positivos
Épocas:10
Métrica: NDCG@40
Friedman pvalue = 0,0000453999297624849



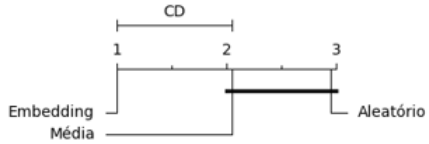
Modelo: Base Modelo com positivos
Épocas:10
Métrica: Recall @50
Friedman pvalue = 0,0000453999297624849



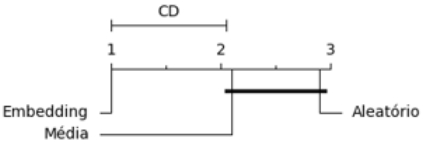
Modelo: Base Modelo com positivos
Épocas:10
Métrica: NDCG@50
Friedman pvalue = 0,0000453999297624849

Figura 40 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação "Base Modelo com Positivos", 10 Épocas

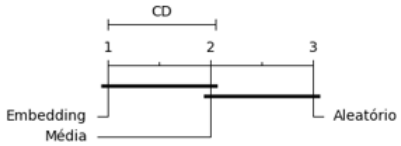
Fonte: Figura do autor



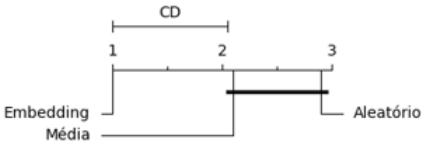
Modelo: Base Modelo com positivos
Épocas:20
Métrica: Recall@10
Friedman pvalue = 0,0000571843230765873



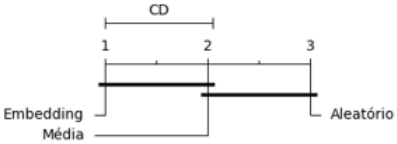
Modelo: Base Modelo com positivos
Épocas:20
Métrica: NDCG@10
Friedman pvalue = 0,000111665808490114



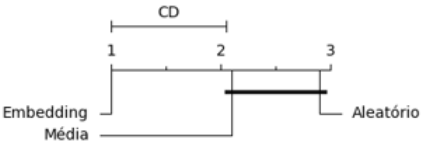
Modelo: Base Modelo com positivos
Épocas:20
Métrica: Recall @20
Friedman pvalue = 0,0000453999297624849



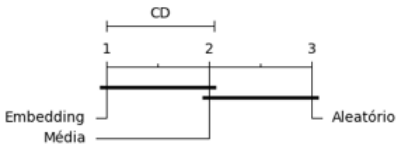
Modelo: Base Modelo com positivos
Épocas:20
Métrica: NDCG@20
Friedman pvalue = 0,000111665808490114



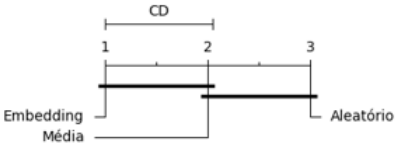
Modelo: Base Modelo com positivos
Épocas:20
Métrica: Recall @30
Friedman pvalue = 0,0000453999297624849



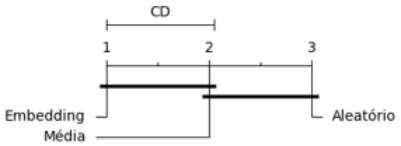
Modelo: Base Modelo com positivos
Épocas:20
Métrica: NDCG@30
Friedman pvalue = 0,000111665808490114



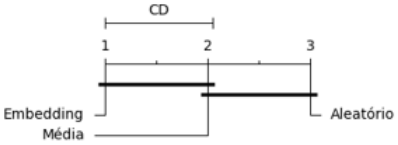
Modelo: Base Modelo com positivos
Épocas:20
Métrica: Recall @40
Friedman pvalue = 0,0000453999297624849



Modelo: Base Modelo com positivos
Épocas:20
Métrica: NDCG@40
Friedman pvalue = 0,0000453999297624849



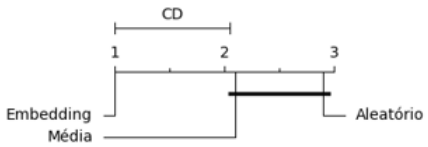
Modelo: Base Modelo com positivos
Épocas:20
Métrica: Recall @50
Friedman pvalue = 0,0000453999297624849



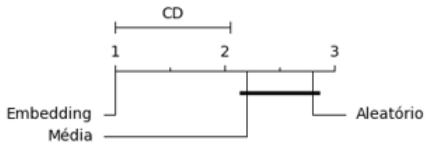
Modelo: Base Modelo com positivos
Épocas:20
Métrica: NDCG@50
Friedman pvalue = 0,0000453999297624849

Figura 41 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação “Base Modelo com Positivos”, 20 Épocas

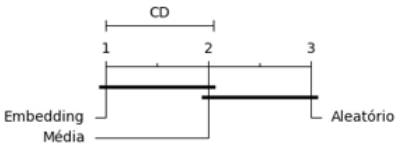
Fonte: Figura do autor



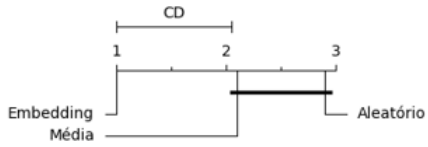
Modelo: Base Modelo com positivos
Épocas:40
Métrica: Recall@10
Friedman pvalue = 0,000111665808490114



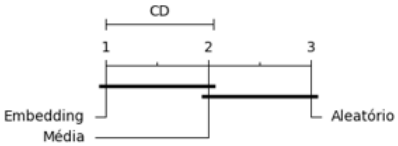
Modelo: Base Modelo com positivos
Épocas:40
Métrica: NDCG@10
Friedman pvalue = 0,000224867324178847



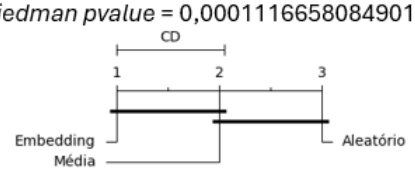
Modelo: Base Modelo com positivos
Épocas:40
Métrica: Recall @20
Friedman pvalue = 0,0000453999297624849



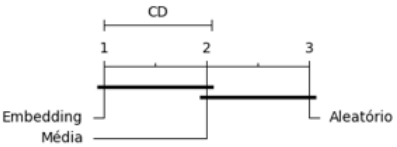
Modelo: Base Modelo com positivos
Épocas:40
Métrica: NDCG@20
Friedman pvalue = 0,000111665808490114



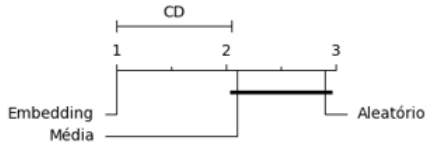
Modelo: Base Modelo com positivos
Épocas:40
Métrica: Recall @30
Friedman pvalue = 0,0000453999297624849



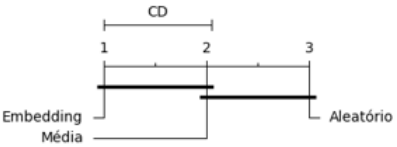
Modelo: Base Modelo com positivos
Épocas:40
Métrica: NDCG@30
Friedman pvalue = 0,0000453999297624849



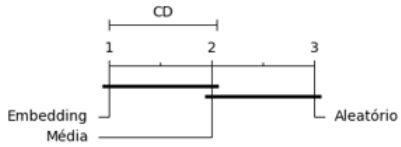
Modelo: Base Modelo com positivos
Épocas:40
Métrica: Recall @40
Friedman pvalue = 0,0000453999297624849



Modelo: Base Modelo com positivos
Épocas:40
Métrica: NDCG@40
Friedman pvalue = 0,000111665808490114



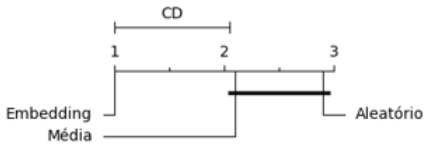
Modelo: Base Modelo com positivos
Épocas:40
Métrica: Recall @50
Friedman pvalue = 0,0000453999297624849



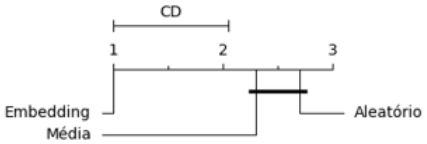
Modelo: Base Modelo com positivos
Épocas:40
Métrica: NDCG@50
Friedman pvalue = 0,0000453999297624849

Figura 42 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação "Base Modelo com Positivos", 40 Épocas

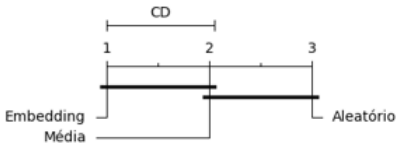
Fonte: Figura do autor



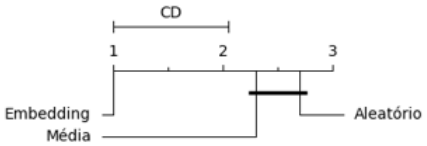
Modelo: Base Modelo com positivos
Épocas:80
Métrica: Recall@10
Friedman pvalue = 0,000111665808490114



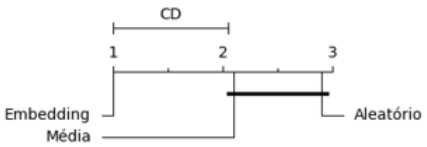
Modelo: Base Modelo com positivos
Épocas:80
Métrica: NDCG@10
Friedman pvalue = 0,000370743540459086



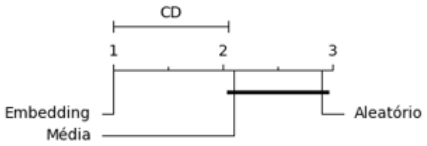
Modelo: Base Modelo com positivos
Épocas:80
Métrica: Recall @20
Friedman pvalue = 0,0000453999297624849



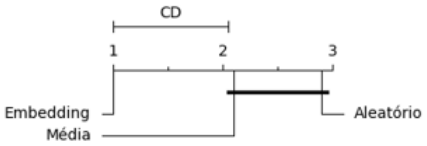
Modelo: Base Modelo com positivos
Épocas:80
Métrica: NDCG@20
Friedman pvalue = 0,000370743540459086



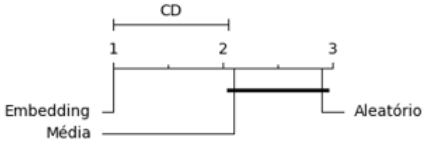
Modelo: Base Modelo com positivos
Épocas:80
Métrica: Recall @30
Friedman pvalue = 0,000111665808490114



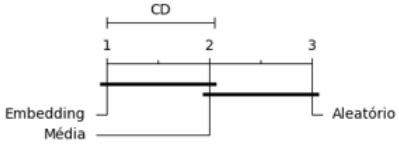
Modelo: Base Modelo com positivos
Épocas:80
Métrica: NDCG@30
Friedman pvalue = 0,000111665808490114



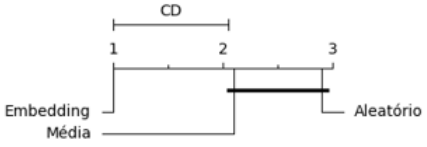
Modelo: Base Modelo com positivos
Épocas:80
Métrica: Recall @40
Friedman pvalue = 0,000111665808490114



Modelo: Base Modelo com positivos
Épocas:80
Métrica: NDCG@40
Friedman pvalue = 0,000111665808490114



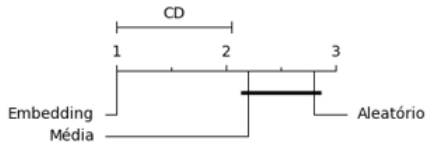
Modelo: Base Modelo com positivos
Épocas:80
Métrica: Recall @50
Friedman pvalue = 0,0000453999297624849



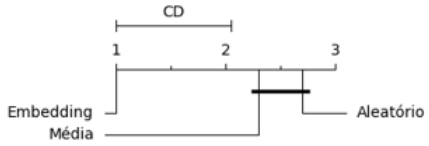
Modelo: Base Modelo com positivos
Épocas:80
Métrica: NDCG@50
Friedman pvalue = 0,000111665808490114

Figura 43 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação “Base Modelo com Positivos”, 80 Épocas

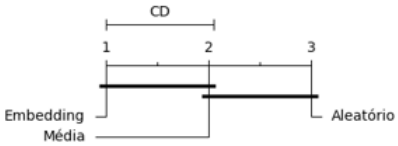
Fonte: Figura do autor



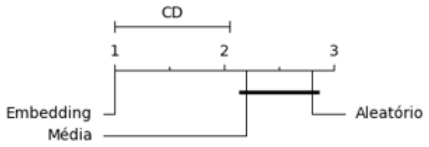
Modelo: Base Modelo com positivos
Épocas:100
Métrica: Recall@10
Friedman pvalue = 0,000224867324178847



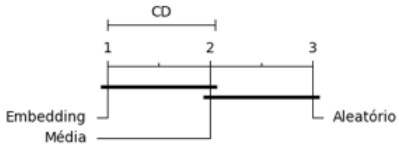
Modelo: Base Modelo com positivos
Épocas:100
Métrica: NDCG@10
Friedman pvalue = 0,000370743540459086



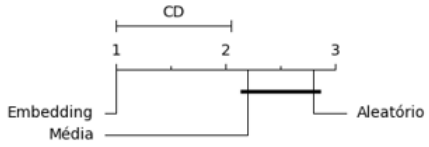
Modelo: Base Modelo com positivos
Épocas:100
Métrica: Recall @20
Friedman pvalue = 0,0000453999297624849



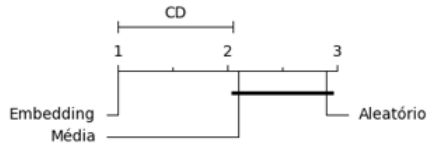
Modelo: Base Modelo com positivos
Épocas:100
Métrica: NDCG@20
Friedman pvalue = 0,000224867324178847



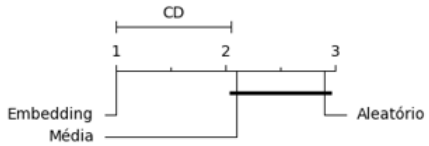
Modelo: Base Modelo com positivos
Épocas:100
Métrica: Recall @30
Friedman pvalue = 0,0000453999297624849



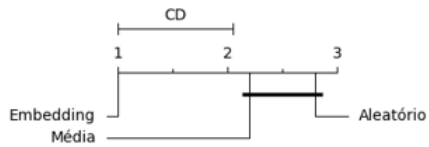
Modelo: Base Modelo com positivos
Épocas:100
Métrica: NDCG@30
Friedman pvalue = 0,000224867324178847



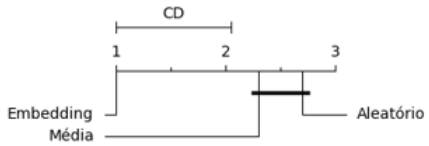
Modelo: Base Modelo com positivos
Épocas:100
Métrica: Recall @40
Friedman pvalue = 0,000111665808490114



Modelo: Base Modelo com positivos
Épocas:100
Métrica: NDCG@40
Friedman pvalue = 0,000111665808490114



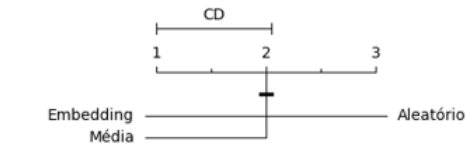
Modelo: Base Modelo com positivos
Épocas:100
Métrica: Recall @50
Friedman pvalue = 0,000224867324178847



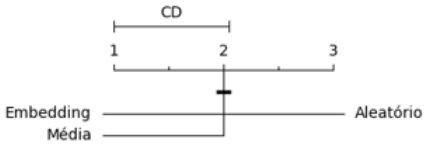
Modelo: Base Modelo com positivos
Épocas:100
Métrica: NDCG@50
Friedman pvalue = 0,000370743540459086

Figura 44 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação “Base Modelo com Positivos”, 100 Épocas

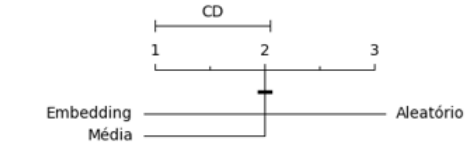
Fonte: Figura do autor



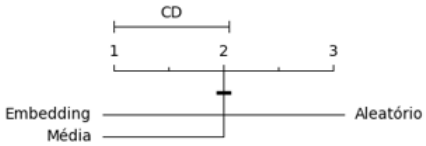
Modelo: Base TOPPOP com positivos
Épocas:10
Métrica: Recall@10
Friedman pvalue = NaN (Not a Number)



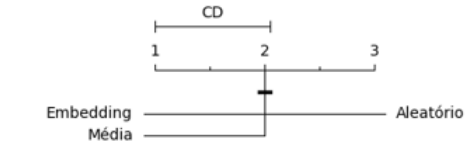
Modelo: Base TOPPOP com positivos
Épocas:10
Métrica: NDCG@10
Friedman pvalue = NaN (Not a Number)



