UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

TAGLINE TREICHEL

Benchmarking e Avaliação de Usuários de um Sistema de Recomendação Baseado na Similaridade entre Itens

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Dipl.-Ing. Vladimir Rybalkin
Coorientador: Prof. Dr. Leandro Krug Wives

Porto Alegre
2016

AGRADECIMENTOS

Gostaria de agradecer imensamente a todos os **professores e funcionários do INF** - Instituto de Informática da UFRGS - em especial aos que tive a oportunidade de trocas e de aprendizados valorosos. É uma satisfação enorme ter feito parte da família UFRGS e deste Instituto altamente qualificado com destaques nacionais e internacionais. Minha gratidão também ao **Prof. Dr. Leandro Krug Wives** por prontamente ter aceitado ser meu orientador neste trabalho, por toda a sua agilidade e presteza.

Também quero deixar aqui registrado a minha profunda gratidão e honra à **Prof. Dra. Taisy Silva Weber** que no ano de 2014 me incentivou a participar de um intercâmbio na Alemanha, uma experiência que é impossível descrever em palavras por todas as transformações, experiências e alegrias que me proporcionaram.

Agradeço também a parceria de todos os **colegas** de graduação do início ao fim do curso, em especial ao Edson, Lisandro, Otávio, Vinícius, Felipe, Gabriel, Marcos, Jozeanne e demais que colaboraram em trabalhos e estudos em grupo, vocês contribuíram para que esta jornada se tornasse mais descontraída e leve.

Minha gratidão também ao meu sócio Gabriel Moser e a todos que fizeram e fazem parte da **equipe Zipernet** pelo importante apoio e suporte ao longo de todos esses anos. Sem o trabalho e comprometimento de cada um de vocês não teria sido possível em paralelo construir a nossa empresa e cursar esta graduação.

Por fim, agradeço profundamente aos **meus pais** que mesmo longe sempre se fizeram presentes e que me ensinaram sobre o que há de mais precioso, sobre VALORES.

GRATIDÃO a todos!!

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ALS	Alternative Least Squares
GT	Ground Truth
HDFS	Hadoop Distributed File System
LAP	Link Assessment Problem
PHP	Hypertext Preprocessor
PPV	Positive Predictive Values
SQL	Structured Query Language
SR	Sistema de Recomendação
UFRGS	Universidade Federal do Rio Grande do Sul

RESUMO

Com a crescente potencialização da Internet como ferramenta para conectar pessoas e negócios ao redor do mundo, surge uma necessidade enorme de explorar as tecnologias de Sistemas de Recomendação. Sistemas de Recomendação auxiliam, através de predições matemáticas, desde sugestões de livros, filmes, roupas, artigos, até interações entre proteínas e genes. São poderosas ferramentas que podem extrair valor adicional para empresas a partir de suas bases de dados. O objetivo deste trabalho foi validar, por meio de um estudo comparativo, o desempenho e a acurácia nas recomendações de filmes geradas em diferentes plataformas de recomendação que utilizam a abordagem de Filtragem Colaborativa baseada em similaridade entre itens, tomando como referência o algoritmo de recomendação aqui chamado de LAP - *Link Assessment Problem* - proposto por Katharina A. Zweig e desenvolvido na Technische Universität Kaiserslautern na Alemanha. Verificou-se, nas diferentes plataformas selecionadas, os seguintes parâmetros: memória consumida, tempo de geração da recomendação e qualidade da recomendação. Para medir o parâmetro de qualidade utilizou-se como métrica estatística o PPV - *Positive Predictive Values*. Por fim, para o algoritmo baseado no *Link Assessment Problem*, foram avaliadas e medidas as recomendações em um experimento com usuários. Ao final foi possível observar que a qualidade das recomendações geradas pelo LAP produziu valores de PPV bem próximos dos valores obtidos nas demais plataformas, no entanto o algoritmo ainda precisa de evoluções e melhorias em sua performance.

Palavras-chave: Sistemas de Recomendação. Link Assessment Problem. Benchmarking. Similaridade entre itens.

Benchmarking and User Evaluation of a Recommendation System Based on Similarity Between Items

ABSTRACT

With the large increasing of the Internet as a tool to connect people and businesses around the world, emerges a demand to explore the recommendation systems technologies. Recommendation systems help, through mathematically predictions, from suggestions of books, movies, clothes, articles, to interactions between proteins and genes. They are powerful tools that can extract additional value for companies from their databases. The aim of this study was to evaluate, through a comparative study, the performance and accuracy in movie recommendations generated in different recommendation platforms which uses the Collaborative Filtering approach based on similarity between items, with reference to the recommendation algorithm called LAP - Link Assessment Problem - proposed by Katharina A. Zweig and developed at the Technische Universität Kaiserslautern in Germany. It was computed in the different selected platforms the following parameters: memory consumed, time to generate the recommendation and quality of these recommendations. To measure the quality parameter was used a statistical metric PPV - Positive Predictive Values. Finally, for the algorithm based on the Link Assessment Problem, the recommendations were evaluated and measured in an experiment with users. At the end it was possible to observe that the quality of the recommendations generated by the LAP was competitive compared to the other platforms, however the algorithm still needs improvements in its performance.

Keywords: Recommendation System, Link Assessment Problem, Benchmarking and User Evaluation, User-based Similarity.

1 INTRODUÇÃO

Sistemas de Recomendação (SR), em geral, são mais popularmente usados para fazer sugestões de filmes e livros. Dois grandes cases de sucesso, nesse contexto, são o Netflix[1] e a Amazon (LINDEN; SMITH; YORK, 2003), ambas as plataformas melhoraram consideravelmente as recomendações de seus produtos através do uso de algoritmos especializados de recomendação. No entanto, os SR não se limitam a recomendações de filmes e livros, podendo ser usados para distintos domínios como: E-commerces, Otimização de Conteúdo, Marketing Online, Buscas na Web, Agências de Viagens, dentre tantos outros domínios.

Algoritmos de SR emergem como uma importante ferramenta especialmente em plataformas de e-commerce, pois melhoram as recomendações e consequentemente a satisfação dos clientes, pois, através de predições matemáticas, é possível, a partir de preferências de outros usuários/clientes, gerar recomendações/dicas/sugestões de itens que podem suprir uma necessidade ou vontade dos clientes. Tomando como exemplo algumas plataformas como Amazon, já citada, Mercado Livre e Submarino, que reúnem inúmeros vendedores e milhares (até milhões) de itens/produtos, portanto oferecer produtos aos clientes com mais precisão não só é uma necessidade para melhorar o relacionamento com o cliente, mas também é uma estratégia para otimizar e gerar ainda mais vendas.

Nesse contexto e dada a relevância de SRs, Katharina A. Zweig propôs em (ZWEIG, 2010) e (SPITZ et al., 2016) uma nova abordagem de Filtragem Colaborativa para encontrar link ocultos em grafos bipartidos e então explorar similaridades entre vizinhanças. Segundo Katharina, muitos dos grandes conjuntos de dados são ruidosos e contêm links que representam relações de baixa intensidade e que são difíceis de diferenciar das interações aleatórias. Isso é especialmente relevante em dados ecológicos em larga escala e também para os dados da Web 2.0 que contém interações humanas. Nesses tipos de redes, chamadas de redes complexas, há *links* faltantes entre as entidades e também *links* falsos e portanto o propósito do LAP é refinar os dados, ou seja refinar esses links, com base no princípio da semelhança estrutural, que avalia a vizinhança compartilhada de dois nós.

O algoritmo acima descrito será referenciado neste trabalho como algoritmo de *Link Assessment Problem* (LAP). A abordagem utilizada para o desenvolvimento do LAP tem a vantagem de não necessitar de informações sobre as entidades (itens ou pessoas), é

[1] http://www.netflixprize.com/

necessário apenas que o banco de dados tenha a representação das relações entre pessoas e itens.

Neste trabalho limitou-se ao domínio de filmes, pois, para poder medir a acurácia e efetuar as validações do método, é necessário ter posse de um arquivo chamado *ground truth*, que é uma seleção gerada por humanos que contém os filmes que verdadeiramente são similares. Em outros domínios, obter esse arquivo pode não ser possível. As bases de dados utilizadas neste estudo foram provenientes do Netflix.

> The Netflix competition asked to build a recommendation system that beats the company's own system. A first approach is to find which films are similar to each other, by using the information of how often two films were rented (and rated) by the same user. The idea behind this approach is that if many users view both films, these films might be similar to each other (ZWEIG, 2010, p.1).

O propósito principal então deste trabalho foi verificar a acurácia e o desempenho da execução do algoritmo LAP comparativamente a outras plataformas (outros algoritmos) de recomendação que usam também a abordagem de Filtragem Colaborativa. Para tal, fez-se um estudo comparativo entre essas distintas plataformas. Além disso, para o algoritmo LAP foi efetuada uma pesquisa com usuários onde solicitou-se que pelos menos 5 pares de filmes fossem avaliados como similares ou não. Ao final será apresentada uma análise de todos os resultados obtidos.

1.1 Motivação

Conforme já citado anteriormente, dada a importância dos SRs especialmente nos negócios empresarias, essa temática tem um amplo e promissor universo a ser explorado. Portanto, neste estudo, desejou-se ter um conhecimento adicional em como as plataformas de recomendação estão se posicionando no mercado, qual a demanda de tempo e recursos de hardware necessários para se obter uma recomendação, qual a acurácia das predição e, por fim, como o algoritmo LAP, que em (SPITZ et al., 2016) apresentou acurácia superior (i.e. maior valor do PPV), estava posicionado perante demais plataformas.

1.2 Objetivo

O objetivo deste trabalho, de validação das plataformas de recomendação por meio de um processo de *benchmarking* e também por um experimento com usuários,

foi validar a qualidade do algoritmo de recomendação LAP. Apesar de outros trabalhos e amplas pesquisas relacionados a este algoritmo terem apresentado bons resultados, a ideia central era analisá-lo comparativamente com plataformas mais robustas e existentes no mercado e que têm propósito de recomendação similar.

1.3 Organização do texto

Esta monografia está organizada como segue:

No Capítulo 2 é apresentado o Referencial Teórico, dividido em duas Seções: História e Abordagens de SR, uma descrição e apresentação das diferentes categorias de Sistemas de Recomendação para diferentes propósitos e domínios, e *Link Assessment Problem*, onde dedica-se exclusivamente para explicar a fundamentação teórica do algoritmo LAP proposto por A. K. Zweig e que é tido como o sistema de recomendação motivador para este estudo.

No Capítulo 3 estão os Trabalhos Relacionados, onde é apresentado uma descrição sobre cada plataforma de recomendação estudada, o critério de seleção destas plataformas, bem como demais plataformas inicialmente analisadas para o *benchmarking*.

O processo de *benchmarking* e a metodologia de avaliação com usuários serão apresentados no Capítulo 4 como minhas Contribuições.

No Capítulo 5 apresenta-se a análise crítica sobre as plataformas estudadas, bem como os resultados obtidos na pesquisa efetuada com usuários. O texto é encerrado no Capítulo 6 com as Conclusões.

2 REFERENCIAL TEÓRICO

Este Capítulo está dividido em duas seções. A primeira discursa sobre o começo dos Sistemas de Recomendação, suas abordagens e categorias existentes. Na segunda Seção é apresentada a fundamentação teórica para o Link Assessment Problem.

2.1 SISTEMAS DE RECOMENDAÇÃO : História e Abordagens

De acordo com Ricci, Rokach e Shapira (2011), "Um Sistema de Recomendação combina várias técnicas computacionais para selecionar itens personalizados com base nos interesses dos usuários e conforme o contexto no qual estão inseridos" (p.1).