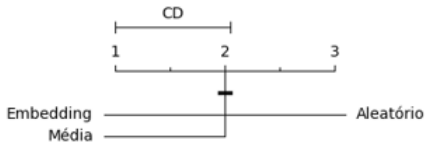
Modelo: Base TOPPOP com positivos
Épocas:10
Métrica: Recall @20
Friedman pvalue = NaN (Not a Number)



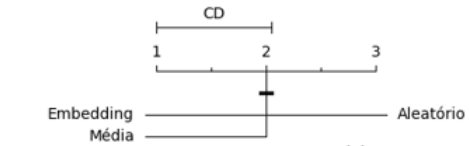
Modelo: Base TOPPOP com positivos
Épocas:10
Métrica: NDCG@20
Friedman pvalue = NaN (Not a Number)



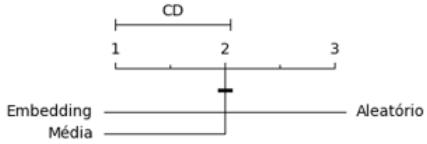
Modelo: Base TOPPOP com positivos
Épocas:10
Métrica: Recall @30
Friedman pvalue = NaN (Not a Number)



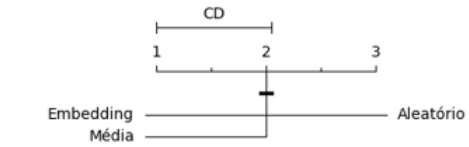
Modelo: Base TOPPOP com positivos
Épocas:10
Métrica: NDCG@30
Friedman pvalue = NaN (Not a Number)



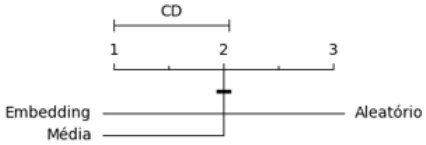
Modelo: Base TOPPOP com positivos
Épocas:10
Métrica: Recall @40
Friedman pvalue = NaN (Not a Number)



Modelo: Base TOPPOP com positivos
Épocas:10
Métrica: NDCG@40
Friedman pvalue = NaN (Not a Number)



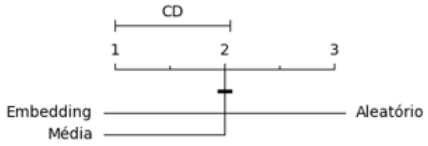
Modelo: Base TOPPOP com positivos
Épocas:10
Métrica: Recall @50
Friedman pvalue = NaN (Not a Number)



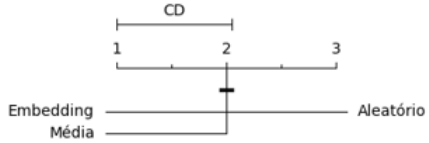
Modelo: Base TOPPOP com positivos
Épocas:10
Métrica: NDCG@50
Friedman pvalue = NaN (Not a Number)

Figura 45 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação “Base TOPPOP com Positivos”, 10 Épocas

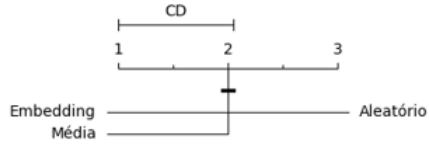
Fonte: Figura do autor



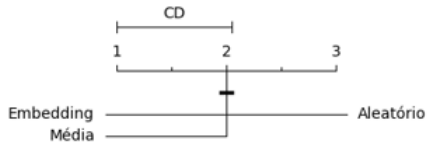
Modelo: Base TOPPOP com positivos
Épocas:20
Métrica: Recall@10
Friedman pvalue = NaN (Not a Number)



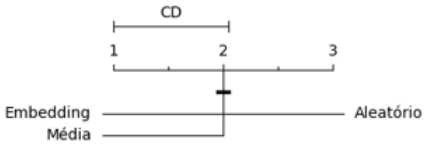
Modelo: Base TOPPOP com positivos
Épocas:20
Métrica: NDCG@10
Friedman pvalue = NaN (Not a Number)



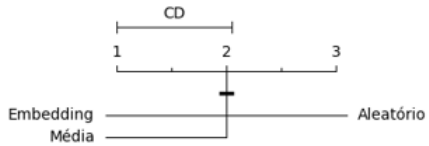
Modelo: Base TOPPOP com positivos
Épocas:20
Métrica: Recall @20
Friedman pvalue = NaN (Not a Number)



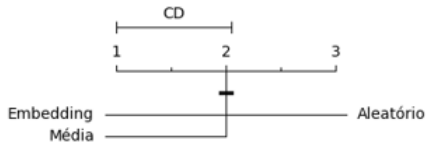
Modelo: Base TOPPOP com positivos
Épocas:20
Métrica: NDCG@20
Friedman pvalue = NaN (Not a Number)



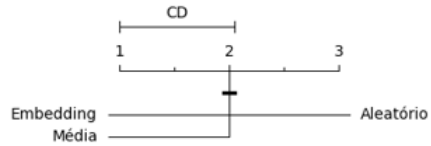
Modelo: Base TOPPOP com positivos
Épocas:20
Métrica: Recall @30
Friedman pvalue = NaN (Not a Number)



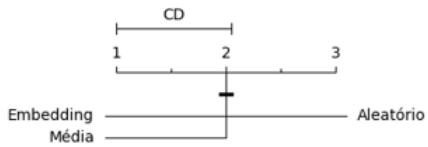
Modelo: Base TOPPOP com positivos
Épocas:20
Métrica: NDCG@30
Friedman pvalue = NaN (Not a Number)



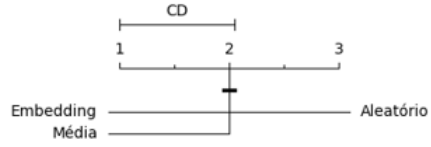
Modelo: Base TOPPOP com positivos
Épocas:20
Métrica: Recall @40
Friedman pvalue = NaN (Not a Number)



Modelo: Base TOPPOP com positivos
Épocas:20
Métrica: NDCG@40
Friedman pvalue = NaN (Not a Number)



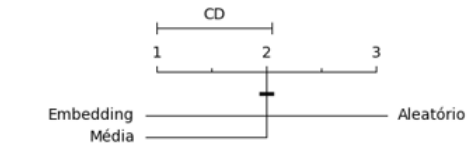
Modelo: Base TOPPOP com positivos
Épocas:20
Métrica: Recall @50
Friedman pvalue = NaN (Not a Number)



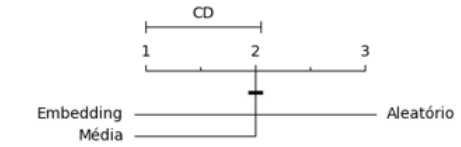
Modelo: Base TOPPOP com positivos
Épocas:20
Métrica: NDCG@50
Friedman pvalue = NaN (Not a Number)

Figura 46 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação "Base TOPPOP com Positivos", 20 Épocas

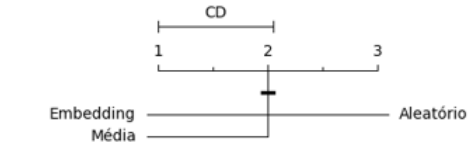
Fonte: Figura do autor



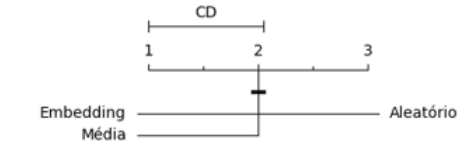
Modelo: Base TOPPOP com positivos
Épocas:40
Métrica: Recall@10
Friedman pvalue = NaN (Not a Number)



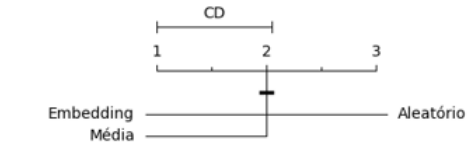
Modelo: Base TOPPOP com positivos
Épocas:40
Métrica: NDCG@10
Friedman pvalue = NaN (Not a Number)



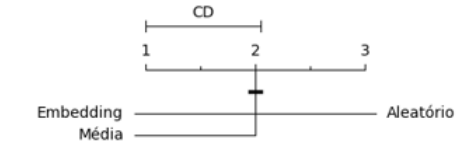
Modelo: Base TOPPOP com positivos
Épocas:40
Métrica: Recall @20
Friedman pvalue = NaN (Not a Number)



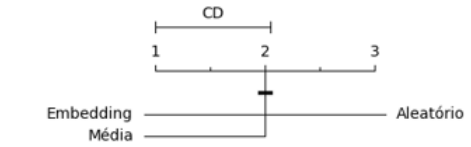
Modelo: Base TOPPOP com positivos
Épocas:40
Métrica: NDCG@20
Friedman pvalue = NaN (Not a Number)



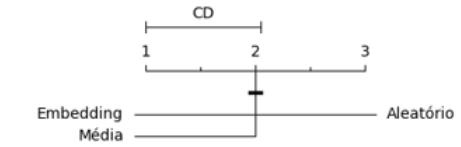
Modelo: Base TOPPOP com positivos
Épocas:40
Métrica: Recall @30
Friedman pvalue = NaN (Not a Number)



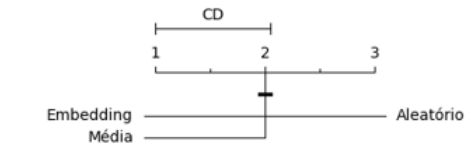
Modelo: Base TOPPOP com positivos
Épocas:40
Métrica: NDCG@30
Friedman pvalue = NaN (Not a Number)



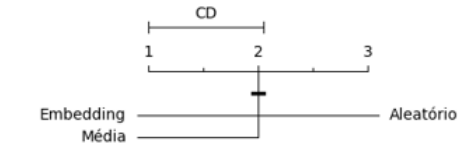
Modelo: Base TOPPOP com positivos
Épocas:40
Métrica: Recall @40
Friedman pvalue = NaN (Not a Number)



Modelo: Base TOPPOP com positivos
Épocas:40
Métrica: NDCG@40
Friedman pvalue = NaN (Not a Number)



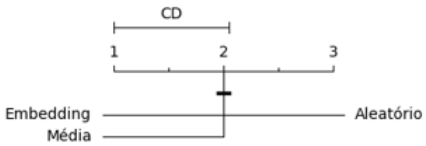
Modelo: Base TOPPOP com positivos
Épocas:40
Métrica: Recall @50
Friedman pvalue = NaN (Not a Number)



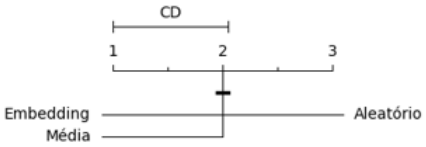
Modelo: Base TOPPOP com positivos
Épocas:40
Métrica: NDCG@50
Friedman pvalue = NaN (Not a Number)

Figura 47 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação “Base TOPPOP com Positivos”, 40 Épocas

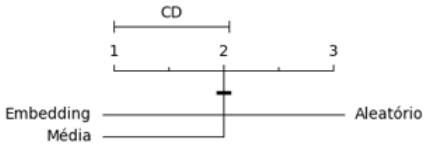
Fonte: Figura do autor



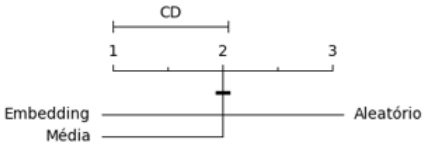
Modelo: Base TOPPOP com positivos
Épocas:80
Métrica: Recall@10
Friedman pvalue = NaN (Not a Number)



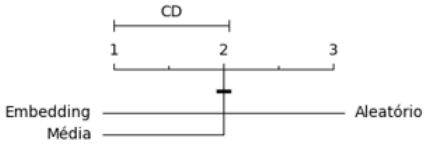
Modelo: Base TOPPOP com positivos
Épocas:80
Métrica: NDCG@10
Friedman pvalue = NaN (Not a Number)



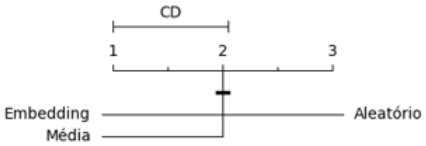
Modelo: Base TOPPOP com positivos
Épocas:80
Métrica: Recall @20
Friedman pvalue = NaN (Not a Number)



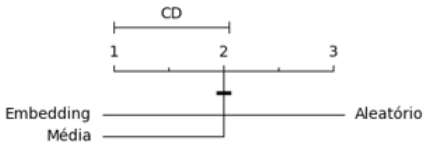
Modelo: Base TOPPOP com positivos
Épocas:80
Métrica: NDCG@20
Friedman pvalue = NaN (Not a Number)



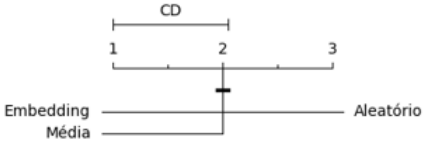
Modelo: Base TOPPOP com positivos
Épocas:80
Métrica: Recall @30
Friedman pvalue = NaN (Not a Number)



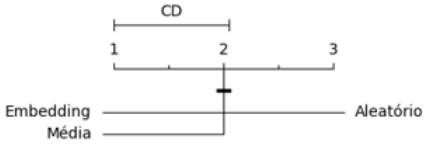
Modelo: Base TOPPOP com positivos
Épocas:80
Métrica: NDCG@30
Friedman pvalue = NaN (Not a Number)



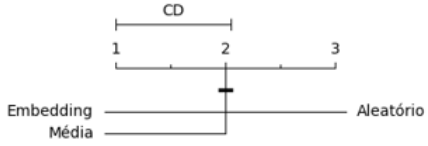
Modelo: Base TOPPOP com positivos
Épocas:80
Métrica: Recall @80
Friedman pvalue = NaN (Not a Number)



Modelo: Base TOPPOP com positivos
Épocas:80
Métrica: NDCG@40
Friedman pvalue = NaN (Not a Number)



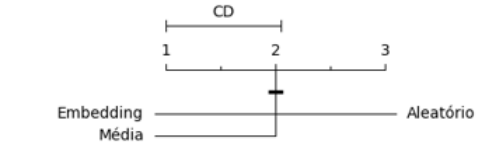
Modelo: Base TOPPOP com positivos
Épocas:80
Métrica: Recall @50
Friedman pvalue = NaN (Not a Number)



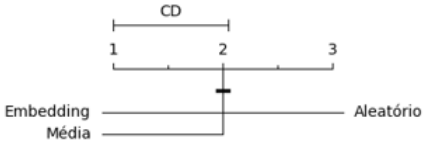
Modelo: Base TOPPOP com positivos
Épocas:80
Métrica: NDCG@50
Friedman pvalue = NaN (Not a Number)

Figura 48 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação "Base TOPPOP com Positivos", 80 Épocas

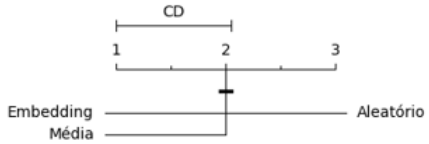
Fonte: Figura do autor



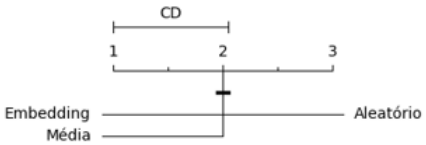
Modelo: Base TOPPOP com positivos
Épocas: 100
Métrica: Recall@10
Friedman pvalue = NaN (Not a Number)



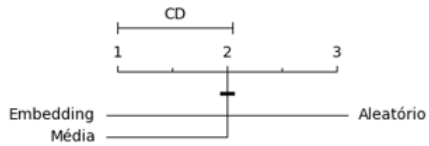
Modelo: Base TOPPOP com positivos
Épocas: 100
Métrica: NDCG@10
Friedman pvalue = NaN (Not a Number)



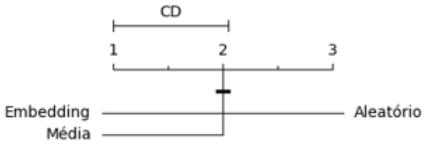
Modelo: Base TOPPOP com positivos
Épocas: 100
Métrica: Recall @20
Friedman pvalue = NaN (Not a Number)



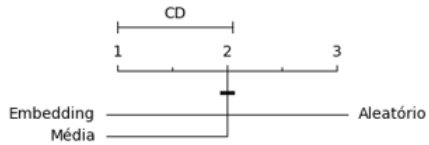
Modelo: Base TOPPOP com positivos
Épocas: 100
Métrica: NDCG@20
Friedman pvalue = NaN (Not a Number)



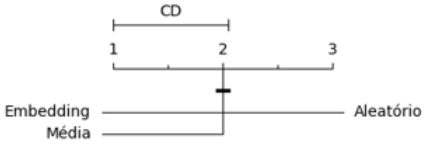
Modelo: Base TOPPOP com positivos
Épocas: 100
Métrica: Recall @30
Friedman pvalue = NaN (Not a Number)