Para entender brevemente a história dos SRs, esses sistemas surgiram como resposta à dificuldade das pessoas em escolher dada uma grande variedade de produtos e serviços e entre as várias alternativas. Foi então em meados da década de 90 que surgem as primeiras pesquisas na área:

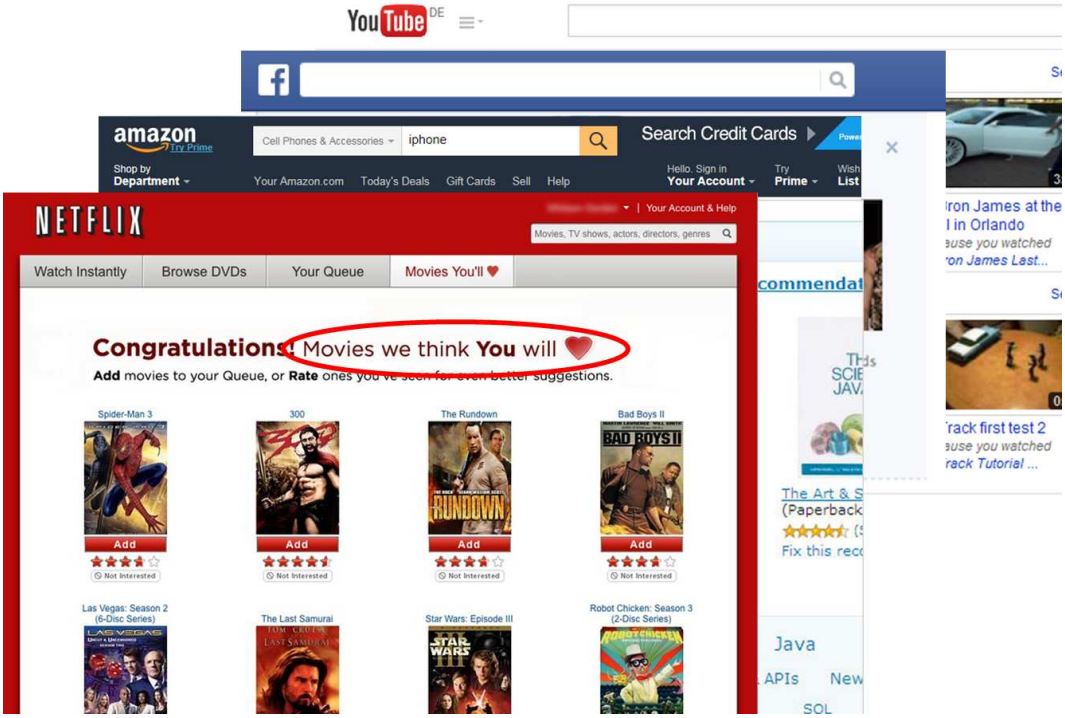
- **1992:** Tapestry (Xerox Palo Alto) - primeiro sistema usando filtragem colaborativa;
- **1994:** GroupLens - primeiro sistema a usar informações de *ratings*;
- **1997:** MovieLens - primeiro sistema de recomendação para filmes.

Foi no começo dos anos 90 que pesquisadores da Xerox trouxeram o conceito de "Filtragem Colaborativa", que designava um tipo de sistema em que era efetuada a filtragem da informação, mas por meio do apoio de humanos e portanto era um processamento manual. Foi nesse momento que iniciou a resolução de problemas de recomendação usando a informação dos *ratings* - estruturas de avaliação. Desde então esse tema tem sido largamente estudado. A automatização da chamada Filtragem Colaborativa se deu da metade dos anos 90 ao começo dos anos 2000, quando então sistemas desse tipo passaram a ser comercializados.

Atualmente todos os grandes *players* de e-commerce, sites de busca e redes sociais fazem amplo uso dos motores de recomendação.

Os sistemas de recomendação utilizam diversos tipos de tecnologias. Autores costumam classificá-los em dois grandes grupos: as chamadas Abordagens Tradicionais e as Abordagens Modernas. Das abordagens modernas pode-se citar: *Context-aware*

Figura 2.1: Sites populares que fazem uso de SR



Fonte: imagens obtidas diretamente no website de cada empresa

Approaches; Semantic based Approaches; Cross-domain based Approaches; Peer-to-Peer Approaches; Cross-ligual Approaches.

As Abordagens Tradicionais, por sua importância, serão apresentadas nas subseções a seguir. Basicamente, o que define qual das abordagens utilizar, será o contexto e o tipo de dados que se tem.

2.1.1 Filtragem Colaborativa

Sistemas de Filtragem Colaborativa fazem a recomendação baseada na **medida de similaridade entre usuários e/ou itens**. Itens são recomendados aos usuários baseado na preferência de usuários semelhantes. A ideia central é: pessoas com gostos similares a um determinado usuário possivelmente gostarão dos mesmos itens que este usuário gosta.

A tarefa comum de SRs de filtragem colaborativa é melhorar a experiência do usuário através de recomendações personalizadas com base no feedback implícito prévio. Esses sistemas analisam diferentes tipos de comportamento do usuário, como histórico de compras, histórico de navegação, filmes/séries assistidos, a fim de modelar as preferências do usuário.

Pesquisadores costumam dividir esse tipo de filtragem em 2 categorias princi-

pais: *User-based* (ou *Memory-based* - baseada em Memória) e *Item-based* (ou *Model-based* - baseada em Modelo).

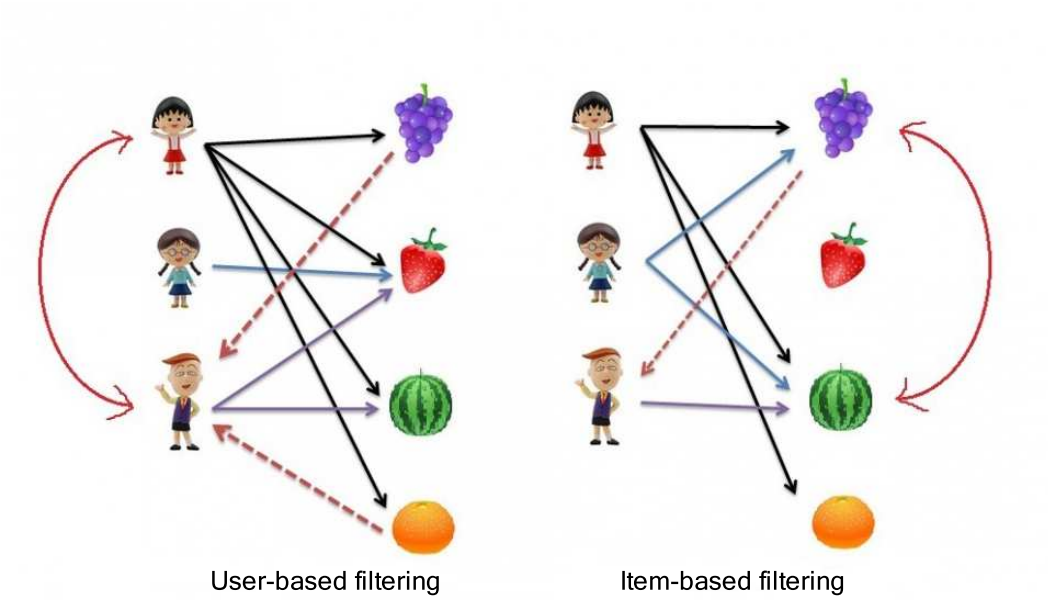
User-based : estes algoritmos utilizam toda a base de dados que relaciona usuários e itens para gerar uma predição. Estes sistemas empregam técnicas estatísticas para encontrar um conjunto de usuários, conhecidos como **vizinhos**, que possuem um histórico semelhante com o usuário de destino (ou seja, ambos avaliaram itens diferentes de forma semelhante ou eles tendem a comprar um conjunto semelhante de itens). Uma vez que uma vizinhança de usuários é formada, estes sistemas usam diferentes algoritmos para combinar as preferências dos vizinhos para então produzir uma previsão ou top-N recomendação para o usuário ativo. As técnicas, também conhecidas como "vizinho mais próximo"ou "filtragem colaborativa baseada no usuário", são mais populares e amplamente utilizadas na prática. Os problemas comuns de algoritmos baseados em memória são a Esparsividade e a Escalabilidade. Esparsividade é quando o número de itens existentes excede muito a quantidade que cada pessoa é capaz de ver/avaliar/explorar. Por exemplo no site da Amazon, existe milhões de produtos disponíveis, mas é provável que apenas na casa de milhares foram avaliados ou ainda poucos produtos tem muitas avaliações. E a escalabilidade se refere a capacidade de processar um grande volume de dados, ou seja, a medida que a bases de usuários e itens aumentam, os algoritmos de recomendação precisam continuar sendo capazes de processar uma recomendação dentro de um tempo aceitável.

Item-based : estes algoritmos geram a recomendação de um item a partir da criação de um modelo de classificação dos usuários. Algoritmos nesta categoria usam uma abordagem probabilística e o processo de filtragem é calculado em cima de valor esperado de uma predição de usuário segundo avaliações sobre outros itens. O processo de construção do modelo é realizado por diferentes algoritmos de aprendizado de máquina, como Redes Bayesianas, Clustering, abordagens Baseadas em Regras, Regressão, etc.

Os dois principais problemas encontrados neste tipo de filtragem são: o *cold-start problem* e o viés de popularidade.

O *cold-start problem* ocorre quando não se tem muitas informações de um novo usuário na base e portanto é difícil fazer recomendações precisas a ele. Já o viés de popularidade é quando o sistema tem usuários que possuem gostos únicos e é muito difícil recomendar itens a eles e portanto o sistema tende a recomendar itens populares.

Figura 2.2: Filtragem Colaborativa



Fonte: http://www.salemmarafi.com/

2.1.2 Filtragem baseada em Conteúdo

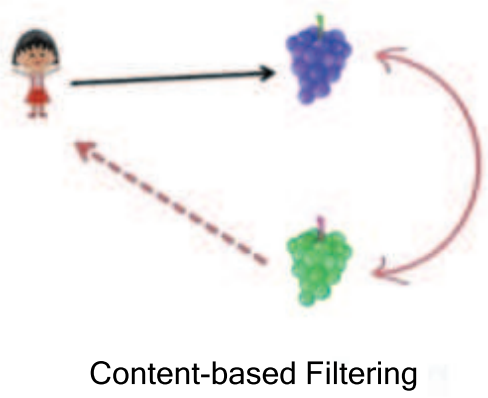
Sistemas de Filtragem baseada em Conteúdo, ou também chamados de Filtragem de Informação, examinam as **propriedades dos itens** recomendados. A similaridade dos itens é medida com base na similaridade de suas propriedades. Por exemplo, se um usuário do Netflix assistiu muitos filmes do comediante Ben Stiller, então uma possível recomendação para esse usuário seria a de filmes classificados como sendo do gênero "comédia".

Dentre os problemas enfrentados nesse contexto, podem ser citados dois críticos: os algoritmos, em geral, são complexos, podendo exigir uma alta carga de desenvolvimento e a necessidade o usuário abastecer o sistema com seus gostos pessoais, permitindo assim que o sistema tenha um maior conhecimento do perfil do usuário.

2.1.3 Filtragem Híbrida

A Filtragem Híbrida, como o próprio nome sugere, é a combinação das duas abordagens anteriores. Na prática é bastante comum fazer o uso desta filtragem, pois combinando a filtragem Colaborativa com a Baseada em Conteúdo é possível minimizar

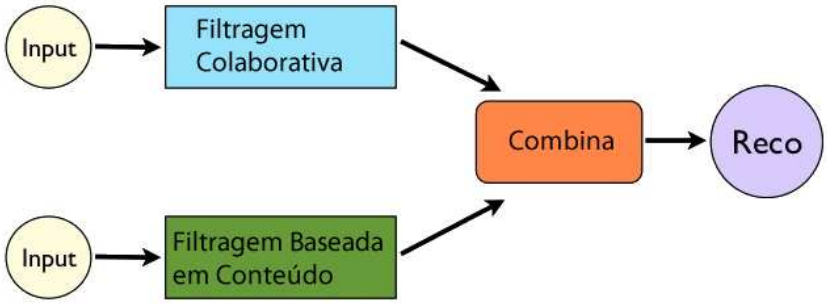
Figura 2.3: Filtragem baseada em Conteúdo



Fonte: adaptado de http://www.salemmarafi.com/

os problemas que isoladamente não podem ser resolvidos.

Figura 2.4: Abordagem Híbrida de Recomendação



Fonte: http://www.dataconomy.com/an-introduction-to-recommendation-engines

Recomendações do tipo híbrida são bastante utilizadas pois, ao combinar de várias maneiras os diferentes sistemas de filtragem, eliminam mutuamente as desvantagens. Dentre os principais benefícios, citam-se:

- Gerar predições com mais acurácia;
- Reduzir o cold-start problem.

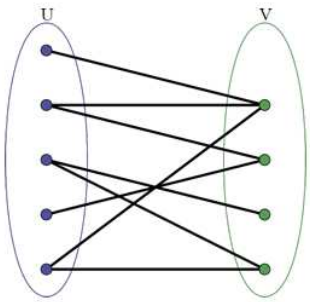
2.2 THE LINK ASSESSMENT PROBLEM

Nesta Seção é apresentado o problema do Link Assessment Problem e conceitos teóricos relacionados.

2.2.1 One-Mode Projection

Em (ZWEIG, 2010), Katharina A. Zweig propôs uma nova abordagem para encontrar links ocultos em grafos bipartidos. Este problema é chamado de *Link Assessment Problem*. Grafo bipartido é um grafo cujos vértices podem ser divididos em dois conjuntos disjuntos U e V de interesse, tais que toda aresta conecta um vértice em U a um vértice em V. Neste trabalho U e V representam **usuários** e **filmes** e as arestas indicam os filmes que foram assistidos (ou avaliados) por cada usuário.

Figura 2.5: Grafo Bipartido



Fonte: http://en.wikipedia.org/wiki/Bipartite_graph

A ideia proposta diz então que, a partir de um grafo bipartido, observa-se o lado de interesse (U ou V) e este será o *One-Mode Projection*. Portanto a saída do One-Mode Projetion será um grafo simples que contém a relação de filmes contendo os pesos de cada aresta. O peso das arestas é chamado de coocorrência (cooc) e representa a quantidade de usuários que assistiram o par de filmes.

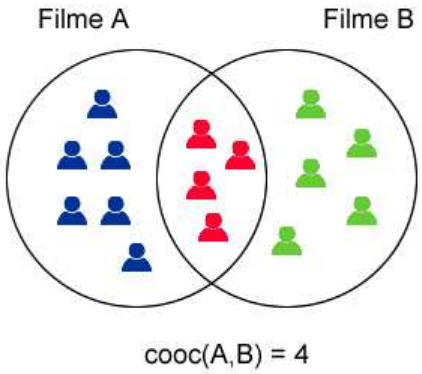
> The most common way to deal with such a bipartite graph of which one side is of main interest, is to turn it into a general graph by a so-called 'one-mode projection'. For this, two objects from the same side are connected if they share neighbors on the other side. The easiest approach is to connect them if they share at least one neighbor. Concerning the Netflix data set, this implies that if a user has rented k films in total, this will cause all k films to be connected to each other. (ZWEIG, 2010, p.1)

Concluindo, tendo a posse então de um grafo One-Mode Projection é possível gerar um *cluster*, um aglomerado de informações que compartilham uma mesma característica, usando diferentes técnicas para obter uma recomendação.

2.2.2 Medida de Similaridade

A significância da *cooc* é assumida intuitivamente como um parâmetro para medir a intensidade entre dois filmes. Em (ZWEIG, 2010) é levantada então a questão: considerando o grau de distribuição de um par de filmes no One-Mode Projection e o seu valor de coocorrência, poderiam esses dados fornecer uma possível medida de similaridade? Caso sim, esta técnica poderia ser aplicada a qualquer domínio e não somente à filmes, uma vez que um grafo bipartido representa a relação, que tenha ocorrido mais de uma vez, entre dois conjuntos, por exemplo clientes que compram produtos em um e-commerce.

Figura 2.6: Medida de similaridade: coocorrência



Na Figura 2.6 é exemplificado o cálculo da *cooc*. A imagem ilustra que o filme A foi assistido por 6 usuários, o filme B por 5 usuários e ambos (A e B) foram assistidos por 4 usuários. Portanto temos que cooc=4, o que significa que o grau de intensidade do par de filmes (A,B) é igual 4.

Katharina A. Zweig em posterior trabalho propõe uma medida estatística para o Link Assessment Problem chamada z-score - z* - que usa como entrada o valor de cooc. Z* juntamente com p-value são então considerados como coeficientes para medir a similaridade entre dois elementos (dois filmes, neste caso):

> [..]we then propose a new methodology for link assessment called z* that assesses the statistical significance of the number of their common neighbors by comparison with the expected value in a suitably chosen random graph model and which is a consistently top-performing algorithm for all benchmarks (SPITZ et al., 2016, p.1).

De maneira simplificada, o algoritmo aplicado ao One-Mode Projection trata-se de um processo de amostragem usando diferentes amostras do próprio conjunto de filmes,

que ao final do processamento irá resultar nos valores de z-score e p-value. Conhecendo esses valores podemos estão predizer a semelhança entre um par de filmes. No Capítulo 3 será apresentando este algoritmo em mais detalhes.

3 TRABALHOS RELACIONADOS

Conforme já mencionado no Capítulo anterior, a área de Sistemas de Recomendação iniciou, de fato, nos começo do anos 90 e desde então as pesquisas e soluções propostas têm evoluído bastante, mas o que se pode perceber é que, em geral, as ferramentas são bastante especializadas e são modeladas para domínios mais específicos.

Para a execução do processo de *benchmarking*, que será descrito em detalhes no próximo Capítulo, efetuou-se primeiramente uma seleção aleatória de uma série de plataformas de recomendação. Num segundo momento foi efetuado o estudo individualizado de cada plataforma, identificando o tipo de filtragem de recomendação utilizado, características técnicas da ferramenta e ambiente de execução necessário para poder efetuar os testes.

Na tabela 3 abaixo estão elencadas as plataformas pesquisadas juntamente com informações técnicas de cada uma.

Tabela 3.1: Plataformas de Recomendação Analisadas

	Tipo de Filtragem	**Código Fonte**	**Linguagem**
Easyrec (API)	Não identificado	Código Aberto	Java
Duine (descontinuada)	Híbrida	Código Aberto	Java
Myrrix(descontinuada)	Colaborativa	Código Aberto	Java
***Oryx**	Colaborativa	Código Aberto	Java
MyMediaLite	Colaborativa	Código Aberto	C#
***GraphLab Create**	Colaborativa	Licença de Uso	Python
Crab	Colaborativa	Código Aberto	Python

Da lista acima selecionou-se então apenas aquelas que foram desenvolvidas em cima do **mesmo tipo de filtragem**, no caso Filtragem Colaborativa, e que aceitassem como entrada um banco de dados implícito. No próximo Capítulo enfatiza-se o porquê dessa escolha. Assim, as plataformas selecionadas foram **Oryx** e **GraphLab Create**.

O que foi possível perceber, em boa parte das plataformas estudadas, é que elas foram criadas partindo de pesquisas acadêmicas. Algumas com maior êxito foram expandidas e tornaram-se plataformas com propósitos comerciais.

Nas seções a seguir são apresentados os detalhes técnicos de cada uma das três plataformas selecionadas para o processo de *benchmarking*: Oryx, GraphLab Create e LAP, sendo esta última o algoritmo o que foi tomado como motivação para este estudo comparativo.

3.1 ORYX

Oryx é um projeto *open-source*, desenvolvido pela empresa chamada Cloudera, que fornece uma infra-estrutura com algoritmos para aprendizado de máquina e análise preditiva (recomendação), em tempo real e para larga escala.

É um framework que descente da plataforma de recomendação chamada Mahout. Trata-se de uma plataforma robusta que utiliza técnicas para SR e também algoritmos de classificação e *clustering*, para aprendizado de máquina.

Características da plataforma:

- Implementa algumas classes de algoritmos comumente usados em aplicações comerciais, como: filtragem colaborativa, classificação/regressão e clustering;
- Também serve consultas desses modelos em tempo real através de uma HTTP REST API e também pode atualizar modelos em resposta a streaming de novos dados;
- Para escalabilidade utiliza Apache Hadoop[1] e HDFS[2];
- Permite a execução de algoritmos em cima de uma estrutura com MapReduce[3].

3.1.1 Arquitetura

A arquitetura da plataforma Oryx está dividida em 2 camadas: Computation Layer (Camada de Computação) e Serving Layer (Camada de Serviço).

[1]É um framework *open-source* usada para processamento e armazenamento de dados distribuídos.

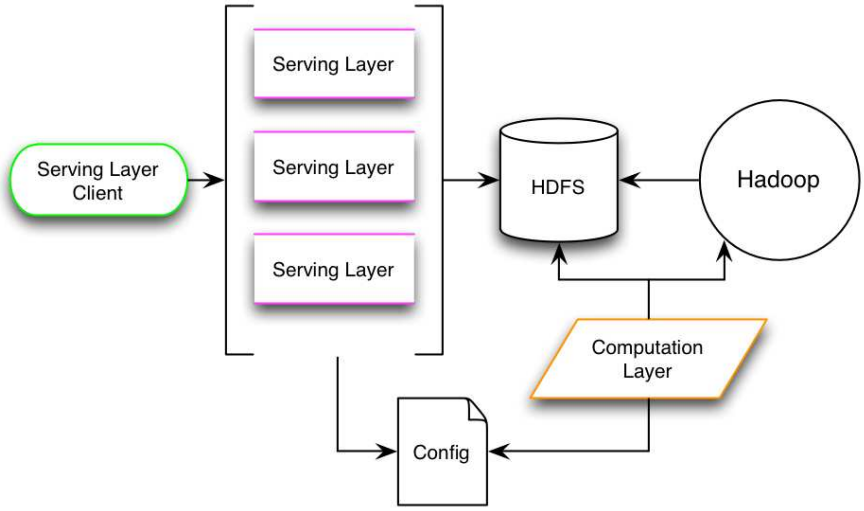
[2]HDFS - Hadoop Distributed File System - é um sistema de arquivos distribuído e escalável usado para armazenar grandes volumes de dados em múltiplas máquinas.

[3]É um modelo de programação utilizado para processar grandes volumes de dados distribuídos em um cluster.

A Camada de Computação é um processo em batch, offline, que constrói um modelo de aprendizagem de máquina a partir dos dados de entrada. Sua operação prossegue em "gerações" onde um modelo é construído instantaneamente a partir da entrada. A camada de Computação é um processo de servidor desenvolvido em Java.

A Camada de Serviço é um processo de servidor também desenvolvido em Java. Essa camada pode ser acessada a partir de um navegador ou qualquer linguagem ou ferramenta que faça solicitações HTTP.

Figura 3.1: Arquitetura da Plataforma Oryx



Fonte: github.com/cloudera/oryx

Os pré-requisitos para rodar as recomendações em uma máquina Linux são:

- Requer Java 7 ou superior;
- Requer Hadoop 2.6.0 ou superior - é possível usar uma configuração de Hadoop Sigle Cluster, i.e, para rodar em apenas 1 núcleo de processador.

3.1.2 Modelo utilizado para Recomendação

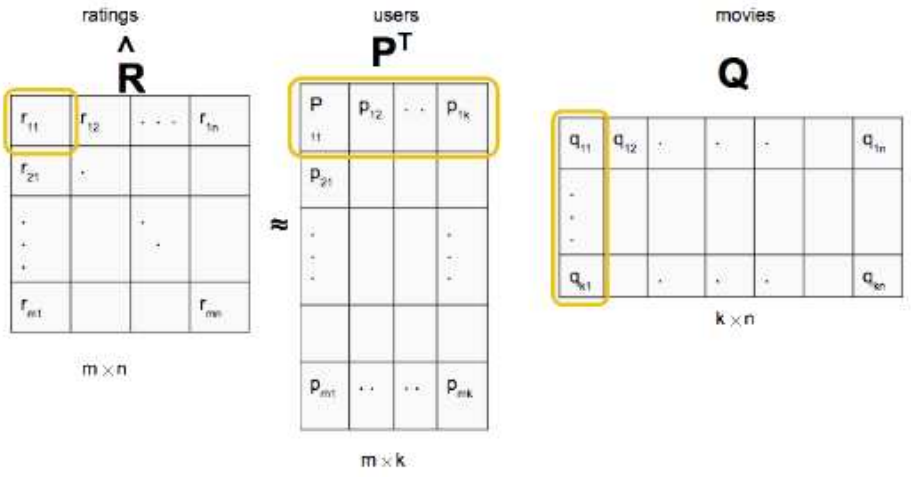
Oryx gera a recomendação utilizando um método baseado em **Fatoração de Matriz** - Matrix Factorization-based Approach - e usa a variante **ALS** - Alternative Least Square (Alternância de Mínimos Quadrados).