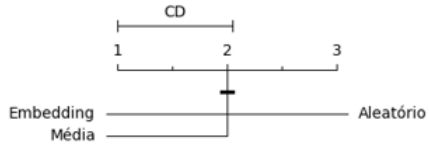
Modelo: Base TOPPOP com positivos
Épocas: 100
Métrica: NDCG@30
Friedman pvalue = NaN (Not a Number)



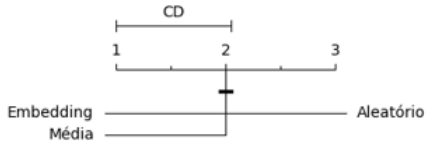
Modelo: Base TOPPOP com positivos
Épocas:100
Métrica: Recall @40
Friedman pvalue = NaN (Not a Number)



Modelo: Base TOPPOP com positivos
Épocas: 100
Métrica: NDCG@40
Friedman pvalue = NaN (Not a Number)



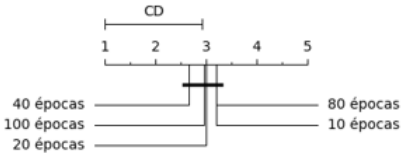
Modelo: Base TOPPOP com positivos
Épocas: 100
Métrica: Recall @50
Friedman pvalue = NaN (Not a Number)



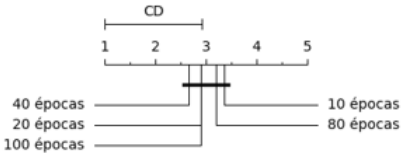
Modelo: Base TOPPOP com positivos
Épocas: 100
Métrica: NDCG@50
Friedman pvalue = NaN (Not a Number)

Figura 49 – Comparação entre os métodos de Inicialização utilizando: Modelo de Recomendação "Base TOPPOP com Positivos", 100 Épocas

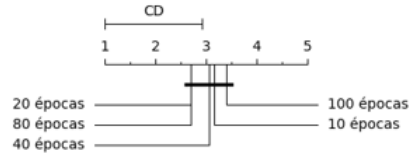
Fonte: Figura do autor



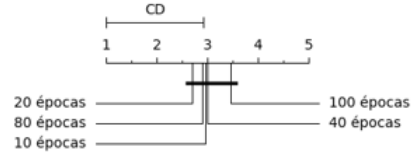
Modelo: Base Aleatório com positivos
Inicialização: Aleatório
Métrica: Recall@10
Friedman pvalue = 0,830623420458789



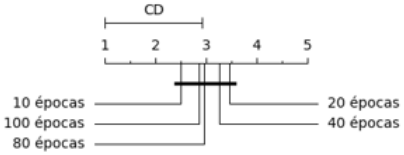
Modelo: Base Aleatório com positivos
Inicialização: Aleatório
Métrica: NDCG@10
Friedman pvalue = 0,716738903973587



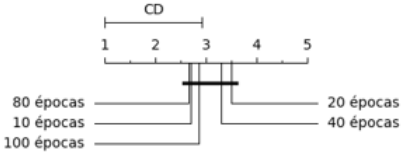
Modelo: Base Aleatório com positivos
Inicialização: Aleatório
Métrica: Recall@20
Friedman pvalue = 0,687410078825394



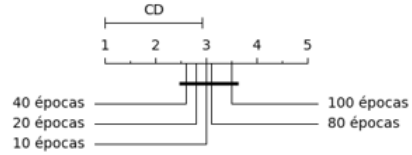
Modelo: Base Aleatório com positivos
Inicialização: Aleatório
Métrica: NDCG@20
Friedman pvalue = 0,780632049518724



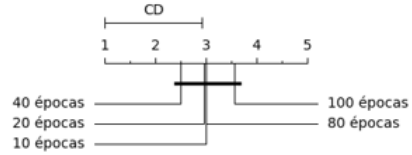
Modelo: Base Aleatório com positivos
Inicialização: Aleatório
Métrica: Recall@30
Friedman pvalue = 0,59715561519262



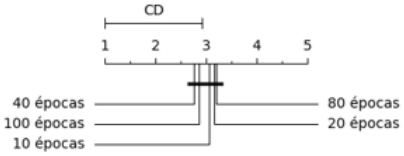
Modelo: Base Aleatório com positivos
Inicialização: Aleatório
Métrica: NDCG@30
Friedman pvalue = 0,621770253957956



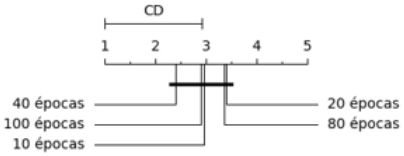
Modelo: Base Aleatório com positivos
Inicialização: Aleatório
Métrica: Recall@40
Friedman pvalue = 0,647057220874733



Modelo: Base Aleatório com positivos
Inicialização: Aleatório
Métrica: NDCG@40
Friedman pvalue = 0,613649202019256



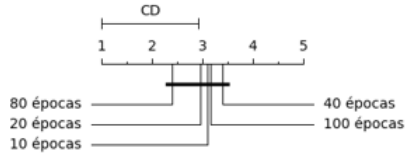
Modelo: Base Aleatório com positivos
Inicialização: Aleatório
Métrica: Recall@50
Friedman pvalue = 0,951112393418281



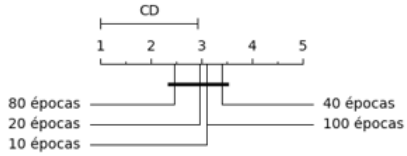
Modelo: Base Aleatório com positivos
Inicialização: Aleatório
Métrica: NDCG@50
Friedman pvalue = 0,588866608665283

Figura 50 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização Aleatório, Modelo de Recomendação "Base Aleatório com Positivos"

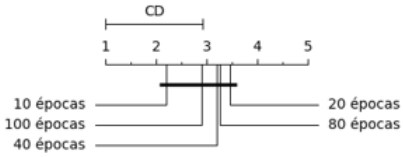
Fonte: Figura do autor



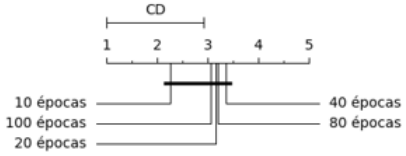
Modelo: Base Completa com positivos
Inicialização: Aleatório
Métrica: Recall@10
Friedman pvalue = 0,180998265360744



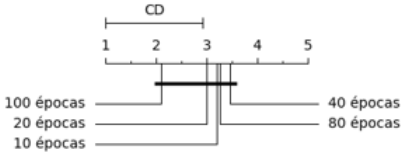
Modelo: Base Completa com positivos
Inicialização: Aleatório
Métrica: NDCG@10
Friedman pvalue = 0,249670244279371



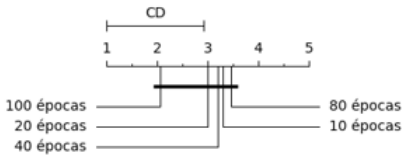
Modelo: Base Completa com positivos
Inicialização: Aleatório
Métrica: Recall@20
Friedman pvalue = 0,205012746395943



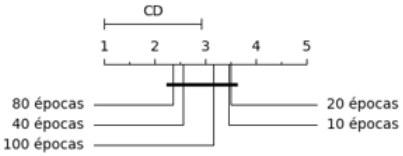
Modelo: Base Completa com positivos
Inicialização: Aleatório
Métrica: NDCG@20
Friedman pvalue = 0,357068102307124



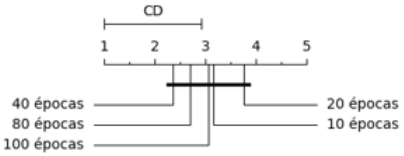
Modelo: Base Completa com positivos
Inicialização: Aleatório
Métrica: Recall@30
Friedman pvalue = 0,179075964564378



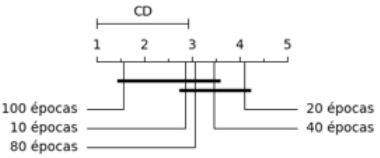
Modelo: Base Completa com positivos
Inicialização: Aleatório
Métrica: NDCG@30
Friedman pvalue = 0,183737841472147



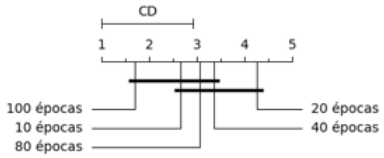
Modelo: Base Completa com positivos
Inicialização: Aleatório
Métrica: Recall@40
Friedman pvalue = 0,287297495183645



Modelo: Base Completa com positivos
Inicialização: Aleatório
Métrica: NDCG@40
Friedman pvalue = 0,33000994554144



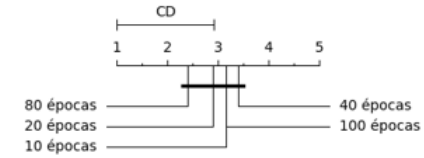
Modelo: Base Completa com positivos
Inicialização: Aleatório
Métrica: Recall@50
Friedman pvalue = 0,00425507976100106



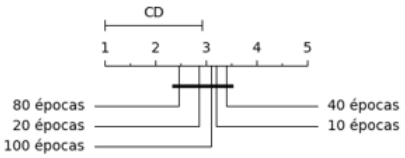
Modelo: Base Completa com positivos
Inicialização: Aleatório
Métrica: NDCG@50
Friedman pvalue = 0,00603396375490433

Figura 51 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização Aleatório, Modelo de Recomendação "Base Completa com Positivos"

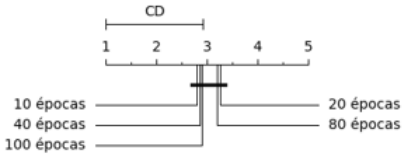
Fonte: Figura do autor



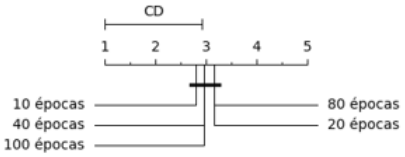
Modelo: Base Completa sem positivos
Inicialização: Aleatório
Métrica: Recall@10
Friedman pvalue = 0,160345077288154



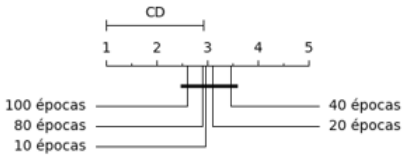
Modelo: Base Completa sem positivos
Inicialização: Aleatório
Métrica: NDCG@10
Friedman pvalue = 0,20333578962794



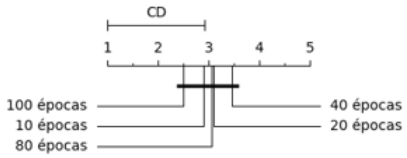
Modelo: Base Completa sem positivos
Inicialização: Aleatório
Métrica: Recall@20
Friedman pvalue = 0,841733483898014



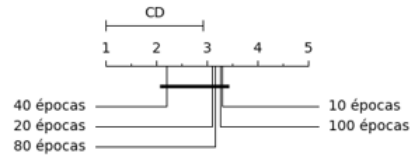
Modelo: Base Completa sem positivos
Inicialização: Aleatório
Métrica: NDCG@20
Friedman pvalue = 0,955375080765049



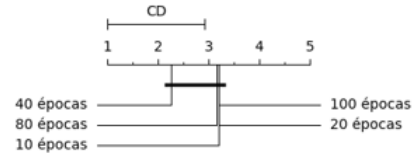
Modelo: Base Completa sem positivos
Inicialização: Aleatório
Métrica: Recall@30
Friedman pvalue = 0,613024005244684



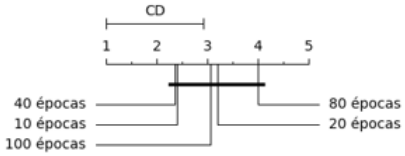
Modelo: Base Completa sem positivos
Inicialização: Aleatório
Métrica: NDCG@30
Friedman pvalue = 0,547087435281247



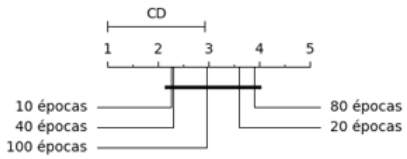
Modelo: Base Completa sem positivos
Inicialização: Aleatório
Métrica: Recall@40
Friedman pvalue = 0,336458317818489



Modelo: Base Completa sem positivos
Inicialização: Aleatório
Métrica: NDCG@40
Friedman pvalue = 0,453627369387309



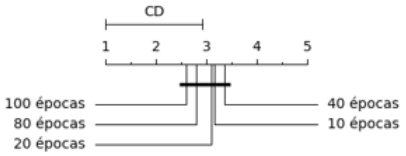
Modelo: Base Completa sem positivos
Inicialização: Aleatório
Métrica: Recall@50
Friedman pvalue = 0,0553611909769726



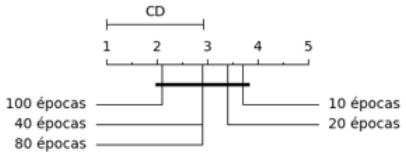
Modelo: Base Completa sem positivos
Inicialização: Aleatório
Métrica: NDCG@50
Friedman pvalue = 0,037636905470702

Figura 52 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização Aleatório, Modelo de Recomendação "Base Completa sem Positivos"

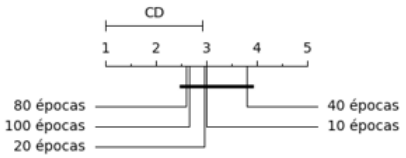
Fonte: Figura do autor



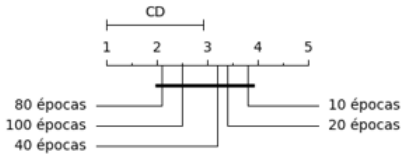
Modelo: Base Modelo com positivos
Inicialização: Aleatório
Métrica: Recall@10
Friedman pvalue = 0,811199897828555



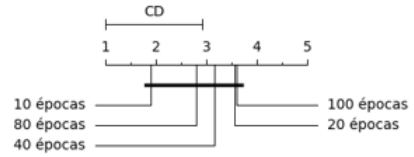
Modelo: Base Modelo com positivos
Inicialização: Aleatório
Métrica: NDCG@10
Friedman pvalue = 0,205202912003993



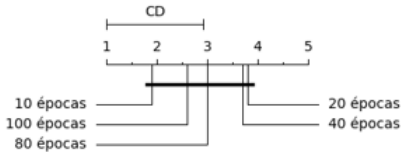
Modelo: Base Modelo com positivos
Inicialização: Aleatório
Métrica: Recall@20
Friedman pvalue = 0,434478105846136



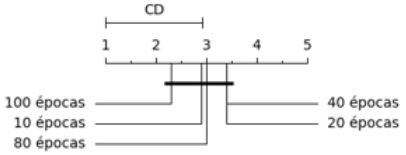
Modelo: Base Modelo com positivos
Inicialização: Aleatório
Métrica: NDCG@20
Friedman pvalue = 0,107379704909595



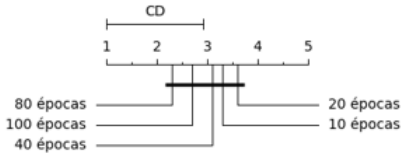
Modelo: Base Modelo com positivos
Inicialização: Aleatório
Métrica: Recall@30
Friedman pvalue = 0,0969256324208419



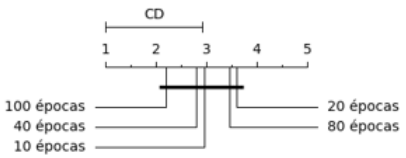
Modelo: Base Modelo com positivos
Inicialização: Aleatório
Métrica: NDCG@30
Friedman pvalue = 0,0404276819945127



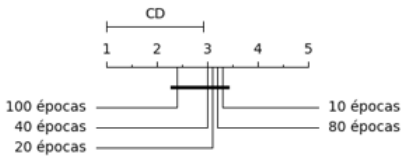
Modelo: Base Modelo com positivos
Inicialização: Aleatório
Métrica: Recall@40
Friedman pvalue = 0,493406445055115



Modelo: Base Modelo com positivos
Inicialização: Aleatório
Métrica: NDCG@40
Friedman pvalue = 0,384785053571634



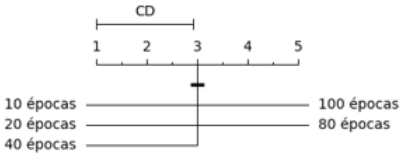
Modelo: Base Modelo com positivos
Inicialização: Aleatório
Métrica: Recall@50
Friedman pvalue = 0,276425008108561



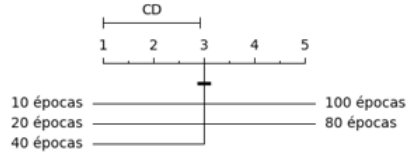
Modelo: Base Modelo com positivos
Inicialização: Aleatório
Métrica: NDCG@50
Friedman pvalue = 0,735758882342884

Figura 53 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização Aleatório, Modelo de Recomendação "Base Modelo com Positivos"