Para realizar a Fatoração de Matriz a ideia é calcular o produto de duas matrizes UV onde, por exemplo, U representa usuários e V os itens. O resultado deste produto gera então uma *matriz de classificação (ratings)*, que é a matriz de onde se obtém as possíveis predições. Para a Fatoração de Matriz é assumido que:

- Cada usuário pode ser "representado" por k atributos (ou vetores latentes)[4]. Um atributo pode ser, por exemplo, o gênero de um filme e o valor que este atributo assume seria o quanto o usuário gosta deste gênero;
- Cada item (filme) é também representado por k atributos. Exemplo: o atributo poderia ser um valor que representa o quanto semelhantes são os gêneros "romance" e "drama".

Segue exemplo:

Figura 3.2: Fatoração de Matrizes



Fonte http://online.cambridgecoding.com[5]

Então, ao se multiplicar a transposta da matriz P^T pela matriz Q, ou seja, calcular os produtos de cada k atributo de P por cada k atributo em Q, esta seria uma boa aproximação para se obter a *matriz de classificação* (ratings).

No entanto, a dificuldade é que em alguns casos os k atributos não são conhecidos ou mesmo não é possível definir quais e quantos são relevantes, e, nesta situação se faz necessário usar uma *função de minimização*, que neste caso é usada a **ALS**. Essa função de minimização auxiliará a "preencher" a matriz P e Q, pois, uma vez que os

[4]Os atributos são, na literatura, chamados de "vetores latentes".

atributos ou vetores latentes não são conhecidos, a ideia é usar o *truque* de alternância de mínimos quadrados para obter ambas as matrizes.

Então, para iniciar o processo, a matriz Q é inicializada com valores aleatórios. Uma vez tendo a matriz Q inicializada, estima-se os valores de atributos (também desconhecidos) da matriz P para cada usuário. E por fim, com os valores da matriz P usa-se ela para estimar os valores dos atributos de cada item na matriz matriz Q. Este é um processo iterativo e deve ser feito coluna a coluna de cada matriz para que ao final seja possível então obter uma boa aproximação da matriz completa de classificação.

Conforme descrito em (HU; KOREN; VOLINSKY, 2008), o processo de geração das matrizes P e Q necessita um número suficiente de iterações até que se atinja um ponto de convergência das duas matrizes e que não apresentem mais mudanças ou em que a mudança seja bem pequena. No entanto, há um pequeno problema nos valores de cada coluna destas matrizes, pois não se tem nem dados completos do usuário nem dados de completos de filmes. Então, para melhorar a estimativa dos atributos, pode ser interessante criar algum tipo de penalização dos filmes que não têm classificações nesta regra de atualização acima citada. Fazendo isso vamos depender somente dos filmes que têm avaliações dos usuários e não fazer qualquer suposição em torno dos filmes que não são classificados na recomendação. Para isso é criada uma matriz que é chamada pelos autores como matriz de peso w_{ui} tal que:

$$w_{ui} = \left\{ \begin{array}{c} \text{0, se o usuário } u \text{ não avaliou o filme } i \\ \\ \text{1, caso contrário} \end{array} \right\}$$

As funções de custo que são usadas para minimizar são as seguintes:

$$J(x_u) = (q_u - x_u Q) W_u (q_u - x_u Q)^T + \gamma x_u x_u^T$$

$$J(y_i) = (q_i - y_i P) W_i (q_i - y_i P)^T + \gamma y_i y_i^T$$

É importante salientar que as funções de minimização acima podem causar o chamado *overfitting* (sobreajuste) dos dados, ou seja, erros ou perturbações nos dados, então é necessário que se faça correções nessas perturbações para que parâmetros sejam ajustados. Essa correção pode ser efetuada utilizando, por exemplo, o RMSE - Root Mean Square Error.

Ao final, tem-se então:

$$x_u = (QW_uQ^T + \gamma I)^(- 1)QW_uq_u$$

$$y_i = (P^TW_iP + \gamma I)^(- 1)P^TW_iq_i$$

Onde W_u e W_i são matrizes diagonais e I é a matriz identidade. As funções acima então serão as função de predição.

3.2 GRAPHLAB CREATE

GraphLab Create é uma biblioteca desenvolvida em Python que faz parte de uma framework que provê soluções de Aprendizagem de Máquina. Esse framework, segundo informações de seu website *www.turi.com*, foi criado para desenvolvedores e cientistas de dados e é mantido por Turi Create Intelligence. Para fazer uso de seus produtos é necessário possuir uma licença de Uso. Para os testes e desenvolvimento deste trabalho foi solicitado uma licença de uso acadêmico o qual tem validade de 1 ano.

Trata-se de uma plataforma robusta que permite análise de dados em larga escala. Para gerenciar a escalabilidade é utilizado uma estrutura de dados chamada SFrame, que permite manipular dados na casa de terabytes de forma muito rápida. Interessante comentar que, durante a execução deste trabalho, vários testes formam performados em um Notebook de baixa capacidade e ainda assim com ótimos tempos de resposta.

No contexto de Recomendação, a plataforma GraphLab Create fornece diferentes pacotes que podem ser combinados de forma fácil e flexível, segundo informações obtidas no Website da Turi[6]. Os modelos de recomendação disponibilizados são: Fatoração de Matriz; Similaridade entre Itens - modelo baseado em Vizinhança e Modelo baseado em Popularidade - “Most-Viewed”.

3.2.1 Arquitetura

Como já mencionado anteriormente, a escalabilidade é possível pelo uso de uma estrutura de dados tabular, desenvolvida também em Python, chamada SFrame. Essa estrutura de dados permite trabalhar com bases de dados de larga escala e que sejam

[6]http://www.turi.com/

maiores do que a quantidade de memória existente na máquina.

A estrutura SFrame armazena os dados em colunas, de forma temporária, na camada de Servidor do GraphLab e posteriormente os dados são copiados para uma memória persistente, como disco rígido.

Os pré-requisitos para rodar as recomendações em uma máquina Linux:

- Distribuição do Linux com GLIBC >= 2.11;
- Arquitetura de processador de 64bit;
- Pelo menos 4 GB de memória RAM;
- Pelo menos 2 GB de espaço livre em disco;
- Requer ambiente instalado: Python 2.7.x e versão pip >= 7 e Anaconda 2 v4.0.0 (64 bits).

3.2.2 Modelo utilizado para Recomendação

As recomendações geradas nesta framework utilizam diferentes algoritmos e técnicas, conforme citado anteriormente. Logo, ao acessar a página *www.turi.com/learn/userguide/recommender/introduction.html*, onde é disponibilizado o Tutorial de Uso da plataforma de Recomendação, é apresentado um tópico *Choosing a Model* (Selecionando o Modelo) onde é possível entender e escolher o modelo de recomendação de acordo com o dados que o usuário possuir.

Para execução deste estudo, foi usada uma base de dados do Netflix do tipo implícita contendo apenas 2 colunas: usuários e itens (que se relacionam). Portanto, o modelo selecionado foi o *ItemSimilarityRecommender*. A métrica utilizada para computar a similaridade entre os itens é a **Jaccard Similarity**. Segundo a definição obtida na Wikipedia[7] o índice de Jaccard, ou coeficiente de semelhança de Jaccard, é uma estatística usada para comparar a similaridade e a diversidade dos conjuntos de amostras. Ainda, o coeficiente de Jaccard mede a similaridade entre conjuntos de amostras finitas e é definido como o tamanho da interseção dividido pelo tamanho da união dos conjuntos de amostras.

O modelo *ItemSimilarityRecommender* primeiro computa a similaridade entre um par de itens baseado na observação dos usuários que interagiram com ambos os itens.

[7] http://en.wikipedia.org/wiki/Jaccard_index

Para medir o valor da similaridade $S(i, j)$ entre o item i e o item j é utilizado a métrica Jaccard, que é dada por:

$$JS(i,j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|} \tag{3.1}$$

onde U_i é o conjunto de usuários que visualizar/gostaram/avaliaram o item i e U_j é o conjunto de usuários que visualizar/gostaram/avaliaram o item j

O resultado da recomendação deste modelo será a sugestão de um item j para um usuário u baseado em uma média ponderada obtida a partir de observações anteriores I_u dos demais usuários.

O resultado final dessa plataforma pode ser obtido via interface gráfica ou arquivo de saída no formato *.csv*. Essa saída contém o par de itens (filmes) e uma coluna *score* que contém um valor de similaridade variando entre 0 e 1, onde os valores maiores indicam um maior grau da similaridade.

3.3 Algoritmo LAP

O algoritmo LAP, conforme previamente apresentado no Capítulo 2, foi proposto por K. A. Zweig e desenvolvido na Technische Universität Kaiserslautern na Alemanha. Esta solução apresenta diferentes implementações em (BRUGGER et al., 2015) que visam principalmente escalabilidade e redução do tempo de execução. A versão utilizada deste algoritmo permite que o usuário indique o número de samples e swaps desejado.

A implementação ainda se encontra em um estágio inicial no que diz respeito a interface e interatividade com o usuário consumidor da ferramenta de recomendação, ou seja, não existe uma interface gráfica para solicitar a recomendação ou mesmo visualizar de forma interativa os resultados de uma recomendação já processada. Basicamente o que se tem é um pacote, desenvolvido em linguagem C, que contém os algoritmos que ao final geram uma saída em um arquivo texto.

A saída da recomendação gerada pelo LAP é um arquivo texto que segue o formato com as seguintes colunas (que serão explicadas a seguir):

$$filme1 \quad | \quad filme2 \quad | \quad p-value \quad | \quad z-score \quad | \quad cooc\_fdsm \quad | \quad cooc$$

Dada a base de dados de entrada, o arquivo de saída contém todas as combinações de pares de filmes possíveis e os respectivos valores para cada uma das colunas acima, estes valores (ou coeficientes) representam a intensidade do relacionamento entre cada par de filme.

Essa implementação gera as recomendações utilizando o modelo User-based, baseada em Memória, ou seja, identifica e gera as vizinhanças olhando para o conjunto de usuários para, a partir daí, poder identificar os filmes similares e então gerar uma recomendação para um determinado usuário.

A seguir é apresentado o funcionamento do algoritmo (como cada um dos valores acima são computados), bem como o modelo de recomendação utilizado.

3.3.1 Modelo utilizado para Recomendação

O fluxo de como o algoritmo LAP realiza as operações e computa os coeficientes de similaridade está ilustrado na Figura 3.4.

O algoritmo recebe como entrada um grafo bipartido, contendo usuários e filmes. Após a leitura dos dados os três macro passos são realizados:

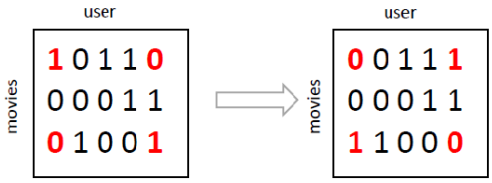
1. PASSO: É computado o valor de $cooc_{inicial}$ para todos os pares de filmes, esse resultado é armazenado. A explicação do que é o *cooc* foi previamente apresentada no Capítulo 2, mas, relembrando, a co-ocorrência é o evento que ocorre quando um par de filmes é assistido ou avaliado por 2 ou mais usuários.

2. PASSO: Inicia o processo de amostragem, que é a base de como a "recomendação" emerge. Para este processo existem dois conceitos importantes: *swaps* - que são as trocas - e *samples* - que são as amostras.

   2.1 São realizadas as trocas de filmes entre os usuários, essas trocas são chamadas de *swaps*. Há estudos e implementações que exploram qual seria a quantidade de trocas necessárias pra produzir bons resultados e de acordo com os resultados apresentados em (BRUGGER et al., 2015) o número de trocas sugerido é de $|arestas|$ x $ln(|arestas|)$, essa quantidade de trocas ocorrerá para cada umas das 10.000 amostras[8]. Para entender como os swaps são calculados, considere a matriz de adjacência M (que armazena usuários e filmes) representada na Figura 3.3.

[8]O número de 10.000 amostras também é dito em (BRUGGER et al., 2015) como um número seguro para produzir bons resultados.

Figura 3.3: Matriz M e Cálculo de Swaps



Fonte: (JOHN, 2015)

De acordo com (JOHN, 2015), para o processo inicial das trocas seleciona-se duas arestas aleatoriamente, no exemplo dado são elas: $M(0,0)$ e $M(2,4)$. Um swap só será possível se as seguintes condições forem satisfeitas: ambas arestas existirem (ou seja, se ambas tiverem o valor 1) e se $M(2,0)$ = $M(0,4) = 0$. No exemplo apresentado pelo autor, um swap seria representado por: $M(0,0) \leftarrow 0$, $M(2,4) \leftarrow 0$, $M(2,0) \leftarrow 1$ e $M(0,4) \leftarrow 1$.

2.2 Para cada amostragem é calculado $cooc_{amostra}$ entre cada par de filmes, o valor é armazenado.

2.3 Ao final de cada amostragem é calculado também o valor de $p-value$, que é definido como:

$$p{-}value(m_i, m_j) = \sum_{i=1}^{amostras} \left\{ \begin{array}{c} 1, \text{ se } cooc_{amostra}(m_i, m_j) > cooc_{inicial}(m_i, m_j) \\ \\ 0, \text{ caso contrário} \end{array} \right\}$$

3. PASSO: No último passo, após a computação de $p-value$, $cooc_{inicial}$ e $cooc_{amostra}$ para cada amostragem, é computado o valor de $z-score$, tal como segue:

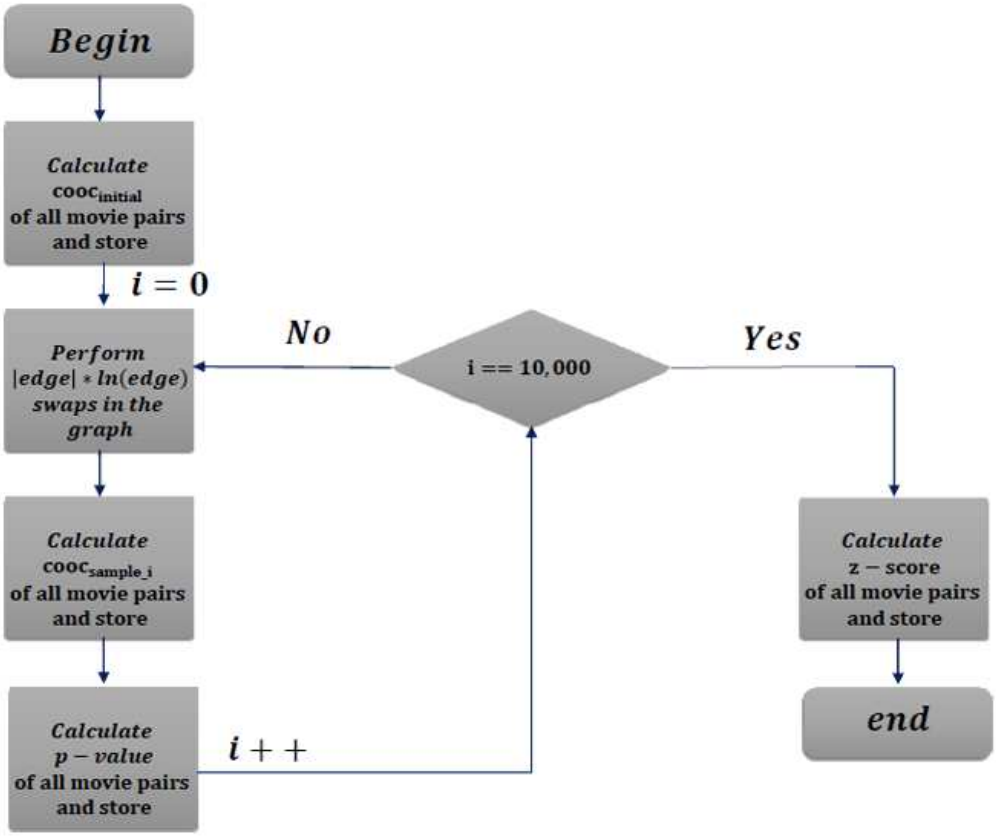
$$z{-}score(m_i, m_j) = \frac{(cooc_{inicial}(m_i, m_j)) - (cooc_{FDSM}(m_i, m_j))}{\sqrt{\sum_{i=1}^{amostras}(\frac{(cooc^2_{amostra}(m_i,m_j))}{amostras}) - \sum_{i=1}^{amostras}(\frac{(cooc_{amostra}(m_i,m_j))}{amostras})^2}}$$

onde

$$cooc_{FDSM}(m_i, m_j) = \frac{1}{amostras} \sum_{i=1}^{amostras} cooc_{amostra}(m_i, m_j)$$

Finalmente é possível obter uma recomendação para um par de filmes, para isso deve-se observar os valores de $p - value$ e $z - score$, eles são os coeficientes de similaridade. Quanto menor o valor $p - value$ ou quanto maior o valor de $z - score$, maior será o grau de similaridade entre um par de filmes *filme 1* e *filme 2*.

Figura 3.4: Fluxo do Algoritmo LAP



Fonte: (JOHN, 2015)

4 ANÁLISE DO ALGORITMO LAP

A execução do trabalho se deu em duas etapas que serão apresentadas em duas seções distintas neste Capítulo. Na primeira Seção é apresentado o **processo de *benchmarking*** das plataformas de recomendação e na segunda apresenta-se a **pesquisa** desenvolvida para validar as recomendações de filmes geradas pelo algoritmo LAP.

O principal objetivo, já previamente discutido no Capítulo 1, foi fazer uma estudo comparativo por meio de um benchmarking e de uma pesquisa com usuários para avaliar a *qualidade* e *performance* do algoritmo LAP perante outras soluções já consolidadas e disponíveis no mercado.

O banco de dados utilizado para gerar as recomendações e também para a pesquisa com usuários foi uma base de filmes do Netflix contendo apenas duas colunas: usuários e filmes. Esta base relaciona usuários e filmes levando em conta as avaliações que cada usuário deu para os filmes, mas não foi usada a informação do valor desta avaliação (*ratings*). Os bancos de dados do Netflix foram disponibilizados, no passado, pela própria empresa e ainda podem ser encontrados na Web. A versão usada aqui é composta por um ID de usuário - um identificador numérico que representa cada usuário individualmente - e também por um ID de filme para identificar cada filme. O formato descrito no exemplo que segue:

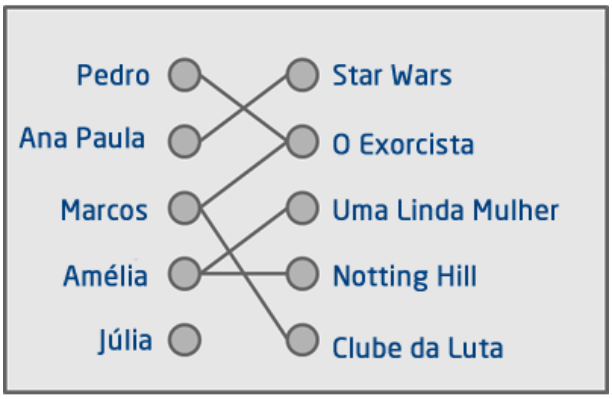
Usuario ID , Filme ID
10001 , 15
11002 , 19
21010 , 53
21010 , 109

Essa base de dados utilizada é chamada de implícita, o que significa que se pode apenas saber quais usuários avaliaram ou assistiram filmes, mas não tem a informação, por exemplo, se o usuário gostou do filme ou não, da nota de avaliação, etc. Por conta desse cenário, foi necessário selecionar apenas as plataformas que aceitavam como entrada dados implícitos e que utilizam todas o mesmo tipo de filtragem, neste caso Filtragem Colaborativa, pois lembrando que para uma Filtragem Baseada em Conteúdo ou mesma uma Híbrida seria necessário um banco de dados com mais informações sobre os filmes ou sobre os usuários ou ainda sobre ambos.

Conforme já explicado no Referencial Teórico, a base de dados de entrada, tal como exemplo acima, tem a estrutura de um grafo bipartido, onde um conjunto representa

os usuários e o outro conjunto os filmes, as arestas representam a relação (ou ligação) entre os usuários e os filmes, como a imagem abaixo ilustra:

Figura 4.1: Dados implícitos representados em um grafo bipartido



Tanto para o etapa de *benchmarking* quanto para a pesquisa, ateve-se a manipular dados de um domínio específico, de filmes, mas poderia ser aplicado a qualquer domínio desde que se tenha posse de um arquivo (o *Ground Truth* verdade para poder efetuar o cálculo do PPV. Um fato importante, ainda não destacado aqui, é que uma das intenções também do algoritmo LAP é que possa processar alta carga de dados e não apenas bases de dados pequenas, visto que o algoritmo LAP foi desenvolvido para ser executado em um cluster utilizando paralelismo[1], no entanto existe uma grande dificuldade de encontrar bancos de dados livres e disponíveis para uso e que tenham volumes expressivos de dados. Durante a execução deste trabalho tentou-se contato com algumas empresas líderes de e-commerce no Brasil a fim de obter dados para estas simulações, mas não tive sucesso. Esse é um dos motivos pelos quais boa parte das pesquisas neste contexto de recomendação ainda se detém no domínio de filmes, que são bases fáceis de obter diretamente na internet e com razoável volume de registros de dados.

Definido então o domínio e o banco de dados a ser usado como entrada para obter as recomendações, a seguir apresento o detalhamento de como cada plataforma gera as recomendações na prática.

4.1 BENCHMARKING

O processo de avaliação e seleção dessas plataformas, já mencionado previamente, no Capítulo 3, deu-se da seguinte forma:

[1]Paralismo é uma técnica usada em computação que possibilita que vários cálculos/tarefas/instruções sejam realizados ao mesmo tempo.

1. Inicialmente, selecionou-se aleatoriamente algoritmos de recomendação (de plataformas) sem um critério específico definido;

2. Fez-se o download, instalação e configuração de cada uma das plataformas elencadas na Tabela 3;

3. Identificou-se as plataformas ativas e com históricos de atualização recente - caso estivesse descontinuada era descartada do processo de *benchmarking*;

4. Entendimento do(s) algoritmo(s) utilizado(s) por cada plataforma;

5. Seleção de apenas aquelas que geram recomendações em cima de uma base de dados implícita de usuários e filmes.

Após a execução das etapas mencionadas, chegou-se então às duas plataformas selecionadas: **Oryx** e **GraphLab Create**. Ambas, após execução das etapas 2 e 4 acima, mostraram-se plataformas interessantes por também gerar recomendação a partir de dados implícitos porém cada uma com diferente algoritmo preditivo, então assumiu-se que fazer uma análise comparativa com elas daria uma boa resposta ou pelo menos um bom indicativo da real aplicabilidade do algoritmo LAP em contextos práticos. Todas elas rodam em ambiente offline, i.e., foi necessário fazer o download e instalação local para obter as recomendações.

Definidas então as plataformas, iniciou-se o trabalho de análise da *performance* e da *qualidade* de cada algoritmo.

O ambiente utilizado para executar o *benchmarking* foi uma máquina virtual com as seguinte características:

- Ubuntu 16.04.1 LTS (GNU/Linux 4.4.042generic x86_64);
- Intel(R) Xeon(R) CPU E5620 @ 2.40GHz;
- 20 GB de memória RAM.

Para medir a *performance* foram observados dois parâmetros: **Memória RAM** - em gigabytes (GB) - e **Tempo** - em segundos (s) - necessários para o processamento completo desde a carga dos dados até a obtenção das recomendações.

Todas as 3 plataformas forneceram diretamente a contagem do tempo de execução em segundos, durante e/ou ao final da sua execução, facilitando assim a obtenção deste dado. Sobre a memória utilizada, a plataforma Oryx informa diretamente na sua

interface o consumo, as demais não. Nesse caso, o que se fez para medir o consumo da memória foi observar via comando *top* no linux.

Para executar o*benchmarking* tanto o Graphlab quanto LA utilizou-se 4 núcleos de processador, e para execução do Oryx foi utilizado um ambiente com Hadoop com apenas 1 núcleo. Acredita-se que utilizando Oryx com paralelismo de 4 núcleos o tempo de resposta da execução da recomendação poderia ser melhorado.