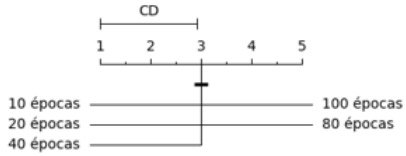
Fonte: Figura do autor



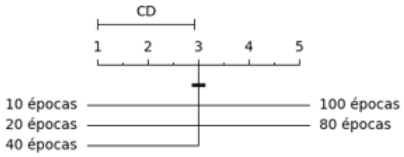
Modelo: Base TOPPOP com positivos
Inicialização: Aleatório
Métrica: Recall@10
Friedman pvalue = NaN (Not a Number)



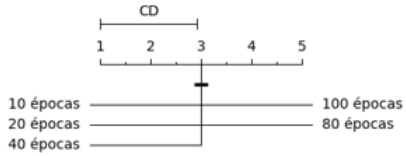
Modelo: Base TOPPOP com positivos
Inicialização: Aleatório
Métrica: NDCG@10
Friedman pvalue = NaN (Not a Number)



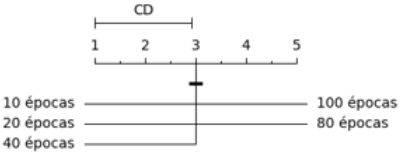
Modelo: Base TOPPOP com positivos
Inicialização: Aleatório
Métrica: Recall@20
Friedman pvalue = NaN (Not a Number)



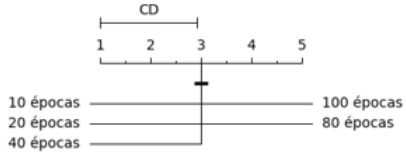
Modelo: Base TOPPOP com positivos
Inicialização: Aleatório
Métrica: NDCG@20
Friedman pvalue = NaN (Not a Number)



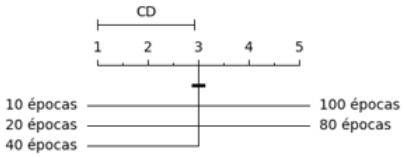
Modelo: Base TOPPOP com positivos
Inicialização: Aleatório
Métrica: Recall@30
Friedman pvalue = NaN (Not a Number)



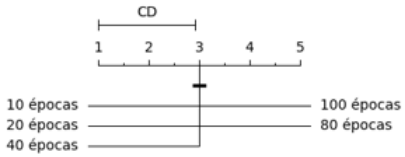
Modelo: Base TOPPOP com positivos
Inicialização: Aleatório
Métrica: NDCG@30
Friedman pvalue = NaN (Not a Number)



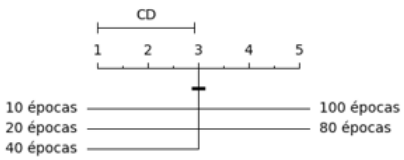
Modelo: Base TOPPOP com positivos
Inicialização: Aleatório
Métrica: Recall@40
Friedman pvalue = NaN (Not a Number)



Modelo: Base TOPPOP com positivos
Inicialização: Aleatório
Métrica: NDCG@40
Friedman pvalue = NaN (Not a Number)



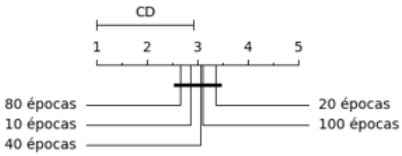
Modelo: Base TOPPOP com positivos
Inicialização: Aleatório
Métrica: Recall@50
Friedman pvalue = NaN (Not a Number)



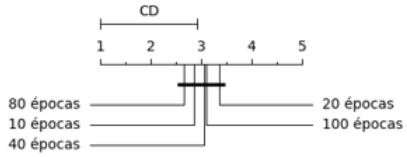
Modelo: Base TOPPOP com positivos
Inicialização: Aleatório
Métrica: NDCG@50
Friedman pvalue = NaN (Not a Number)

Figura 54 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização Aleatório, Modelo de Recomendação "Base TOPPOP com Positivos"

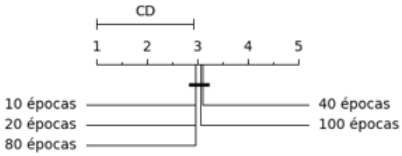
Fonte: Figura do autor



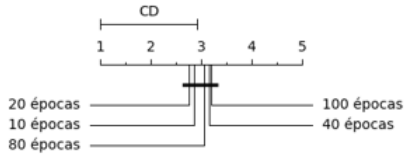
Modelo: Base Aleatório com positivos
Inicialização: *Embedding*
Métrica: Recall@10
Friedman pvalue = 0,452192256454332



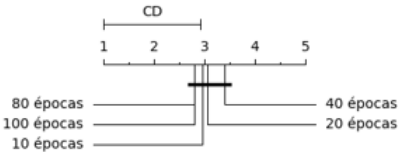
Modelo: Base Aleatório com positivos
Inicialização: *Embedding*
Métrica: NDCG@10
Friedman pvalue = 0,452192256454332



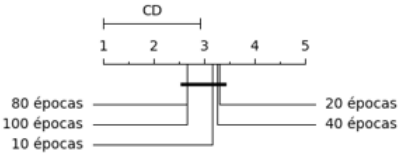
Modelo: Base Aleatório com positivos
Inicialização: *Embedding*
Métrica: Recall@20
Friedman pvalue = 0,996647203530265



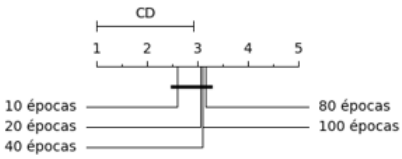
Modelo: Base Aleatório com positivos
Inicialização: *Embedding*
Métrica: NDCG@20
Friedman pvalue = 0,916928566003763



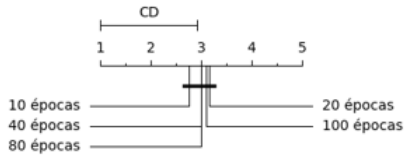
Modelo: Base Aleatório com positivos
Inicialização: *Embedding*
Métrica: Recall@30
Friedman pvalue = 0,845918889354662



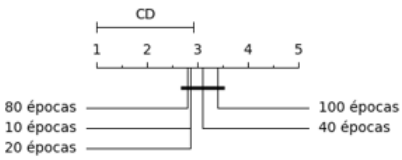
Modelo: Base Aleatório com positivos
Inicialização: *Embedding*
Métrica: NDCG@30
Friedman pvalue = 0,77130178094235



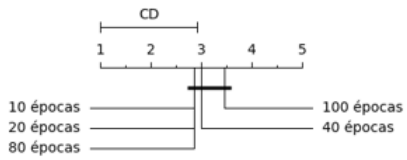
Modelo: Base Aleatório com positivos
Inicialização: *Embedding*
Métrica: Recall@40
Friedman pvalue = 0,896472428630878



Modelo: Base Aleatório com positivos
Inicialização: *Embedding*
Métrica: NDCG@40
Friedman pvalue = 0,980210152798874



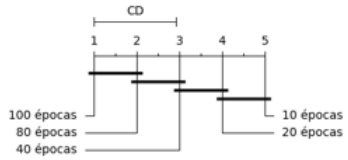
Modelo: Base Aleatório com positivos
Inicialização: *Embedding*
Métrica: Recall@50
Friedman pvalue = 0,862877087895892



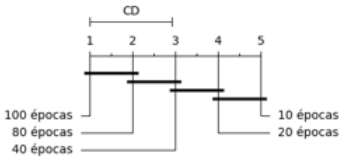
Modelo: Base Aleatório com positivos
Inicialização: *Embedding*
Métrica: NDCG@50
Friedman pvalue = 0,884429859912257

Figura 55 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização *Embedding*, Modelo de Recomendação "Base Aleatório com Positivos"

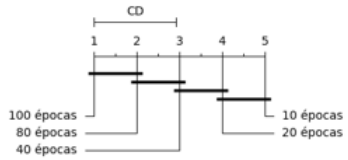
Fonte: Figura do autor



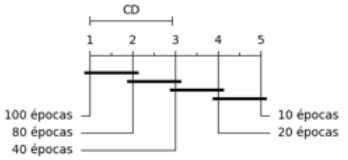
Modelo: Base Completa com positivos
Inicialização: *Embedding*
Métrica: Recall@10
Friedman pvalue = 4,32842260712096E-08



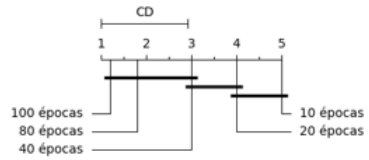
Modelo: Base Completa com positivos
Inicialização: *Embedding*
Métrica: NDCG@10
Friedman pvalue = 4,32842260712096E-08



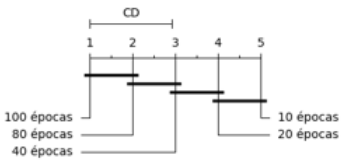
Modelo: Base Completa com positivos
Inicialização: *Embedding*
Métrica: Recall@20
Friedman pvalue = 4,32842260712096E-08



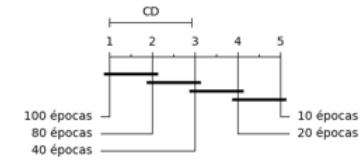
Modelo: Base Completa com positivos
Inicialização: *Embedding*
Métrica: NDCG@20
Friedman pvalue = 4,32842260712096E-08



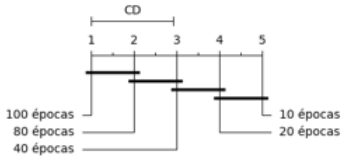
Modelo: Base Completa com positivos
Inicialização: *Embedding*
Métrica: Recall@30
Friedman pvalue = 7,95859865216326E-08



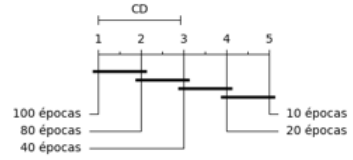
Modelo: Base Completa com positivos
Inicialização: *Embedding*
Métrica: NDCG@30
Friedman pvalue = 4,32842260712096E-08



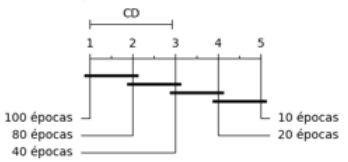
Modelo: Base Completa com positivos
Inicialização: *Embedding*
Métrica: Recall@40
Friedman pvalue = 4,32842260712096E-08



Modelo: Base Completa com positivos
Inicialização: *Embedding*
Métrica: NDCG@40
Friedman pvalue = 4,32842260712096E-08



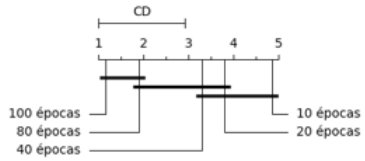
Modelo: Base Completa com positivos
Inicialização: *Embedding*
Métrica: Recall@50
Friedman pvalue = 4,32842260712096E-08



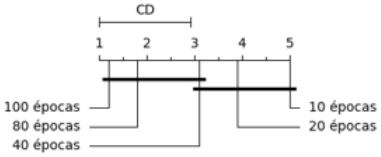
Modelo: Base Completa com positivos
Inicialização: *Embedding*
Métrica: NDCG@50
Friedman pvalue = 4,32842260712096E-08

Figura 56 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização *Embedding*, Modelo de Recomendação "Base Completa com Positivos"

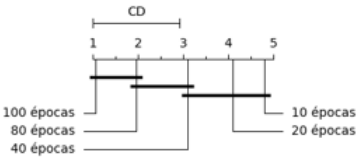
Fonte: Figura do autor



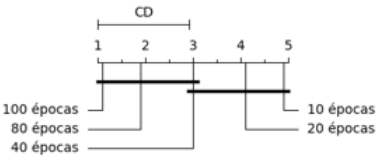
Modelo: Base Completa sem positivos
Inicialização: *Embedding*
Métrica: Recall@10
Friedman pvalue = 2,37769083473926E-07



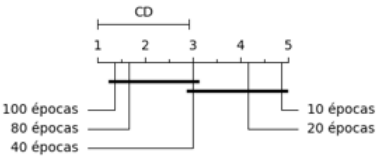
Modelo: Base Completa sem positivos
Inicialização: *Embedding*
Métrica: NDCG@10
Friedman pvalue = 1,12055928750745E-07



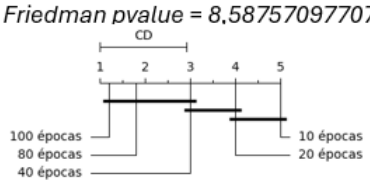
Modelo: Base Completa sem positivos
Inicialização: *Embedding*
Métrica: Recall@20
Friedman pvalue = 7,59390759174533E-08



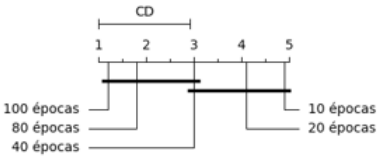
Modelo: Base Completa sem positivos
Inicialização: *Embedding*
Métrica: NDCG@20
Friedman pvalue = 8,58757097707038E-08



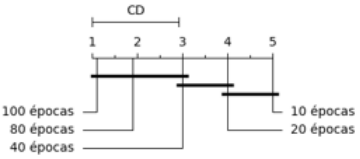
Modelo: Base Completa sem positivos
Inicialização: *Embedding*
Métrica: Recall@30
Friedman pvalue = 1,16486051404173E-07



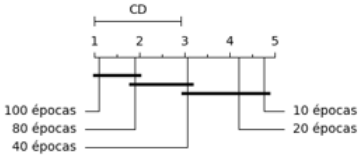
Modelo: Base Completa sem positivos
Inicialização: *Embedding*
Métrica: NDCG@30
Friedman pvalue = 7.95859865216326E-08



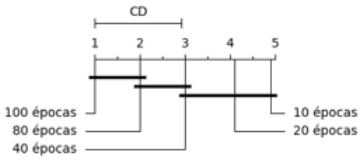
Modelo: Base Completa sem positivos
Inicialização: *Embedding*
Métrica: Recall@40
Friedman pvalue = 7,75129652134012E-08



Modelo: Base Completa sem positivos
Inicialização: *Embedding*
Métrica: NDCG@40
Friedman pvalue = 6,09770020529556E-08



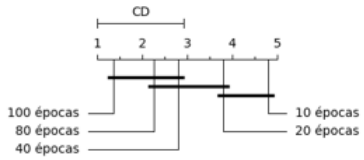
Modelo: Base Completa sem positivos
Inicialização: *Embedding*
Métrica: Recall@50
Friedman pvalue = 9,92044408968387E-08



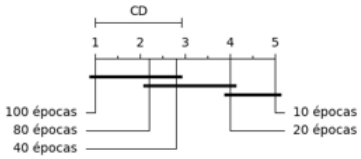
Modelo: Base Completa sem positivos
Inicialização: *Embedding*
Métrica: NDCG@50
Friedman pvalue = 6,09770020529556E-08

Figura 57 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização *Embedding*, Modelo de Recomendação "Base Completa sem Positivos"

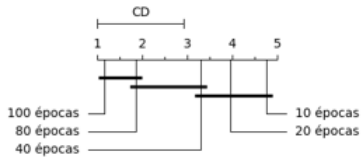
Fonte: Figura do autor



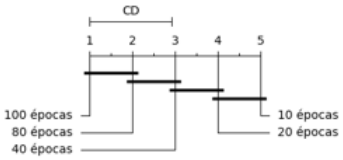
Modelo: Base Modelo com positivos
Inicialização: *Embedding*
Métrica: Recall@10
Friedman pvalue = 5,60336677234057E-06



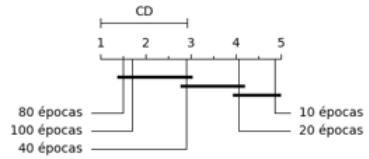
Modelo: Base Modelo com positivos
Inicialização: *Embedding*
Métrica: NDCG@10
Friedman pvalue = 7,95859865216326E-08



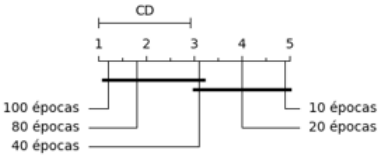
Modelo: Base Modelo com positivos
Inicialização: *Embedding*
Métrica: Recall@20
Friedman pvalue = 1,6008385553517E-07



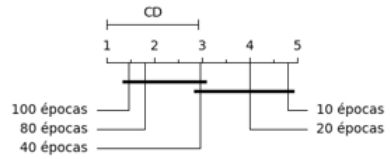
Modelo: Base Modelo com positivos
Inicialização: *Embedding*
Métrica: NDCG@20
Friedman pvalue = 4,32842260712096E-08



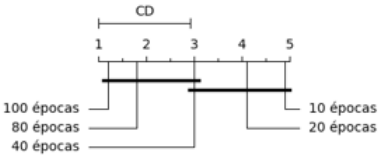
Modelo: Base Modelo com positivos
Inicialização: *Embedding*
Métrica: Recall@30
Friedman pvalue = 3,67123637933161E-07



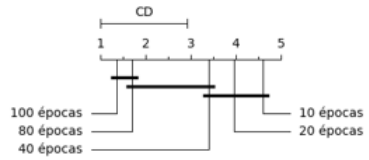
Modelo: Base Modelo com positivos
Inicialização: *Embedding*
Métrica: NDCG@30
Friedman pvalue = 1,63824445986941E-07