Para medir a *qualidade* utilizou-se o PPV - *Positive Predictive Values*[2] (Valores Preditivos Positivos) - que é uma medida utilizada para interpretar a acurácia estatística. Essa foi a medida usada em (BRUGGER et al., 2015) para obter a acurácia do LAP. É uma métrica relativamente fácil de ser mensurada. No entanto para poder utilizá-la é necessário ter uma base de dados chamada *Ground Truth* (GT). Essa base de dados (esse arquivo), como o nome mesmo sugere, é um "conjunto de dados verdade", é gerado explicitamente por humanos e contém a relação de pares de filmes que são considerados de fato semelhantes, este filmes obviamente foram retirados do mesmo banco de dados do Netflix usado como entrada para gerar as recomendações. O cálculo do PPV é o que segue:

$$PPV = \frac{numero\ de\ verdadeiros\ positivos}{numero\ de\ verdadeiros\ positivos\ +\ numero\ de\ falsos\ positivos} \quad (4.1)$$

onde:

numero de verdadeiros positivos + *numero de falsos positivos* : é a quantidade total de pares de filmes existentes no arquivo GT.

numero de verdadeiros positivos : é o número de pares de filmes presentes no GT e que estão também presentes no arquivo de saída da recomendação gerada pela plataforma, i.e., representa a intersecção entre os dois conjuntos de dados: a saída gerada pela plataforma e o arquivo GT.

Como cada uma das 3 plataformas requer um formato distinto de entrada de dados e gera também um formato de saída diferente, se fez necessário fazer um tratamento destas entradas e saídas para que ao final fosse possível computar o valor do PPV. O fluxo de como o processamento para cada uma das plataformas foi executado está representado na Figura 4.2 e descrito em itens, abaixo:

1. Obter os arquivos de saída de cada uma das 3 plataformas;

[2]https://en.wikipedia.org/wiki/Positive_and_negative_predictive_values

2. Gerar uma nova saída mantendo apenas os filmes presentes no arquivo GT;

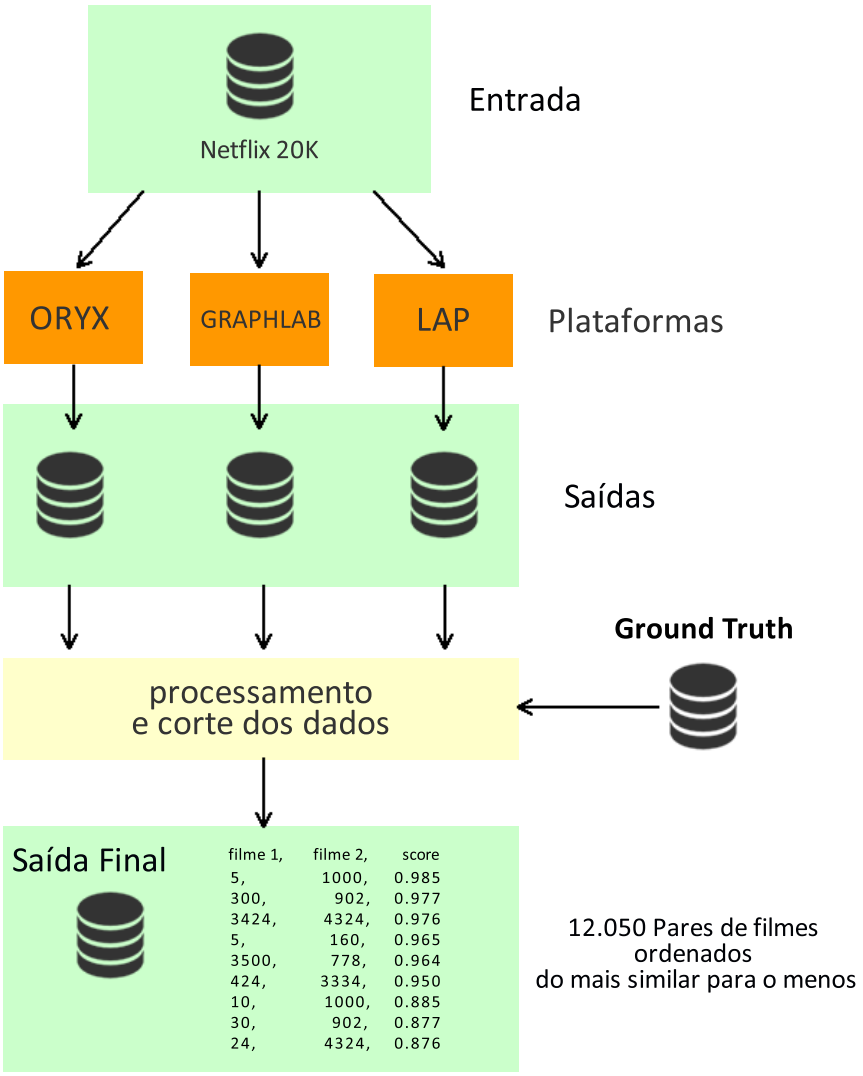
3. Ordenar a nova saída pelo score usado para medir a similaridade;

4. Da saída ordenada obter os 12.050 pares tidos como melhores recomendações (esta etapa está representada na imagem a seguir como "*processamento e corte dos dados*");

5. Cálculo do PPV usando esta última saída (item 4) e o arquivo GT original.

Para cada um dos passos acima se fez necessário programar um script diferente. Esses scripts foram desenvolvidos em linguagem de programação C++ e PHP.

Figura 4.2: Fluxo do tratamento das saídas de cada plataforma para chegar ao cálculo do PPV



A saída final, após o processamento descrito nas sub-seções anteriores, conterá exatamente a mesma quantidade de pares de filmes existente no arquivo GT, que é de

12.050 pares, e então é calculado o PPV. O algoritmo desenvolvido para o cálculo do PPV encontra-se nos Anexos deste documento. O valor do PPV dá o percentual de quantas das recomendações sugeridas por cada plataforma está presente no arquivo GT, ou seja, quantas recomendações geradas estariam "corretas".

Na sub-seções a seguir é descrito muito brevemente como cada uma das três plataformas geram na prática as recomendações, dado o arquivo de entrada com filmes e usuários.

4.1.1 Oryx

Considerando que o ambiente já está configurado e preparado com os pré-requisitos necessários e descritos no Capítulo de Trabalhos Relacionados, a execução da recomendação no Oryx é obtida seguindo os passos:

1. Iniciar o serviço do HADOOP - nestes experimentos foi utilizado uma versão Single Cluster[3], i.e., uso de um único nodo de processador;

2. Salvar o arquivo de entrada em formato .csv contendo filmes e usuários no diretório *oryx/example/00000/inbound*;

3. Rodar a recomendação (camada de computação) através do comando:
   java -Dconfig.file=oryx.conf -jar computation/target/oryx-computation-1.1.1-SNAPSHOT.jar

4. Iniciar a camada de serviço, que permite obter as recomendações na página web da plataforma - que também é instalado localmente com todo o pacote desta ferramenta
   sudo java -Dconfig.file=oryx.conf -jar serving/target/oryx-serving-1.1.1-SNAPSHOT.jar.

O modelo fornecido no Oryx para o cálculo das recomendações é o ALS. O embasamento teórico desse modelo foi fornecido no Capítulo 3. Os parâmetros utilizados foram os sugeridos como padrão do algoritmo[4], que seguem:

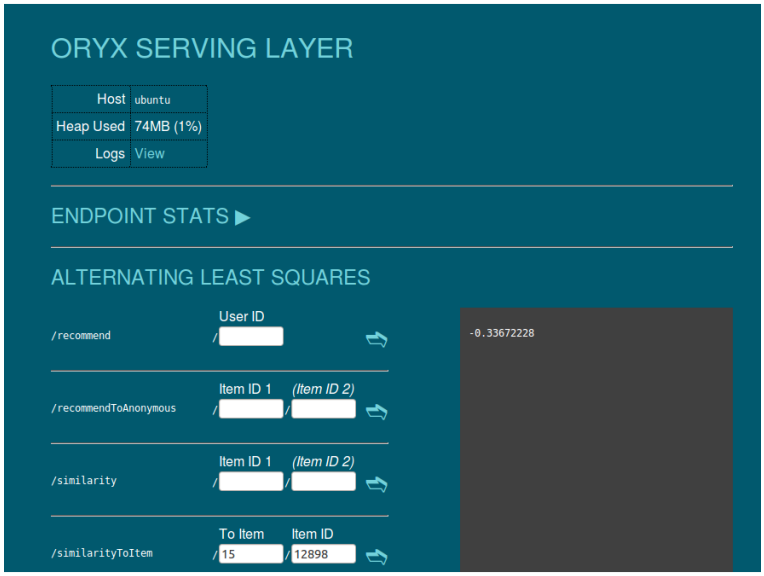
[3]https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SingleCluster.html
[4]https://github.com/cloudera/oryx#clustering-example

$$model = \$als - model$$
$$model.features = 25$$
$$model.lambda = 0.065$$

Para a computação das recomendações foi utilizada a arquitetura do HADOOP, como já descrito acima, na versão *Single Node*.

No caso do Oryx não foi identificado na sua instalação o arquivo de saída final com todos os pares de filmes e respectivas medidas de similaridade, então para obter as recomendações foi necessário desenvolver um script que acessava a página Web, da camada de serviço, representada na Figura 4.3 , que percorria todo o arquivo de entrada Netflix 20k e o arquivo GT e inseria o par de filmes nos campos *To Item* e *Item ID* no label *similarityToItem* e por meio de um requisição HTTP do tipo GET solicitava e obtinha, desta página web, a medida de similaridade (na caixa preta à esquerda - ver na imagem seguinte).

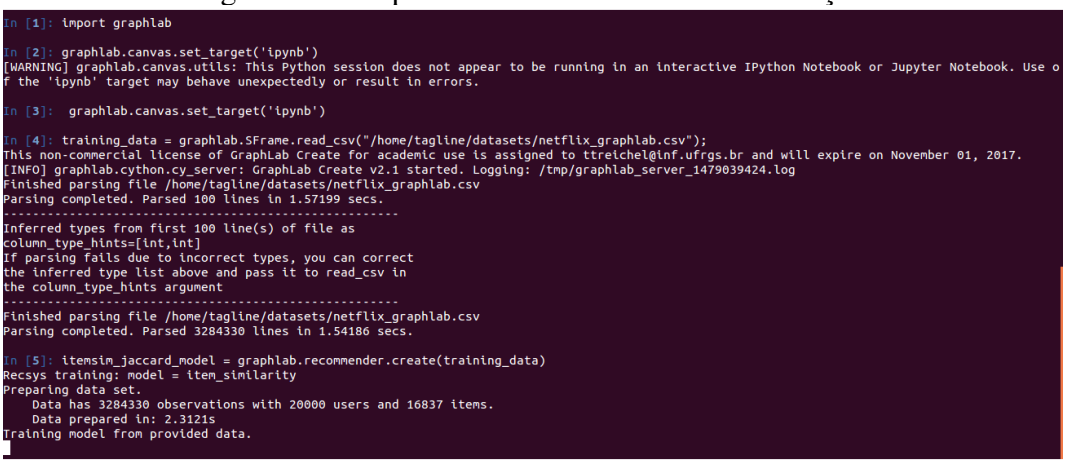
Figura 4.3: Página Web para Solicitar Recomendações



4.1.2 GraphLab Create

Após configurar o ambiente de acordo com os pré-requisitos enumerados no Capitulo 3, foi possível obter as recomendações facilmente via execução de algumas instruções diretamente em código usando linguagem Python. O modelo de recomendação utilizado foi o $get\_similar\_items$[5].

Figura 4.4: Graphlab Processando as Recomendações



A recomendação obteve-se então executando os passos seguintes:

1. Iniciar o serviço Anaconda com o comando: $ipython\ notebook$
2. $ipython$ - Comnado para acessar o ambiente e poder digitar os comandos seguintes
3. $import\ graphlab$ e $graphlab.canvas.set\_target('ipynb')$ - para instanciar a framework
4. $training\_data = graphlab.SFrame.read\_csv(\text{"}/home/tagline/anaconda2/lib/python2.7/site-packages/graphlab/dataset/netflix.csv\text{"})$ - carregar o arquivo de dados de entrada
5. $itemsim\_jaccard\_model = graphlab.recommender.create(training\_data)$ - para treinar o modelo
6. $nn = itemsim\_jaccard\_model.get\_similar\_items(items = None, k = 100)$ - solicitar, por exemplo, 100 recomendações para cada filme

[5]https://turi.com/products/create/docs/generated/graphlab.recommender.item_similarity_recommender.ItemSimilarityRecommender.get_similar_items.html#graphlab.recommender.item_similarity_recommender.ItemSimilarityRecommender.get_similar_items

7. $nn.export\_csv('output/training\_data.csv')$ - salvar as recomendações no arquivo de saída

Graphlab oferece ainda uma interface gráfica muito interessante, e é possível usá-la para fazer a análise dos dados, tanto de entrada quanto dados de saída, organizando-os em listas e também em grafos.