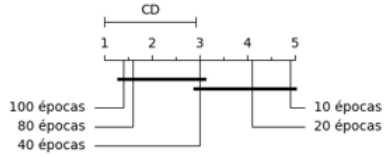
Modelo: Base Modelo com positivos
Inicialização: *Embedding*
Métrica: Recall@40
Friedman pvalue = 5,16883866197466E-07



Modelo: Base Modelo com positivos
Inicialização: *Embedding*
Métrica: NDCG@40
Friedman pvalue = 1,12055928750745E-07



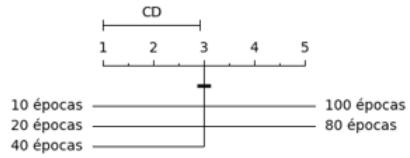
Modelo: Base Modelo com positivos
Inicialização: *Embedding*
Métrica: Recall@50
Friedman pvalue = 7,41474292283749E-07



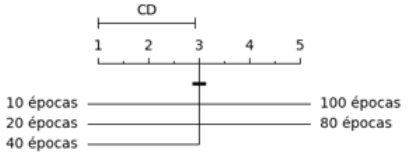
Modelo: Base Modelo com positivos
Inicialização: *Embedding*
Métrica: NDCG@50
Friedman pvalue = 1,51846285350844E-07

Figura 58 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização *Embedding*, Modelo de Recomendação "Base Modelo com Positivos"

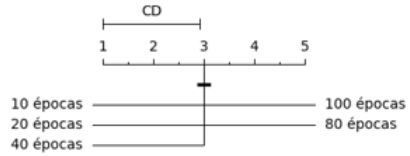
Fonte: Figura do autor



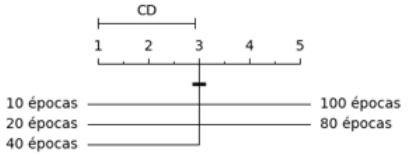
Modelo: Base TOPPOP com positivos
Inicialização: *Embedding*
Métrica: Recall@10
Friedman pvalue = NaN (Not a Number)



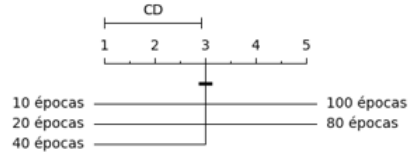
Modelo: Base TOPPOP com positivos
Inicialização: *Embedding*
Métrica: NDCG@10
Friedman pvalue = NaN (Not a Number)



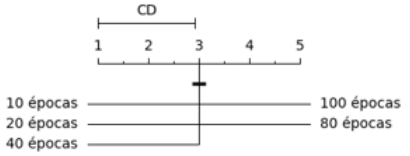
Modelo: Base TOPPOP com positivos
Inicialização: *Embedding*
Métrica: Recall@20
Friedman pvalue = NaN (Not a Number)



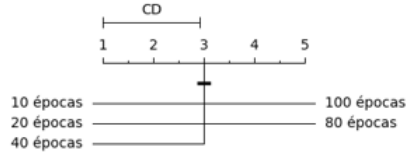
Modelo: Base TOPPOP com positivos
Inicialização: *Embedding*
Métrica: NDCG@20
Friedman pvalue = NaN (Not a Number)



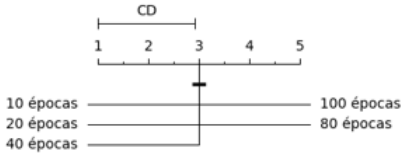
Modelo: Base TOPPOP com positivos
Inicialização: *Embedding*
Métrica: Recall@30
Friedman pvalue = NaN (Not a Number)



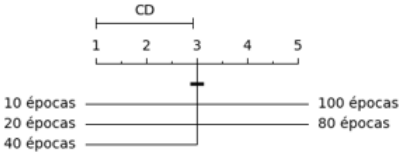
Modelo: Base TOPPOP com positivos
Inicialização: *Embedding*
Métrica: NDCG@30
Friedman pvalue = NaN (Not a Number)



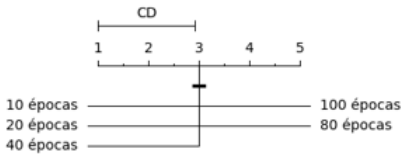
Modelo: Base TOPPOP com positivos
Inicialização: *Embedding*
Métrica: Recall@40
Friedman pvalue = NaN (Not a Number)



Modelo: Base TOPPOP com positivos
Inicialização: *Embedding*
Métrica: NDCG@40
Friedman pvalue = NaN (Not a Number)



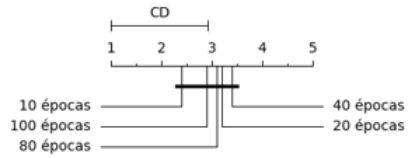
Modelo: Base TOPPOP com positivos
Inicialização: *Embedding*
Métrica: Recall@50
Friedman pvalue = NaN (Not a Number)



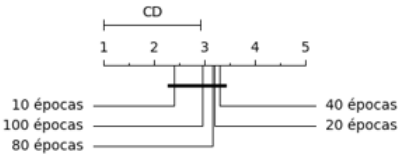
Modelo: Base TOPPOP com positivos
Inicialização: *Embedding*
Métrica: NDCG@50
Friedman pvalue = NaN (Not a Number)

Figura 59 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização *Embedding*, Modelo de Recomendação "Base TOPPOP com Positivos"

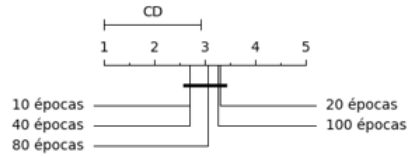
Fonte: Figura do autor



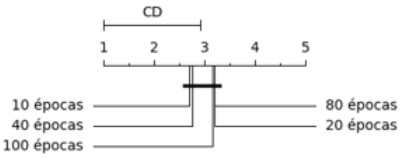
Modelo: Base Aleatório com positivos
Inicialização: Média
Métrica: Recall@10
Friedman pvalue = 0,30482993448164



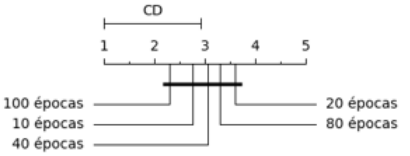
Modelo: Base Aleatório com positivos
Inicialização: Média
Métrica: NDCG@10
Friedman pvalue = 0,400724598883387



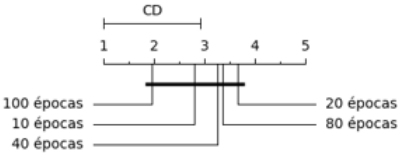
Modelo: Base Aleatório com positivos
Inicialização: Média
Métrica: Recall@20
Friedman pvalue = 0,763639170246016



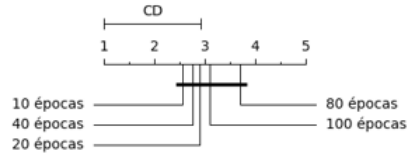
Modelo: Base Aleatório com positivos
Inicialização: Média
Métrica: NDCG@20
Friedman pvalue = 0,855695198387651



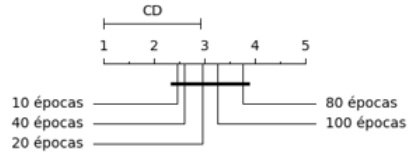
Modelo: Base Aleatório com positivos
Inicialização: Média
Métrica: Recall@30
Friedman pvalue = 0,275104050799818



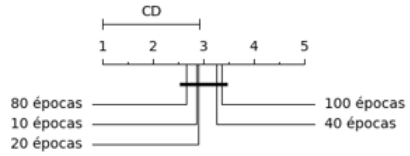
Modelo: Base Aleatório com positivos
Inicialização: Média
Métrica: NDCG@30
Friedman pvalue = 0,107010692344625



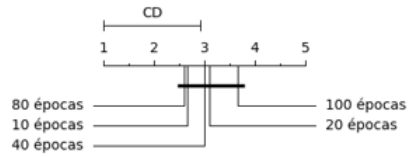
Modelo: Base Aleatório com positivos
Inicialização: Média
Métrica: Recall@40
Friedman pvalue = 0,477448948520737



Modelo: Base Aleatório com positivos
Inicialização: Média
Métrica: NDCG@40
Friedman pvalue = 0,329271160312194



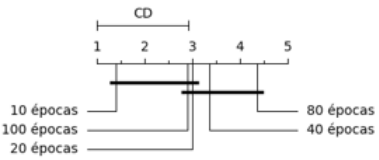
Modelo: Base Aleatório com positivos
Inicialização: Média
Métrica: Recall@50
Friedman pvalue = 0,808792135410996



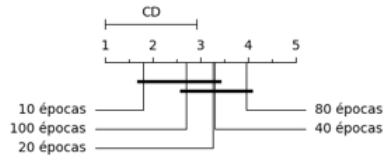
Modelo: Base Aleatório com positivos
Inicialização: Média
Métrica: NDCG@50
Friedman pvalue = 0,56648829579412

Figura 60 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização Média, Modelo de Recomendação "Base Aleatório com Positivos"

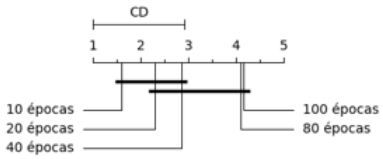
Fonte: Figura do autor



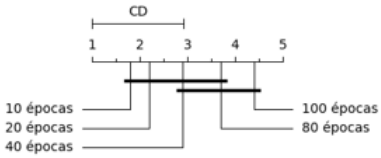
Modelo: Base Completa com positivos
Inicialização: Média
Métrica: Recall@10
Friedman pvalue = 0,00019819231157968



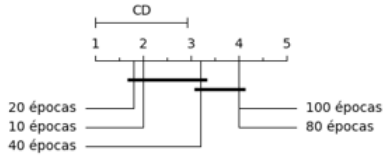
Modelo: Base Completa com positivos
Inicialização: Média
Métrica: NDCG@10
Friedman pvalue = 0,0265640143500163



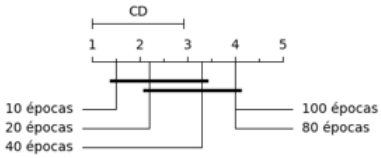
Modelo: Base Completa com positivos
Inicialização: Média
Métrica: Recall@20
Friedman pvalue = 0,00024890372810298



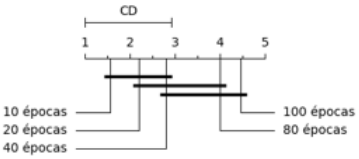
Modelo: Base Completa com positivos
Inicialização: Média
Métrica: NDCG@20
Friedman pvalue = 0,0011483298031644



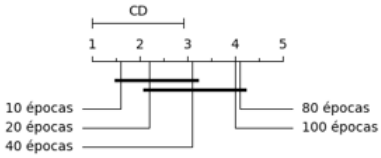
Modelo: Base Completa com positivos
Inicialização: Média
Métrica: Recall@30
Friedman pvalue = 0,00108509535086316



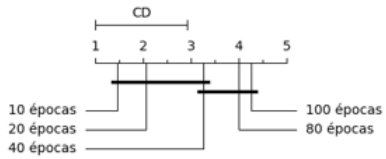
Modelo: Base Completa com positivos
Inicialização: Média
Métrica: NDCG@30
Friedman pvalue = 0,000517889987046312



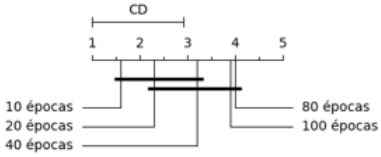
Modelo: Base Completa com positivos
Inicialização: Média
Métrica: Recall@40
Friedman pvalue = 0,000079125915860499



Modelo: Base Completa com positivos
Inicialização: Média
Métrica: NDCG@40
Friedman pvalue = 0,000692377302513856



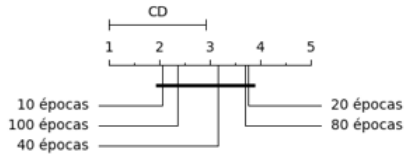
Modelo: Base Completa com positivos
Inicialização: Média
Métrica: Recall@50
Friedman pvalue = 0,0000575798302616858



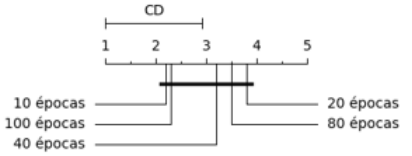
Modelo: Base Completa com positivos
Inicialização: Média
Métrica: NDCG@50
Friedman pvalue = 0,00176741561920874

Figura 61 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização Média, Modelo de Recomendação "Base Completa com Positivos"

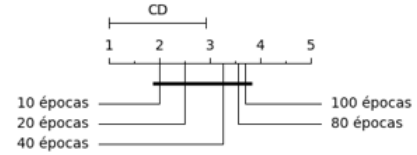
Fonte: Figura do autor



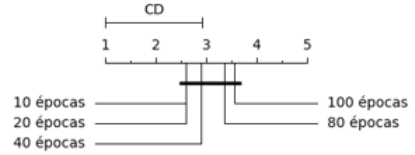
Modelo: Base Completa sem positivos
Inicialização: Média
Métrica: Recall@10
Friedman pvalue = 0,0215111840142601



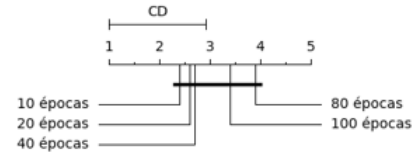
Modelo: Base Completa sem positivos
Inicialização: Média
Métrica: NDCG@10
Friedman pvalue = 0,0634378331091099



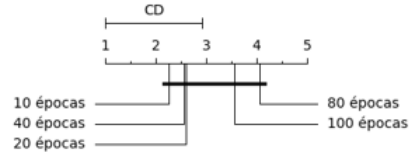
Modelo: Base Completa sem positivos
Inicialização: Média
Métrica: Recall@20
Friedman pvalue = 0,0505616081611062



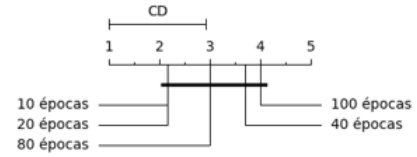
Modelo: Base Completa sem positivos
Inicialização: Média
Métrica: NDCG@20
Friedman pvalue = 0,541654203981598



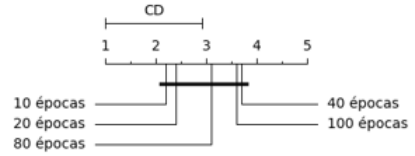
Modelo: Base Completa sem positivos
Inicialização: Média
Métrica: Recall@30
Friedman pvalue = 0,153333613725754



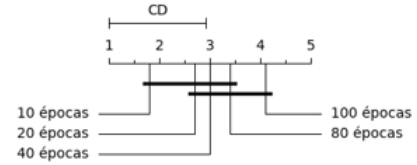
Modelo: Base Completa sem positivos
Inicialização: Média
Métrica: NDCG@30
Friedman pvalue = 0,0515416620732086



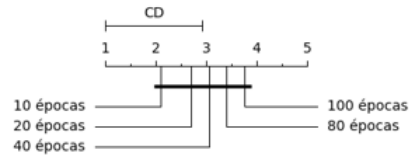
Modelo: Base Completa sem positivos
Inicialização: Média
Métrica: Recall@40
Friedman pvalue = 0,0144661506721558



Modelo: Base Completa sem positivos
Inicialização: Média
Métrica: NDCG@40
Friedman pvalue = 0,114384328231662



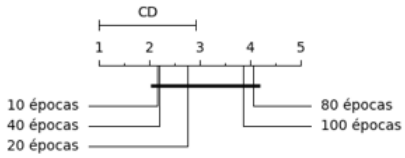
Modelo: Base Completa sem positivos
Inicialização: Média
Métrica: Recall@50
Friedman pvalue = 0,0176605639796262



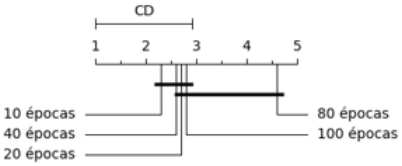
Modelo: Base Completa sem positivos
Inicialização: Média
Métrica: NDCG@50
Friedman pvalue = 0,162743929731693

Figura 62 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização Média, Modelo de Recomendação "Base Completa sem Positivos"

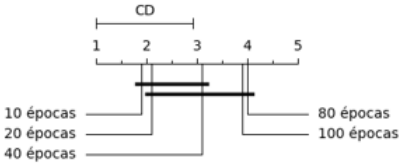
Fonte: Figura do autor



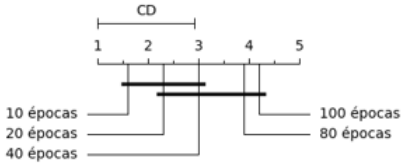
Modelo: Base Modelo com positivos
Inicialização: Média
Métrica: Recall@10
Friedman pvalue = 0,00891111482569499



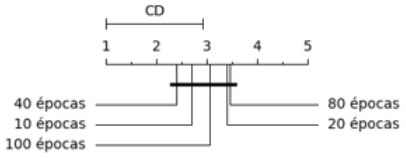
Modelo: Base Modelo com positivos
Inicialização: Média
Métrica: NDCG@10
Friedman pvalue = 0,00964437482927893



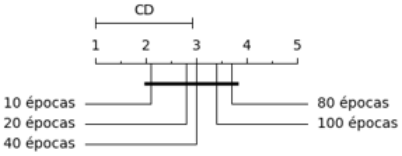
Modelo: Base Modelo com positivos
Inicialização: Média
Métrica: Recall@20
Friedman pvalue = 0,00374379067670379



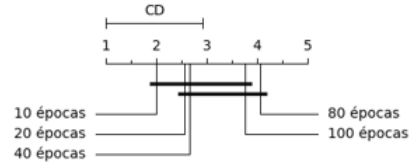
Modelo: Base Modelo com positivos
Inicialização: Média
Métrica: NDCG@20
Friedman pvalue = 0,000860330281788971



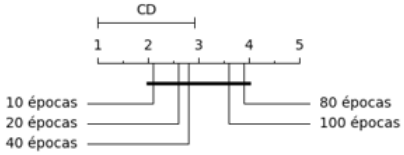
Modelo: Base Modelo com positivos
Inicialização: Média
Métrica: Recall@30
Friedman pvalue = 0,488357880433489



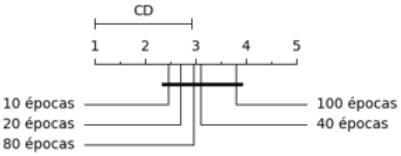
Modelo: Base Modelo com positivos
Inicialização: Média
Métrica: NDCG@30
Friedman pvalue = 0,199148273471455



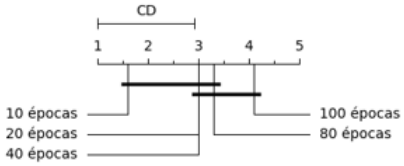
Modelo: Base Modelo com positivos
Inicialização: Média
Métrica: Recall@40
Friedman pvalue = 0,016325046673466



Modelo: Base Modelo com positivos
Inicialização: Média
Métrica: NDCG@40
Friedman pvalue = 0,0684921578015117



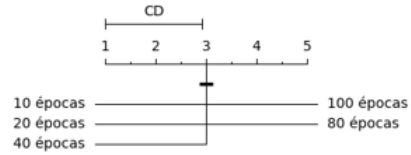
Modelo: Base Modelo com positivos
Inicialização: Média
Métrica: Recall@50
Friedman pvalue = 0,36576469879051



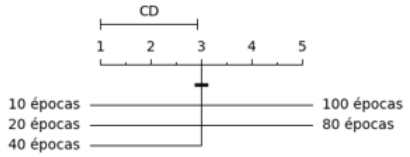
Modelo: Base Modelo com positivos
Inicialização: Média
Métrica: NDCG@50
Friedman pvalue = 0,011081991649702

Figura 63 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização Média, Modelo de Recomendação "Base Modelo com Positivos"

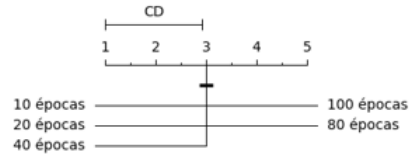
Fonte: Figura do autor



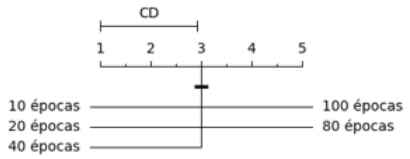
Modelo: Base TOPPOP com positivos
Inicialização: Média
Métrica: Recall@10
Friedman pvalue = NaN (Not a Number)



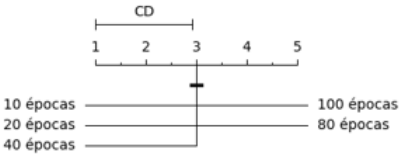
Modelo: Base TOPPOP com positivos
Inicialização: Média
Métrica: NDCG@10
Friedman pvalue = NaN (Not a Number)



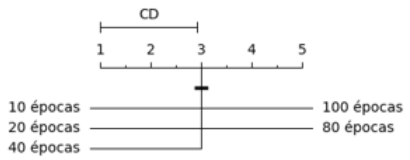
Modelo: Base TOPPOP com positivos
Inicialização: Média
Métrica: Recall@20
Friedman pvalue = NaN (Not a Number)



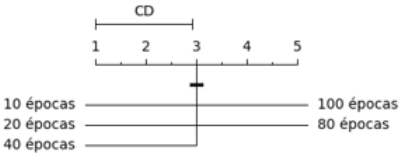
Modelo: Base TOPPOP com positivos
Inicialização: Média
Métrica: NDCG@20
Friedman pvalue = NaN (Not a Number)



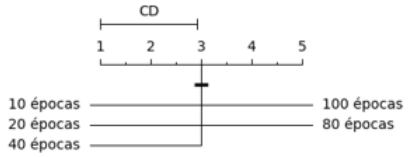
Modelo: Base TOPPOP com positivos
Inicialização: Média
Métrica: Recall@30
Friedman pvalue = NaN (Not a Number)



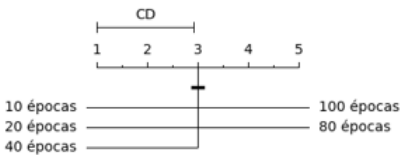
Modelo: Base TOPPOP com positivos
Inicialização: Média
Métrica: NDCG@30
Friedman pvalue = NaN (Not a Number)



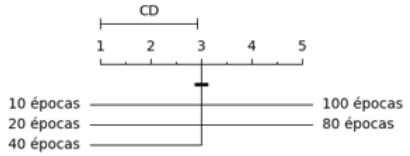
Modelo: Base TOPPOP com positivos
Inicialização: Média
Métrica: Recall@40
Friedman pvalue = NaN (Not a Number)



Modelo: Base TOPPOP com positivos
Inicialização: Média
Métrica: NDCG@40
Friedman pvalue = NaN (Not a Number)



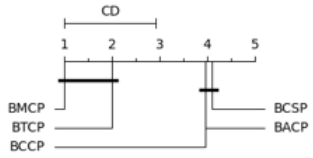
Modelo: Base TOPPOP com positivos
Inicialização: Média
Métrica: Recall@50
Friedman pvalue = NaN (Not a Number)



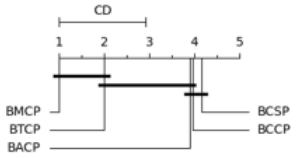
Modelo: Base TOPPOP com positivos
Inicialização: Média
Métrica: NDCG@50
Friedman pvalue = NaN (Not a Number)

Figura 64 – Comparação entre as épocas de treinamento utilizando: Método de Inicialização Média, Modelo de Recomendação "Base TOPPOP com Positivos"

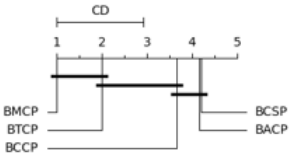
Fonte: Figura do autor



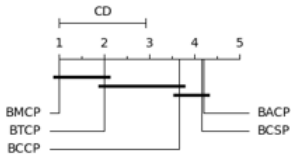
Inicialização: Aleatório
Épocas:10
Métrica: Recall@10
Friedman pvalue = 8,32032455152762E-08



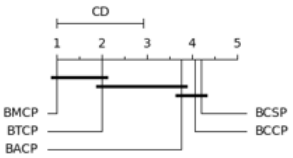
Inicialização: Aleatório
Épocas:10
Métrica: NDCG@10
Friedman pvalue = 8,87381452869023E-08



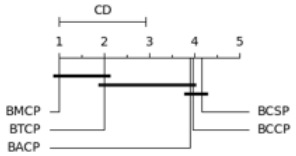
Inicialização: Aleatório
Épocas:10
Métrica: Recall @20
Friedman pvalue = 1,63471493697245E-07



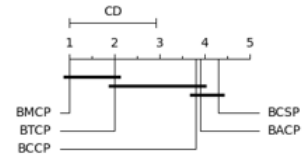
Inicialização: Aleatório
Épocas:10
Métrica: NDCG@20
Friedman pvalue = 3,20930869945861E-07



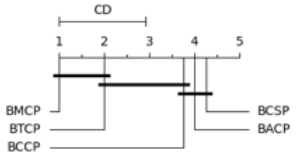
Inicialização: Aleatório
Épocas:10
Métrica: Recall @30
Friedman pvalue = 1,75788897556177E-07



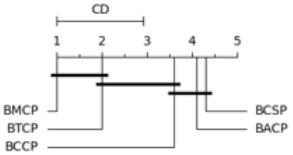
Inicialização: Aleatório
Épocas:10
Métrica: NDCG@30
Friedman pvalue = 5,23833344345205E-07



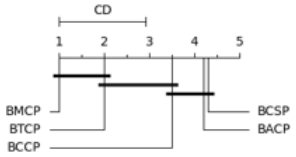
Inicialização: Aleatório
Épocas:10
Métrica: Recall @40
Friedman pvalue = 2,65996561038506E-07



Inicialização: Aleatório
Épocas:10
Métrica: NDCG@40
Friedman pvalue = 6,19360588220407E-07



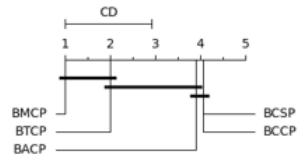
Inicialização: Aleatório
Épocas:10
Métrica: Recall @50
Friedman pvalue = 4,32934179462961E-07



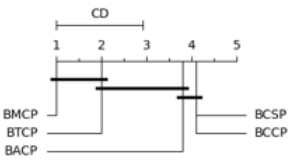
Inicialização: Aleatório
Épocas:10
Métrica: NDCG@50
Friedman pvalue = 6,76714658550388E-07

Figura 65 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização Aleatório, 10 Épocas

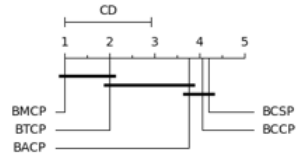
Fonte: Figura do autor



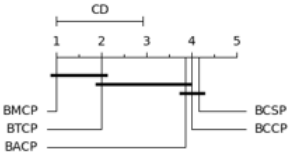
Inicialização: Aleatório
Épocas:20
Métrica: Recall@10
Friedman pvalue = 1,15249981821593E-07



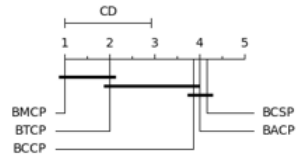
Inicialização: Aleatório
Épocas:20
Métrica: NDCG@10
Friedman pvalue = 1,4287053364552E-07



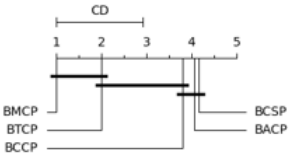
Inicialização: Aleatório
Épocas:20
Métrica: Recall @20
Friedman pvalue = 2,36369195567566E-07



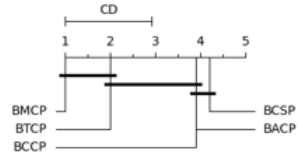
Inicialização: Aleatório
Épocas:20
Métrica: NDCG@20
Friedman pvalue = 3,56737362054677E-07



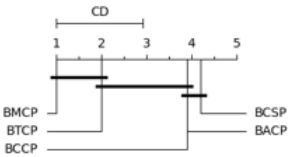
Inicialização: Aleatório
Épocas:20
Métrica: Recall @30
Friedman pvalue = 4,69335759539846E-07



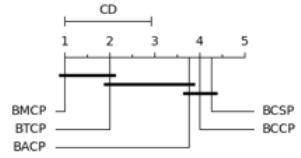
Inicialização: Aleatório
Épocas:20
Métrica: NDCG@30
Friedman pvalue = 4,92666737417971E-07



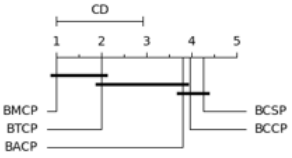
Inicialização: Aleatório
Épocas:20
Métrica: Recall @40
Friedman pvalue = 3,14789563953327E-07



Inicialização: Aleatório
Épocas:20
Métrica: NDCG@40
Friedman pvalue = 3,14789563953327E-07



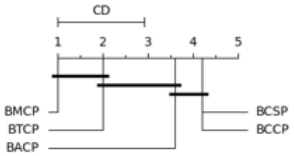
Inicialização: Aleatório
Épocas:20
Métrica: Recall @50
Friedman pvalue = 4,35770029515963E-07



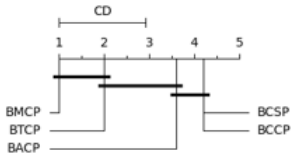
Inicialização: Aleatório
Épocas:20
Métrica: NDCG@50
Friedman pvalue = 6,44636576882386E-07

Figura 66 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização Aleatório, 20 Épocas

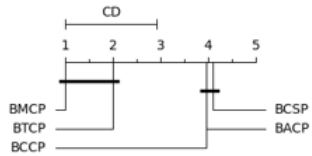
Fonte: Figura do autor



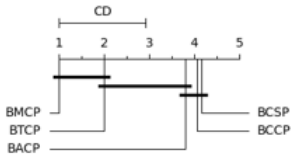
Inicialização: Aleatório
Épocas:40
Métrica: Recall@10
Friedman pvalue = 9,59967177096615E-08



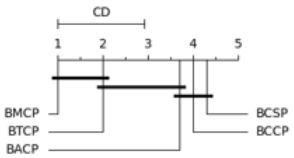
Inicialização: Aleatório
Épocas:40
Métrica: NDCG@10
Friedman pvalue = 9,59967177096615E-08



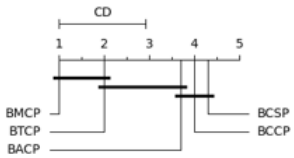
Inicialização: Aleatório
Épocas:40
Métrica: Recall @20
Friedman pvalue = 2,13664422639631E-07



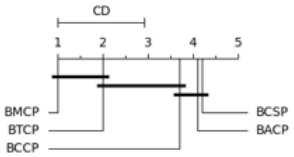
Inicialização: Aleatório
Épocas:40
Métrica: NDCG@20
Friedman pvalue = 2,83307783530139E-07



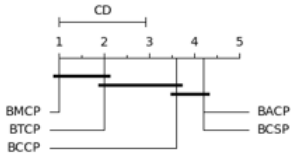
Inicialização: Aleatório
Épocas:40
Métrica: Recall @30
Friedman pvalue = 3,55522491193517E-07



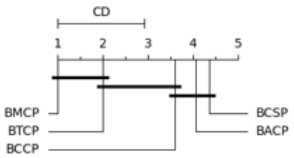
Inicialização: Aleatório
Épocas:40
Métrica: NDCG@30
Friedman pvalue = 3,55522491193517E-07



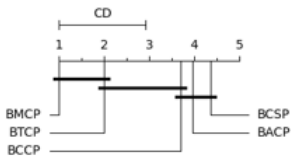
Inicialização: Aleatório
Épocas:40
Métrica: Recall @40
Friedman pvalue = 4,22600554655937E-07



Inicialização: Aleatório
Épocas:40
Métrica: NDCG@40
Friedman pvalue = 5,36670101614192E-07



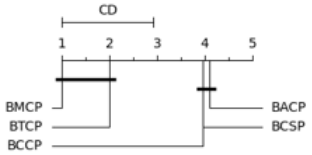
Inicialização: Aleatório
Épocas:40
Métrica: Recall @50
Friedman pvalue = 1,95977599736207E-07



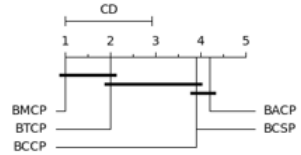
Inicialização: Aleatório
Épocas:40
Métrica: NDCG@50
Friedman pvalue = 6,14450921983921E-07

Figura 67 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização Aleatório, 40 Épocas

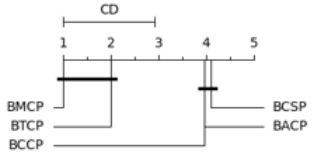
Fonte: Figura do autor



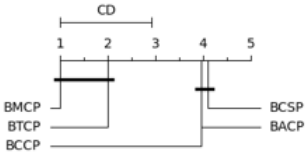
Inicialização: Aleatório
Épocas:80
Métrica: Recall@10
Friedman pvalue = 1,15249981821593E-07



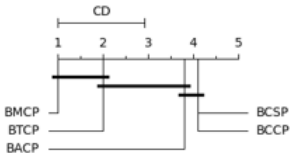
Inicialização: Aleatório
Épocas:80
Métrica: NDCG@10
Friedman pvalue = 1,4287053364552E-07



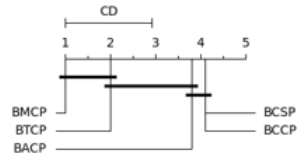
Inicialização: Aleatório
Épocas:80
Métrica: Recall @20
Friedman pvalue = 2,8630568031139E-07



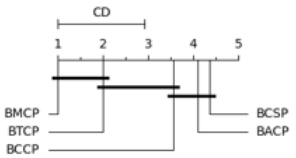
Inicialização: Aleatório
Épocas:80
Métrica: NDCG@20
Friedman pvalue = 2,8630568031139E-07