4.1.3 LAP

O algoritmo LAP possui diferentes implementações práticas conforme descrito em (BRUGGER et al., 2015). As simulações aqui executadas usaram a versão LAP_samples_and_swaps_settable[6].

Esta versão permite setar a quantidade de $samples$ e $swaps$ desejada. No referencial teórico é dito que $|arestas|$ x $ln(|arestas|)$ é um número seguro de swaps e 10.000 um número seguro de amostras, no entanto o mesmo artigo (BRUGGER et al., 2015) também apresenta resultados que começam a convergir e se estabilizam a partir de 1 milhão de swaps e entre 1.000 e 5.000 samples. Sendo assim, este experimento foi submetido usando os seguintes parâmetros: 10.000 samples e 1 milhão de swaps.

Os pré-requisitos para rodar o algoritmo são ter a biblioteca *openmpi* instalada e setar a quantidade de threads a ser usada *export OMP_NUM_THREADS*. Com isso, executar o comando $./LAP\_AA\ \$DATASETPATH\ \$INPUTFILE\ \$SAMPLES\ 1\ 0\ \$SWAPS$ dentro do diretório onde estão os arquivos fontes que compõe o algoritmo.

A Figura 4.5 ilustra o a computação da recomendação com as rodadas de swaps a cada amostragem.

4.2 PESQUISA COM USUÁRIOS

A pesquisa realizada com usuários partiu da motivação de ter um método adicional, além do cálculo do PPV, para verificar a qualidade das recomendações geradas pelo algoritmo LAP. Portanto, este experimento foi executado utilizando apenas as recomendações de filmes geradas pelo LAP.

O desenvolvimento e condução desta pesquisa foi realizada em 6 etapas que serão explicadas a seguir.

[6]https://git.rhrk.uni-kl.de/EIT-Wehn/bigdata.graphs

Figura 4.5: LA: Algoritmo Processando Recomendações

```
Rank 0 here!
Input file: /home/tagline/datasets//Netflix_Dataset_Good_20k
Graph info: #events 16837, #actors 20000, #edges 3284330
Reading duration: 4.770255 s
Calculating original co-occurrence...
Time for first index setting: 0.089603 s
Time for Initial Co-occurrence: 3.647140 s
Original co-occurrence done.
Sampling... Seed: 396764002
Randomizing samples 0-16 of 10000
Calculating co-occurrence...
16 of 10000 samples done.
Randomizing samples 16-32 of 10000
Calculating co-occurrence...
32 of 10000 samples done.
Randomizing samples 32-48 of 10000
Calculating co-occurrence...
48 of 10000 samples done.
Randomizing samples 48-64 of 10000
Calculating co-occurrence...
64 of 10000 samples done.
Randomizing samples 64-80 of 10000
Calculating co-occurrence...
80 of 10000 samples done.
Randomizing samples 80-96 of 10000
Calculating co-occurrence...
96 of 10000 samples done.
Randomizing samples 96-112 of 10000
Calculating co-occurrence...
112 of 10000 samples done.
```

- Geração da recomendação usando o algoritmo LAP;
- Importação dos resultados da recomendação para um banco de dados MySQL;
- Preparação do questionário e avaliação dos filmes;
- Modelagem e desenvolvimento de uma página web para condução da pesquisa;
- Divulgação da pesquisa;
- Coleta e análise dos resultados.

4.2.1 Geração da Recomendação usando o algoritmo LAP

A geração da recomendação foi executada da mesma forma como descrito na sub-Seção **4.1.3**.

O resultado da recomendação foi um arquivo contendo aproximados 34 milhões (34.425.909) de pares de filmes, cada um associado a um valor de $p-value$ e um valor de $z-score$. A variação do $z-score$ foi de [0.002273 , 133.788]. A variação do $p-value$ foi de [0.0 , 0.9].

4.2.2 Importação dos resultados da recomendação para um banco de dado MySQL

Como a saída do LAP é gerada em arquivo texto, optou-se por importar os resultados para um banco de dados MySQL a fim de melhorar a performance da página Web de pesquisa, uma vez que bancos de dados já oferecem todo o aparato necessário para otimizar buscas de dados em arquivos. A estrutura da tabela que contém as recomendações é mostrada na Figura 4.6.

Figura 4.6: Tabela que armazena as recomendações

#	Nome	Tipo	Colação	Atributos	Nulo	Padrão	Extra
1	recommendation_result_id	bigint(20)			Não	*Nenhum wrap (padrão: none)*	AUTO_INCREMENT
2	movie_id_1	bigint(20)			Não	*Nenhum wrap (padrão: none)*	
3	movie_id_2	bigint(20)			Não	*Nenhum wrap (padrão: none)*	
4	pvalue	double			Sim	*NULL*	
5	zscore	double			Sim	*NULL*	
6	cooc_fdsm	float			Sim	*NULL*	
7	cooc	float			Sim	*NULL*	

Em (BRUGGER et al., 2015) e (ZWEIG, 2010) é informado que quanto maior o valor de $z - score$ e menor o $p - value$ maior é a semelhança entre dois filmes, mas ainda não é possível definir um valor preciso como ponto de corte para decidir sobre a similaridade. Por conta disso, intuitivamente foi utilizado para este experimento com usuários apenas os pares de filmes que possuíam um $z - score \geq 10$. Assim, reduziu-se a base de dados de 34.425.909 registros para 615.159 registros. Para melhorar o tempo de respostas das consultas (recomendações) foram adicionados índices para os IDs de filmes $movie\_id\_1$ e $movie\_id\_2$ e também as colunas de $p - value$ e $z - score$. Durante a pesquisa foram sugeridos os pares de filmes obtidos da tabela contendo os 615.159 registros.

4.2.3 Preparação do questionário para avaliação dos filmes

O questionário foi dividido em 2 partes. A primeira parte, na primeira página, foi solicitado ao participante responder sobre sua **faixa etária** e a **frequência com que assiste filmes**. Consideramos necessárias essas informações para auxiliar na posterior análise e entendimento dos resultados.

Na segunda parte da pesquisa, pediu-se para, dado 1 par de filmes, avaliá-los

como semelhantes respondendo as seguintes perguntas:

1. *Você considera que os filmes acima tem alguma semelhança? (SIM | NÃO)*

2. *Em uma escala de 1 (pouco semelhante) à 5 (muito semelhante) como você classifica os filmes?*

A pesquisa solicitou que fossem avaliados pelo menos 5 pares de filmes e foi disponibilizado um botão de "PULAR" caso o usuário não quisesse avaliar o par de filmes. Não era pré-requisito já ter assistindo os filmes e, por isso, abaixo de cada nome de filme foi disponibilizado um link que levava para a página do *Google*, já com a busca filtrando pelo nome do filme, onde o participante poderia ler algumas informações e tomar a decisão sobre a semelhança dos dois filmes sugeridos.

Como a pesquisa foi divulgada no Brasil e no exterior, a página Web com o questionário foi disponibilizada em 2 idiomas: Português e Inglês.

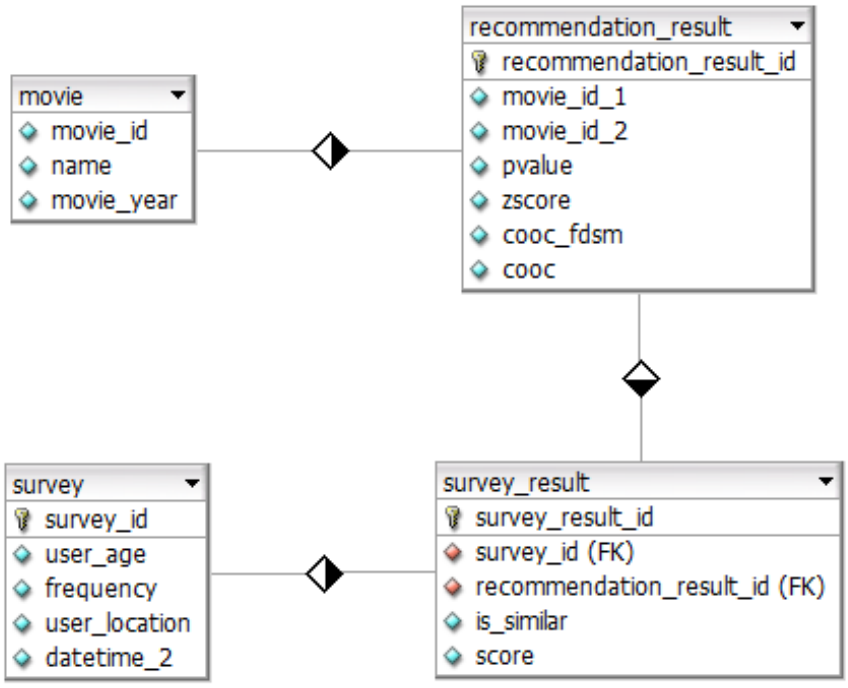
4.2.4 Modelagem e desenvolvimento de uma página web para condução da pesquisa

Para que os participantes pudessem responder a pesquisa, foi criada uma página na internet. Para o desenvolvimento desta página foram utilizadas as linguagens de programação e de marcação: PHP, HTML/CSS e JavaScript (JQUERY). A identidade visual do website seguiu os padrões do website da Technische Universität Kaiserslautern, pois os resultados do *benchmarking* foram apresentados também para professores daquela instituição, durante o período em que parte deste trabalho foi executado naquela universidade.

Este website utiliza ainda um banco de dados MySQL onde estão armazenados os dados necessários para a pesquisa. A Figura 4.7 apresenta um Diagrama ER - Entidade Relacionamento - da estrutura de como os dados dos filmes, das recomendações e das respostas da pesquisa estão armazenadas.

O fluxo de como o participante foi conduzido a responder a pesquisa segue abaixo, primeira etapa como na Figura 4.8 e segunda parte como na Figura 4.9.

Figura 4.7: Estrutura do Banco de Dados



4.2.5 Divulgação da Pesquisa

O link de acesso à página Web de pesquisa[7] foi divulgado na lista de alunos da graduação e pós graduação do departamento de Informática da UFRGS e em minha página pessoal no Facebook. A pesquisa foi divulgada no Brasil e no exterior e por isso o site foi desenvolvido em multilíngua (inglês e português).

4.2.6 Coleta e análise dos resultados

Os resultados da pesquisa ficaram armazenados nas tabelas *"survey"* e *"survey_result"*. Todas as respostas obtidas foram exportadas para um arquivo único *.csv*.

Para poder fazer a análise dos dados e gerar os gráficos de interesse foi utilizada uma ferramenta open-source[8] desenvolvida em Python chamada **IPython Notebook**. O arquivo *.csv*, acima descrito, foi importado nesta ferramenta para então executar a análise e obter os resultados de forma gráfica e visual.

Os resultados são apresentados no próximo Capítulo.