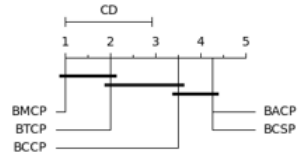
Inicialização: Aleatório
Épocas:80
Métrica: Recall @30
Friedman pvalue = 2,60145609046379E-07



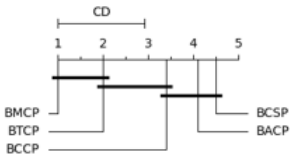
Inicialização: Aleatório
Épocas:80
Métrica: NDCG@30
Friedman pvalue = 4,55079367066224E-07



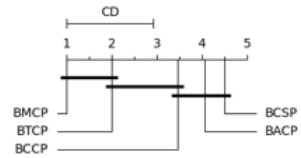
Inicialização: Aleatório
Épocas:80
Métrica: Recall @40
Friedman pvalue = 4,06821844054971E-07



Inicialização: Aleatório
Épocas:80
Métrica: NDCG@40
Friedman pvalue = 5,31547093920179E-07



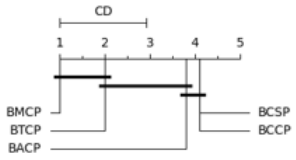
Inicialização: Aleatório
Épocas:80
Métrica: Recall @50
Friedman pvalue = 1,90951398253893E-07



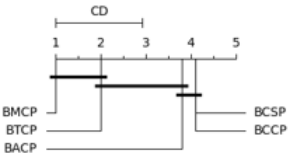
Inicialização: Aleatório
Épocas:80
Métrica: NDCG@50
Friedman pvalue = 2,38927286694425E-07

Figura 68 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização Aleatório, 80 Épocas

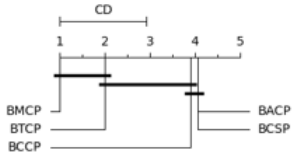
Fonte: Figura do autor



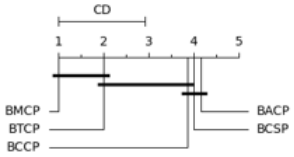
Inicialização: Aleatório
Épocas:100
Métrica: Recall@10
Friedman pvalue = 1,4287053364552E-07



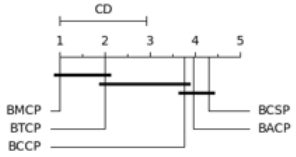
Inicialização: Aleatório
Épocas:100
Métrica: NDCG@10
Friedman pvalue = 1,4287053364552E-07



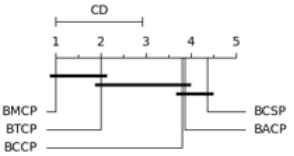
Inicialização: Aleatório
Épocas:100
Métrica: Recall @20
Friedman pvalue = 1,15249981821593E-07



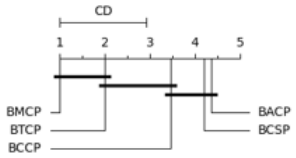
Inicialização: Aleatório
Épocas:100
Métrica: NDCG@20
Friedman pvalue = 1,47680954631934E-07



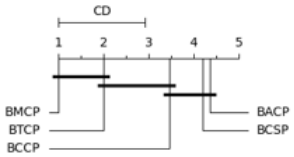
Inicialização: Aleatório
Épocas:100
Métrica: Recall @30
Friedman pvalue = 3,11242108219785E-07



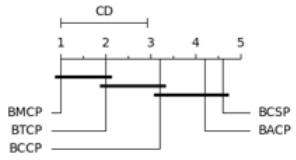
Inicialização: Aleatório
Épocas:100
Métrica: NDCG@30
Friedman pvalue = 4,61023341185083E-07



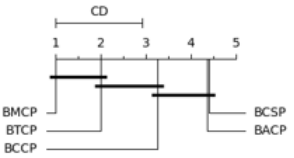
Inicialização: Aleatório
Épocas:100
Métrica: Recall @40
Friedman pvalue = 2,86467838114539E-07



Inicialização: Aleatório
Épocas:100
Métrica: NDCG@40
Friedman pvalue = 2,86467838114539E-07



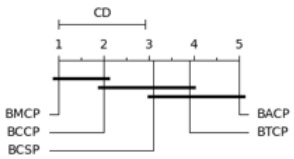
Inicialização: Aleatório
Épocas:100
Métrica: Recall @50
Friedman pvalue = 1,72832589575255E-07



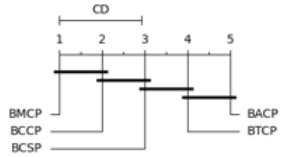
Inicialização: Aleatório
Épocas:100
Métrica: NDCG@50
Friedman pvalue = 3,27694816223182E-07

Figura 69 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização Aleatório, 100 Épocas

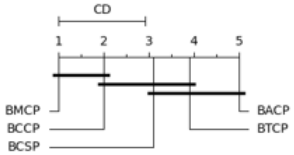
Fonte: Figura do autor



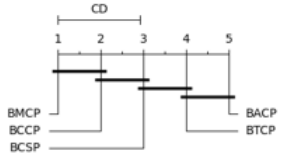
Inicialização: *Embedding*
Épocas:10
Métrica: Recall@10
Friedman pvalue = 6,09770020529557E-08



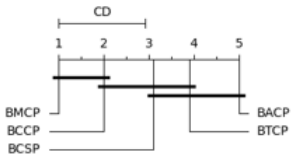
Inicialização: *Embedding*
Épocas:10
Métrica: NDCG@10
Friedman pvalue = 4,32842260712097E-08



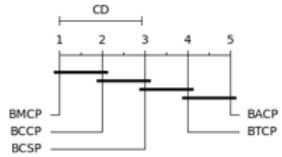
Inicialização: *Embedding*
Épocas:10
Métrica: Recall @20
Friedman pvalue = 6,09770020529557E-08



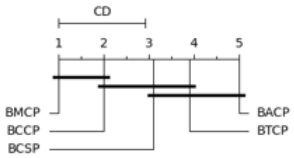
Inicialização: *Embedding*
Épocas:10
Métrica: NDCG@20
Friedman pvalue = 4,32842260712097E-08



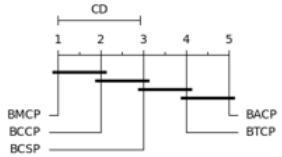
Inicialização: *Embedding*
Épocas:10
Métrica: Recall @30
Friedman pvalue = 6,09770020529557E-08



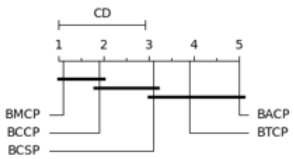
Inicialização: *Embedding*
Épocas:10
Métrica: NDCG@30
Friedman pvalue = 4,32842260712097E-08



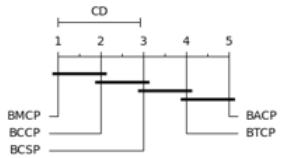
Inicialização: *Embedding*
Épocas:10
Métrica: Recall @40
Friedman pvalue = 6,09770020529557E-08



Inicialização: *Embedding*
Épocas:10
Métrica: NDCG@40
Friedman pvalue = 4,32842260712097E-08



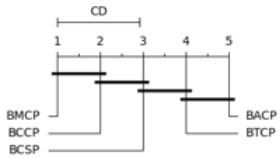
Inicialização: *Embedding*
Épocas:10
Métrica: Recall @50
Friedman pvalue = 8,58757097707038E-08



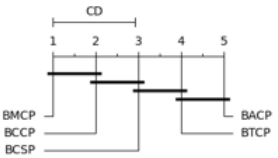
Inicialização: *Embedding*
Épocas:10
Métrica: NDCG@50
Friedman pvalue = 4,32842260712097E-08

Figura 70 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização *Embedding*, 10 Épocas

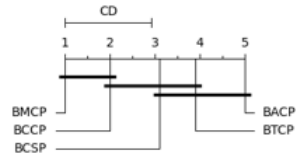
Fonte: Figura do autor



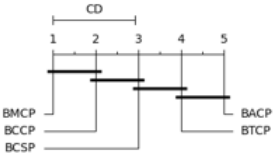
Inicialização: *Embedding*
Épocas:20
Métrica: Recall@10
Friedman pvalue = 4,32842260712097E-08



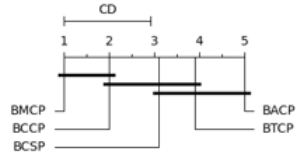
Inicialização: *Embedding*
Épocas:20
Métrica: NDCG@10
Friedman pvalue = 4,32842260712097E-08



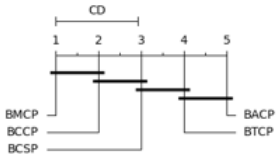
Inicialização: *Embedding*
Épocas:20
Métrica: Recall @20
Friedman pvalue = 6,09770020529557E-08



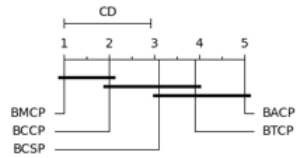
Inicialização: *Embedding*
Épocas:20
Métrica: NDCG@20
Friedman pvalue = 4,32842260712097E-08



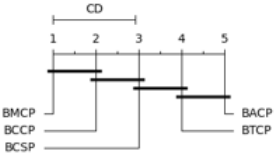
Inicialização: *Embedding*
Épocas:20
Métrica: Recall @30
Friedman pvalue = 6,09770020529557E-08



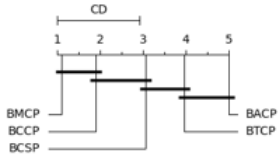
Inicialização: *Embedding*
Épocas:20
Métrica: NDCG@30
Friedman pvalue = 4,32842260712097E-08



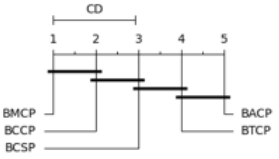
Inicialização: *Embedding*
Épocas:20
Métrica: Recall @40
Friedman pvalue = 6,09770020529557E-08



Inicialização: *Embedding*
Épocas:20
Métrica: NDCG@40
Friedman pvalue = 4,32842260712097E-08



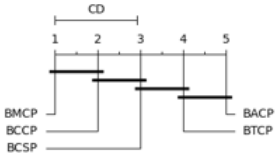
Inicialização: *Embedding*
Épocas:20
Métrica: Recall @50
Friedman pvalue = 6,65715985983287E-08



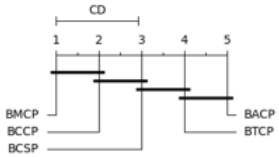
Inicialização: *Embedding*
Épocas:20
Métrica: NDCG@50
Friedman pvalue = 4,32842260712097E-08

Figura 71 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização *Embedding*, 20 Épocas

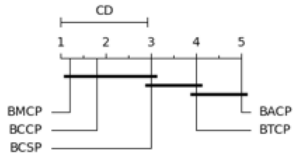
Fonte: Figura do autor



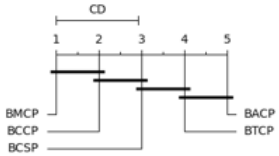
Inicialização: *Embedding*
Épocas:40
Métrica: Recall@10
Friedman pvalue = 4,32842260712097E-08



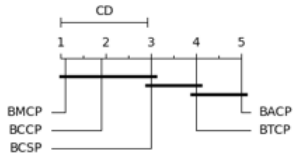
Inicialização: *Embedding*
Épocas:40
Métrica: NDCG@10
Friedman pvalue = 4,32842260712097E-08



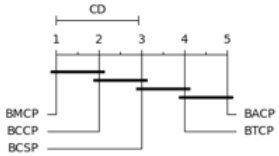
Inicialização: *Embedding*
Épocas:40
Métrica: Recall @20
Friedman pvalue = 7,95859865216327E-08



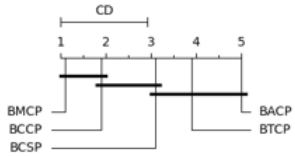
Inicialização: *Embedding*
Épocas:40
Métrica: NDCG@20
Friedman pvalue = 4,32842260712097E-08



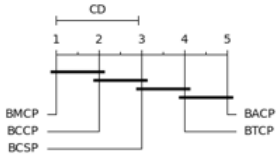
Inicialização: *Embedding*
Épocas:40
Métrica: Recall @30
Friedman pvalue = 6,09770020529557E-08



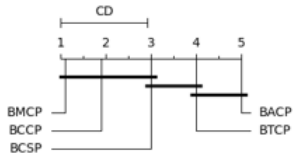
Inicialização: *Embedding*
Épocas:40
Métrica: NDCG@30
Friedman pvalue = 4,32842260712097E-08



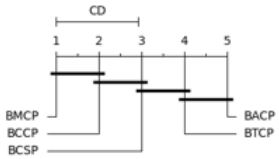
Inicialização: *Embedding*
Épocas:40
Métrica: Recall @40
Friedman pvalue = 8,58757097707038E-08



Inicialização: *Embedding*
Épocas:40
Métrica: NDCG@40
Friedman pvalue = 4,32842260712097E-08



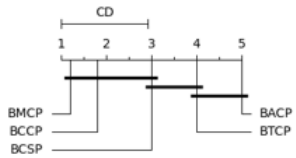
Inicialização: *Embedding*
Épocas:40
Métrica: Recall @50
Friedman pvalue = 6,09770020529557E-08



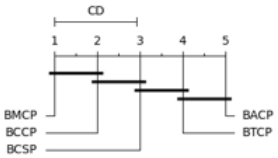
Inicialização: *Embedding*
Épocas:40
Métrica: NDCG@50
Friedman pvalue = 4,32842260712097E-08

Figura 72 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização *Embedding*, 40 Épocas

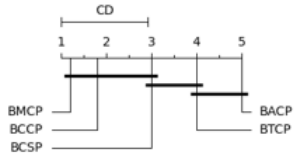
Fonte: Figura do autor



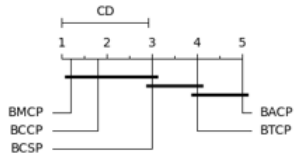
Inicialização: *Embedding*
Épocas:80
Métrica: Recall@10
Friedman pvalue = 7,95859865216327E-08



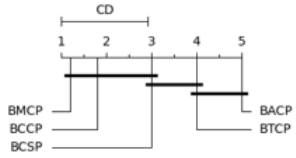
Inicialização: *Embedding*
Épocas:80
Métrica: NDCG@10
Friedman pvalue = 4,32842260712097E-08



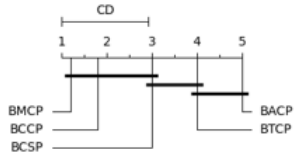
Inicialização: *Embedding*
Épocas:80
Métrica: Recall @20
Friedman pvalue = 7,95859865216327E-08



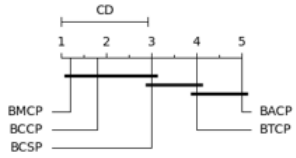
Inicialização: *Embedding*
Épocas:80
Métrica: NDCG@20
Friedman pvalue = 7,95859865216327E-08



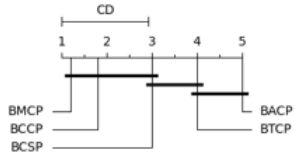
Inicialização: *Embedding*
Épocas:80
Métrica: Recall @30
Friedman pvalue = 7,95859865216327E-08



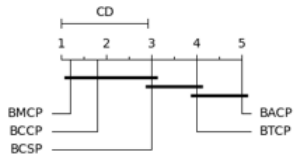
Inicialização: *Embedding*
Épocas:80
Métrica: NDCG@30
Friedman pvalue = 7,95859865216327E-08



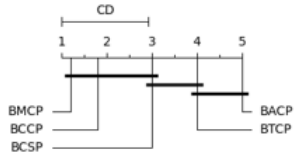
Inicialização: *Embedding*
Épocas:80
Métrica: Recall @40
Friedman pvalue = 7,95859865216327E-08



Inicialização: *Embedding*
Épocas:80
Métrica: NDCG@40
Friedman pvalue = 7,95859865216327E-08



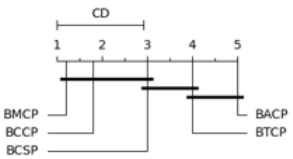
Inicialização: *Embedding*
Épocas:80
Métrica: Recall @50
Friedman pvalue = 7,95859865216327E-08



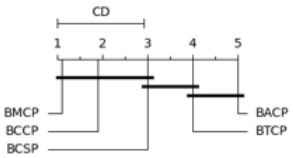
Inicialização: *Embedding*
Épocas:80
Métrica: NDCG@50
Friedman pvalue = 7,95859865216327E-08

Figura 73 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização *Embedding*, 80 Épocas

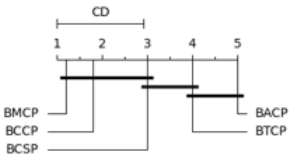
Fonte: Figura do autor



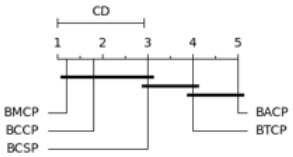
Inicialização: *Embedding*
Épocas:100
Métrica: Recall@10
Friedman pvalue = 7,95859865216327E-08



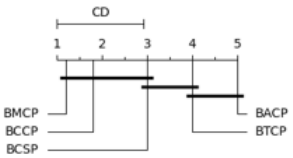
Inicialização: *Embedding*
Épocas:100
Métrica: NDCG@10
Friedman pvalue = 6,09770020529557E-08



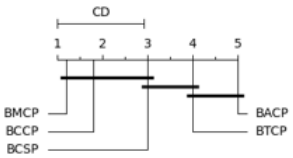
Inicialização: *Embedding*
Épocas:100
Métrica: Recall @20
Friedman pvalue = 7,95859865216327E-08



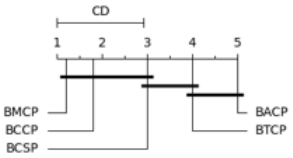
Inicialização: *Embedding*
Épocas:100
Métrica: NDCG@20
Friedman pvalue = 7,95859865216327E-08



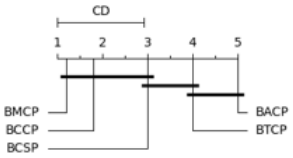
Inicialização: *Embedding*
Épocas:100
Métrica: Recall @30
Friedman pvalue = 7,95859865216327E-08



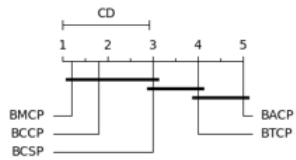
Inicialização: *Embedding*
Épocas:100
Métrica: NDCG@30
Friedman pvalue = 7,95859865216327E-08



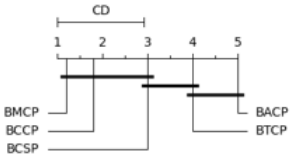
Inicialização: *Embedding*
Épocas:100
Métrica: Recall @40
Friedman pvalue = 7,95859865216327E-08



Inicialização: *Embedding*
Épocas:100
Métrica: NDCG@40
Friedman pvalue = 7,95859865216327E-08



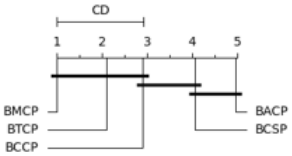
Inicialização: *Embedding*
Épocas:100
Métrica: Recall @50
Friedman pvalue = 7,95859865216327E-08



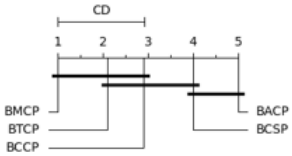
Inicialização: *Embedding*
Épocas:100
Métrica: NDCG@50
Friedman pvalue = 7,95859865216327E-08

Figura 74 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização *Embedding*, 100 Épocas

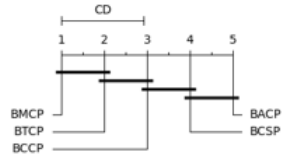
Fonte: Figura do autor



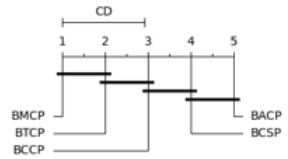
Inicialização: Média
Épocas:10
Métrica: Recall@10
Friedman pvalue = 6,65715985983287E-08



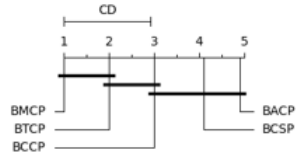
Inicialização: Média
Épocas:10
Métrica: NDCG@10
Friedman pvalue = 6,09770020529557E-08



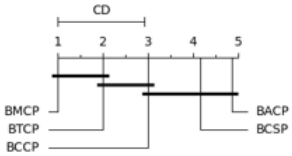
Inicialização: Média
Épocas:10
Métrica: Recall @20
Friedman pvalue = 4,32842260712097E-08



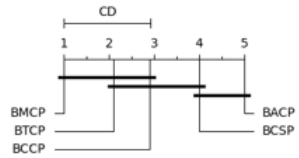
Inicialização: Média
Épocas:10
Métrica: NDCG@20
Friedman pvalue = 4,32842260712097E-08



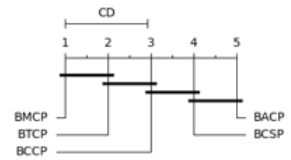
Inicialização: Média
Épocas:10
Métrica: Recall @30
Friedman pvalue = 5,04851211158285E-08



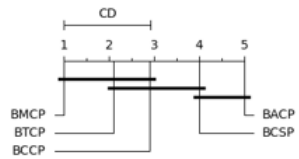
Inicialização: Média
Épocas:10
Métrica: NDCG@30
Friedman pvalue = 6,40735467330834E-08



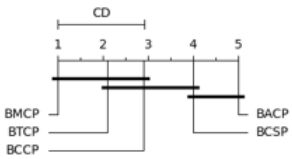
Inicialização: Média
Épocas:10
Métrica: Recall @40
Friedman pvalue = 6,09770020529557E-08



Inicialização: Média
Épocas:10
Métrica: NDCG@40
Friedman pvalue = 4,32842260712097E-08



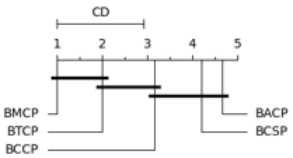
Inicialização: Média
Épocas:10
Métrica: Recall @50
Friedman pvalue = 6,09770020529557E-08



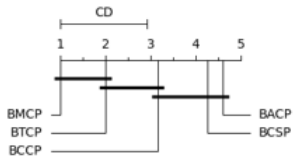
Inicialização: Média
Épocas:10
Métrica: NDCG@50
Friedman pvalue = 6,09770020529557E-08

Figura 75 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização Média, 10 Épocas

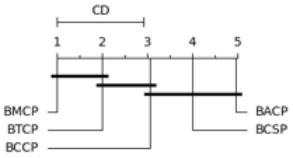
Fonte: Figura do autor



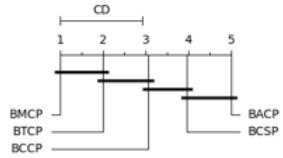
Inicialização: Média
Épocas:20
Métrica: Recall@10
Friedman pvalue = 9,85268852879187E-08



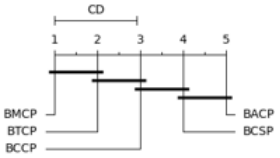
Inicialização: Média
Épocas:20
Métrica: NDCG@10
Friedman pvalue = 1,17131344323597E-07



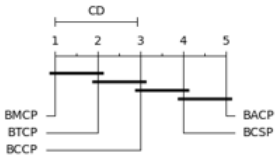
Inicialização: Média
Épocas:20
Métrica: Recall @20
Friedman pvalue = 5,19627149140128E-08



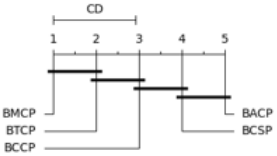
Inicialização: Média
Épocas:20
Métrica: NDCG@20
Friedman pvalue = 4,71778254546477E-08



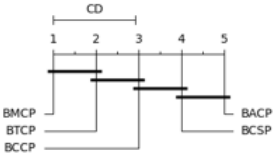
Inicialização: Média
Épocas:20
Métrica: Recall @30
Friedman pvalue = 4,32842260712097E-08



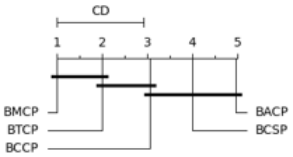
Inicialização: Média
Épocas:20
Métrica: NDCG@30
Friedman pvalue = 4,32842260712097E-08



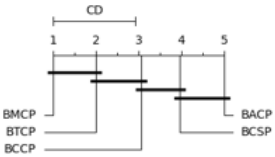
Inicialização: Média
Épocas:20
Métrica: Recall @40
Friedman pvalue = 4,32842260712097E-08



Inicialização: Média
Épocas:20
Métrica: NDCG@40
Friedman pvalue = 4,32842260712097E-08



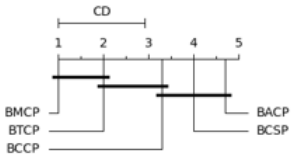
Inicialização: Média
Épocas:20
Métrica: Recall @50
Friedman pvalue = 5,19627149140128E-08



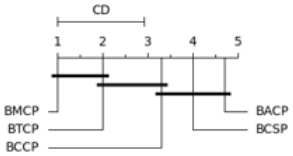
Inicialização: Média
Épocas:20
Métrica: NDCG@50
Friedman pvalue = 4,71778254546477E-08

Figura 76 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização Média, 20 Épocas

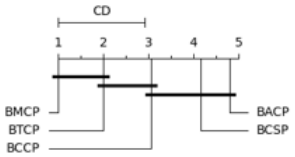
Fonte: Figura do autor



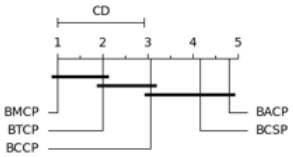
Inicialização: Média
Épocas:40
Métrica: Recall@10
Friedman pvalue = 1,0128316257924E-07



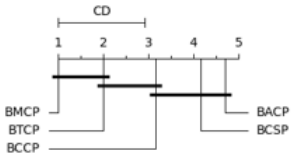
Inicialização: Média
Épocas:40
Métrica: NDCG@10
Friedman pvalue = 1,0128316257924E-07



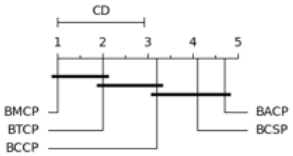
Inicialização: Média
Épocas:40
Métrica: Recall @20
Friedman pvalue = 7,50778430941207E-08



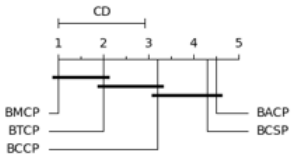
Inicialização: Média
Épocas:40
Métrica: NDCG@20
Friedman pvalue = 9,0387912647645E-08



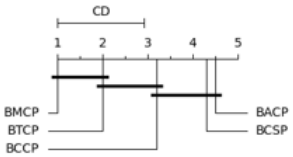
Inicialização: Média
Épocas:40
Métrica: Recall @30
Friedman pvalue = 1,07750754368741E-07



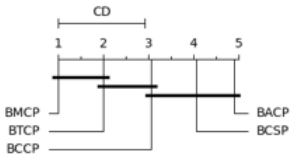
Inicialização: Média
Épocas:40
Métrica: NDCG@30
Friedman pvalue = 1,55778904972066E-07



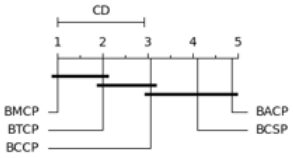
Inicialização: Média
Épocas:40
Métrica: Recall @40
Friedman pvalue = 2,53080080640289E-07



Inicialização: Média
Épocas:40
Métrica: NDCG@40
Friedman pvalue = 3,00544952091183E-07



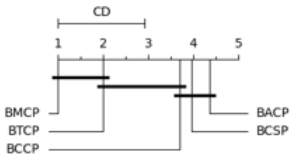
Inicialização: Média
Épocas:40
Métrica: Recall @50
Friedman pvalue = 5,61912513388683E-08



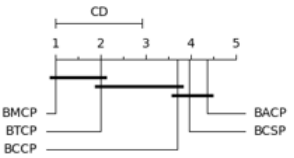
Inicialização: Média
Épocas:40
Métrica: NDCG@50
Friedman pvalue = 7,20469656441725E-08

Figura 77 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização Média, 40 Épocas

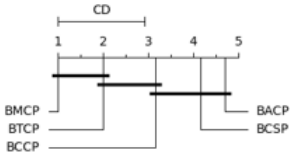
Fonte: Figura do autor



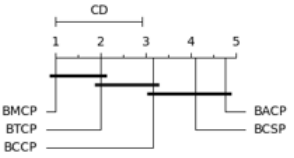
Inicialização: Média
Épocas:80
Métrica: Recall@10
Friedman pvalue = 3,62471505456428E-07



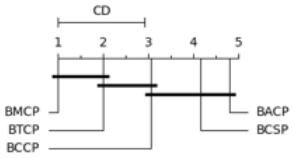
Inicialização: Média
Épocas:80
Métrica: NDCG@10
Friedman pvalue = 3,62471505456428E-07



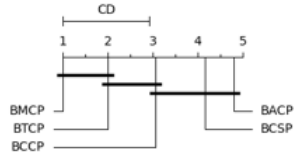
Inicialização: Média
Épocas:80
Métrica: Recall @20
Friedman pvalue = 9,81083230849965E-08



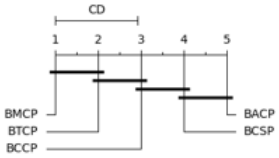
Inicialização: Média
Épocas:80
Métrica: NDCG@20
Friedman pvalue = 9,58000546493867E-08



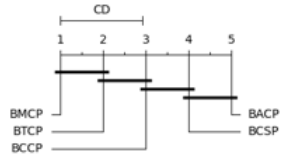
Inicialização: Média
Épocas:80
Métrica: Recall @30
Friedman pvalue = 9,0387912647645E-08



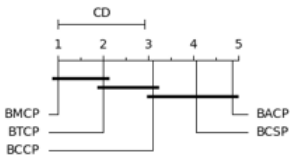
Inicialização: Média
Épocas:80
Métrica: NDCG@30
Friedman pvalue = 9,0387912647645E-08



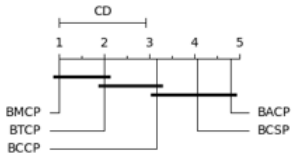
Inicialização: Média
Épocas:80
Métrica: Recall @40
Friedman pvalue = 4,32842260712097E-08



Inicialização: Média
Épocas:80
Métrica: NDCG@40
Friedman pvalue = 4,32842260712097E-08



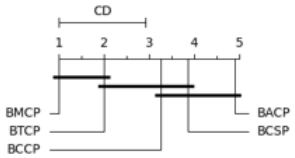
Inicialização: Média
Épocas:80
Métrica: Recall @50
Friedman pvalue = 6,58638038609014E-08



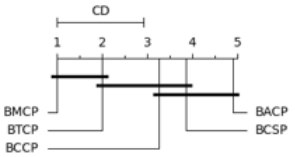
Inicialização: Média
Épocas:80
Métrica: NDCG@50
Friedman pvalue = 1,32428733782957E-07

Figura 78 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização Média, 80 Épocas

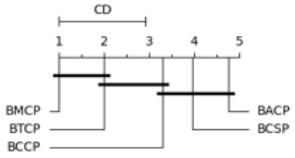
Fonte: Figura do autor



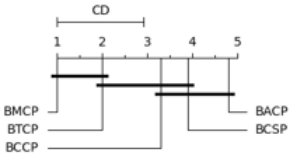
Inicialização: Média
Épocas:100
Métrica: Recall@10
Friedman pvalue = 7,15786645044178E-08



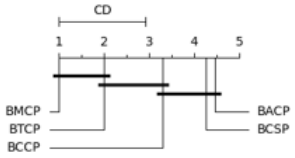
Inicialização: Média
Épocas:100
Métrica: NDCG@10
Friedman pvalue = 7,15786645044178E-08



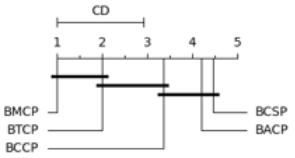
Inicialização: Média
Épocas:100
Métrica: Recall @20
Friedman pvalue = 9,58845207693087E-08



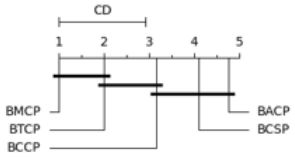
Inicialização: Média
Épocas:100
Métrica: NDCG@20
Friedman pvalue = 8,90257298845065E-08



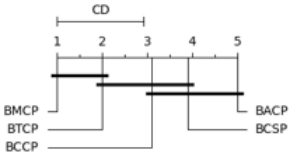
Inicialização: Média
Épocas:100
Métrica: Recall @30
Friedman pvalue = 3,00837086795888E-07



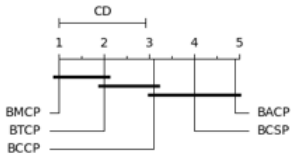
Inicialização: Média
Épocas:100
Métrica: NDCG@30
Friedman pvalue = 5,02431443312627E-07



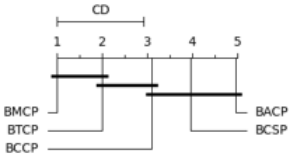
Inicialização: Média
Épocas:100
Métrica: Recall @40
Friedman pvalue = 9,58000546493867E-08



Inicialização: Média
Épocas:100
Métrica: NDCG@40
Friedman pvalue = 5,04851211158285E-08



Inicialização: Média
Épocas:100
Métrica: Recall @50
Friedman pvalue = 6,14047808809106E-08



Inicialização: Média
Épocas:100
Métrica: NDCG@50
Friedman pvalue = 5,61912513388683E-08

Figura 79 – Comparação entre os Modelos de Recomendação utilizando: Método de Inicialização Média, 100 Épocas

Fonte: Figura do autor