[7] http://tagline.zipernet.com.br/evalrec/recommendation
[8] http://jupyter.org

Figura 4.8: Primeira parte: participante entra com seus dados

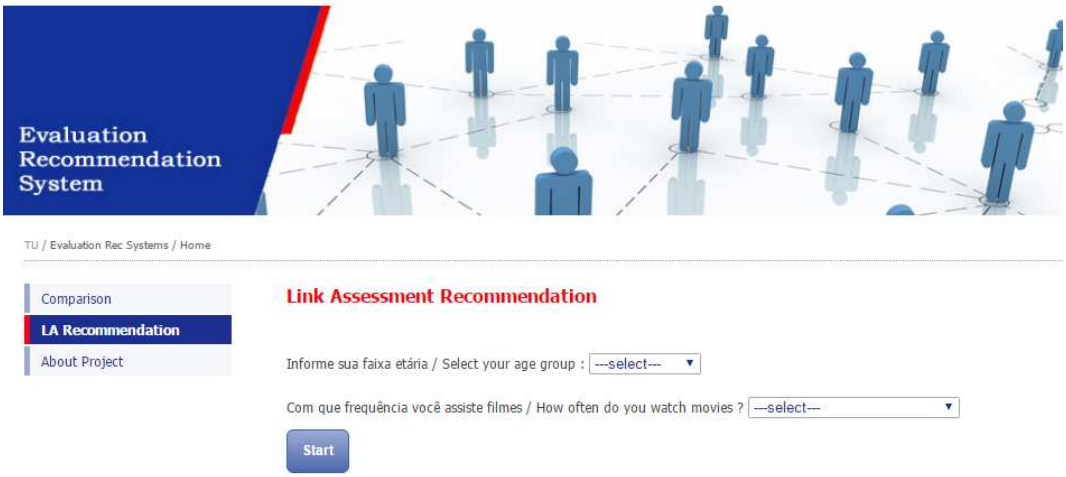
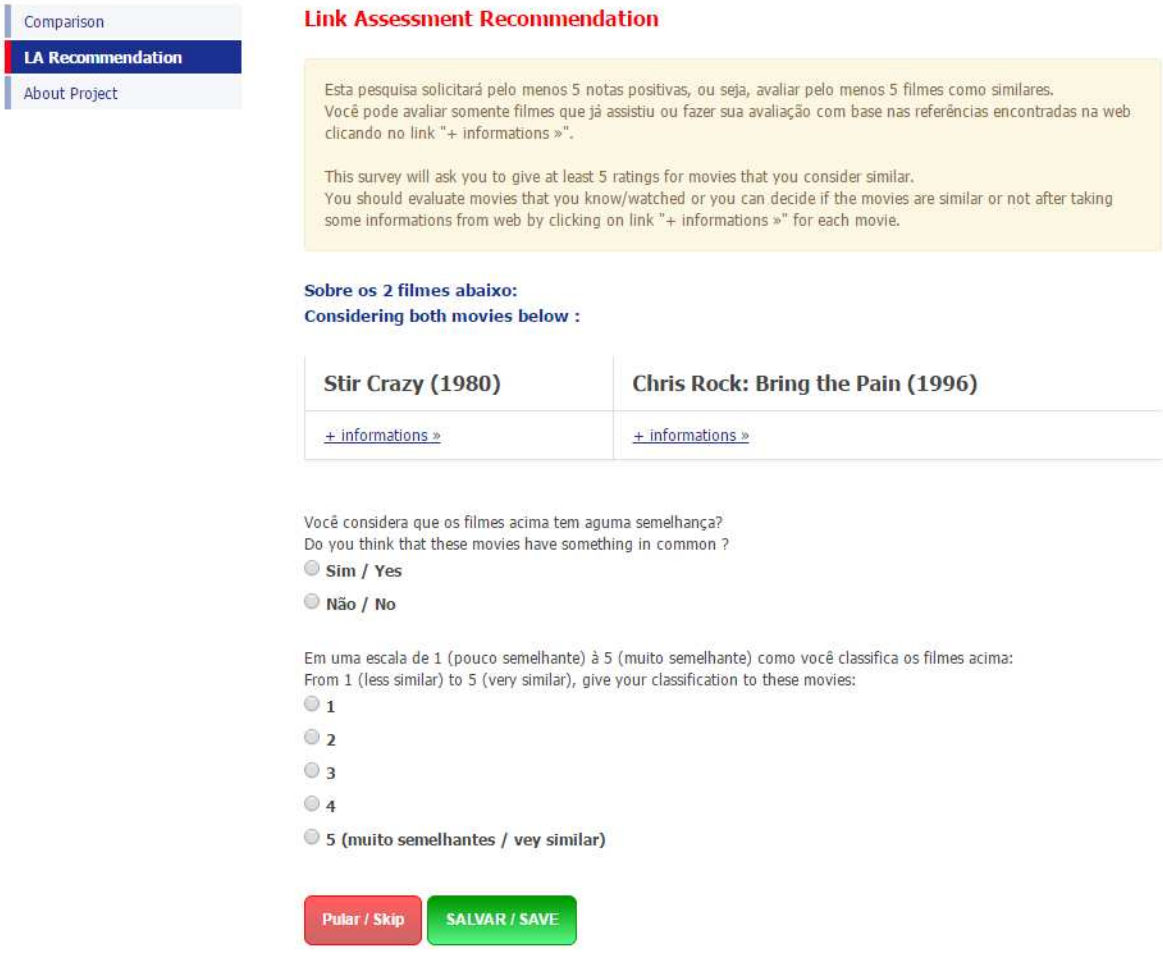


Figura 4.9: Segunda parte: participante é convidado a avaliar a similaridade entre dois filmes



5 ANÁLISE DOS RESULTADOS

Neste Capítulo é apresentado o conjunto de dados, utilizado para o efetuar o benchmarking e para a pesquisa com usuários, e também os resultados obtidos nessas execuções e as respectivas conclusões. O objetivo do benchmarking foi verificar o desempenho e a qualidade das recomendações geradas a partir do algoritmo LAP comparativamente com as recomendações geradas nas outras plataformas: Oryx e GraphLab Create. Todos os algoritmos aqui estudados utilizam filtragem colaborativa e aceitam como entrada dados implícitos.

5.1 O conjunto de dados

A base de dados de entrada utilizada foi o Netflix 20k, contendo 20 mil usuários, 16.837 filmes e 3.284.330 arestas (relacionamentos entre usuários e filmes). Essa mesma base de dados foi usada tanto para o processo de benchmarking quanto para a pesquisa com usuários. Lembrando ainda que para a pesquisa com usuários foram utilizadas apenas recomendações geradas a partir do algoritmo LAP.

Além disso, para o experimento com os usuários foi necessário também uma base de dados que continha as informações de nome e ano de cada filme, correspondente aos filmes presentes na base Netflix 20k. Essa base foi obtida também na Internet e segue a estrutura:

$$filmeID \quad | \quad nome \quad | \quad ano$$

5.2 Resultados - Benchmarking

Como mencionado previamente, o propósito do benchmarking era verificar o consumo de memória RAM, medido em GigaBytes, e tempo de relógio, medido em segundos, necessário para gerar as recomendações. A seguir, nas Figuras 5.1 e 5.2, são apresentados os valores mensurados e o comparativo entre as plataformas.

Observando-se as Figuras, tornou-se evidente que o desempenho do algoritmo LAP ainda precisa ser melhorado, especialmente no que diz respeito às heurísticas de

Figura 5.1: Memória Consumida

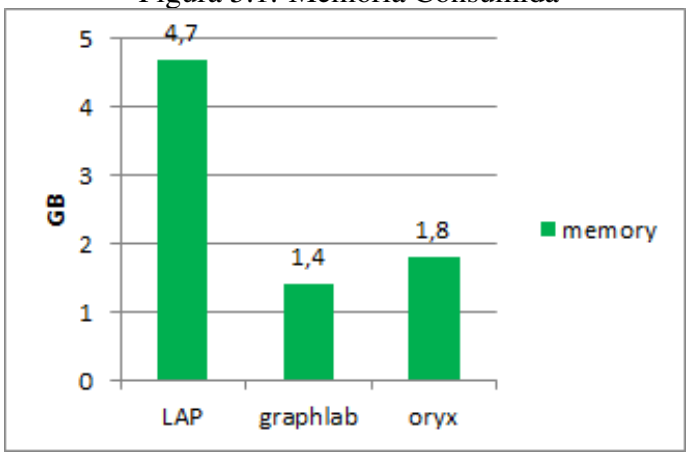
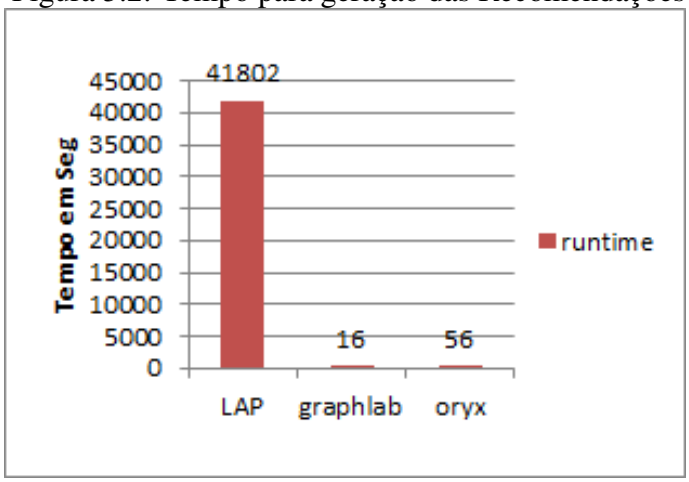


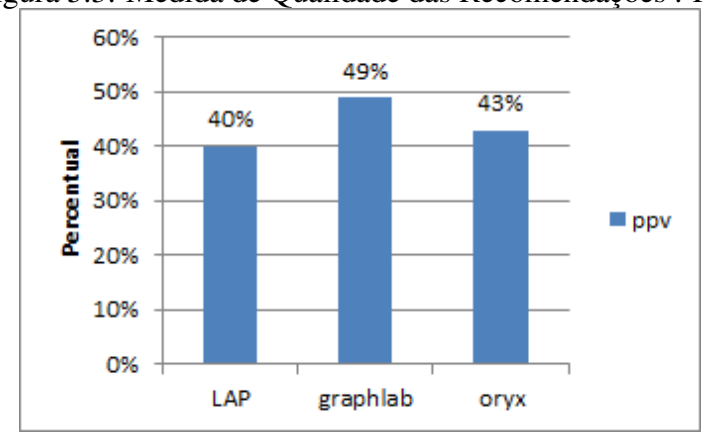
Figura 5.2: Tempo para geração das Recomendações



swaps, que acontecem todas em memória e demandam uma alta carga de processamento. Também ficou claro que os resultados obtidos nas recomendações geradas pelo GraphLab Create são bastante interessantes, dada a qualidade versus recursos necessários. É uma ferramenta de fácil utilização, o que pode ser um fator primordial para que a plataforma seja aplicável efetivamente.

Já em relação ao PPV, que foi a medida utilizada para computar a qualidade das recomendações, observando-se os valores apresentados na Figura 5.3 é possível verificar uma boa equivalência na qualidade, ficando o algoritmo LAP com o menor percentual mas ainda competitivo perante as demais plataformas. Lembrando que o cálculo do PPV, considerava as melhores recomendações de cada plataforma e que estavam presentes no arquivo GT.

Figura 5.3: Medida de Qualidade das Recomendações : PPV

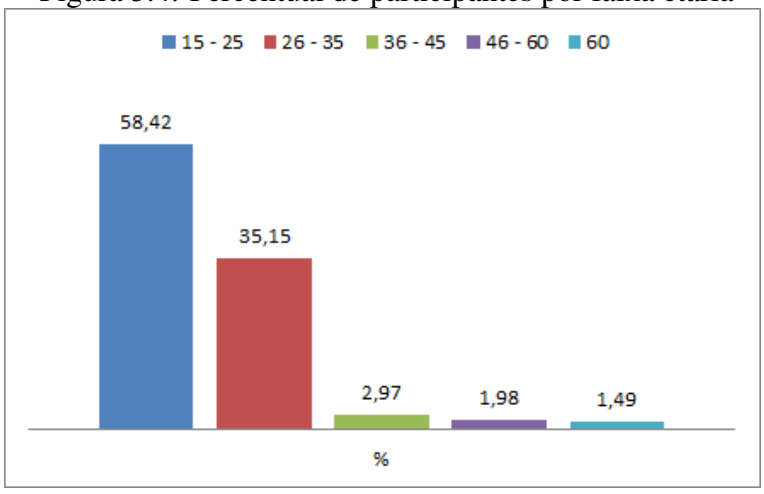


5.3 Resultados - Pesquisa com Usuários

A pesquisa contou com a participação de 202 pessoas. Os participantes em sua maioria eram brasileiros, mas também com participação de estrangeiros residentes na Alemanha.

A base de dados continha 615.159 pares de filmes com $z - score \geq 10$. As recomendações com valores de $z - score$ inferior à 10 foram descartadas. Na página onde o participante efetuava a avaliação era exibido randomicamente um par de filmes, da base de dados citada, e ao participante era solicitado que avaliasse pelo menos 5 pares de filmes.

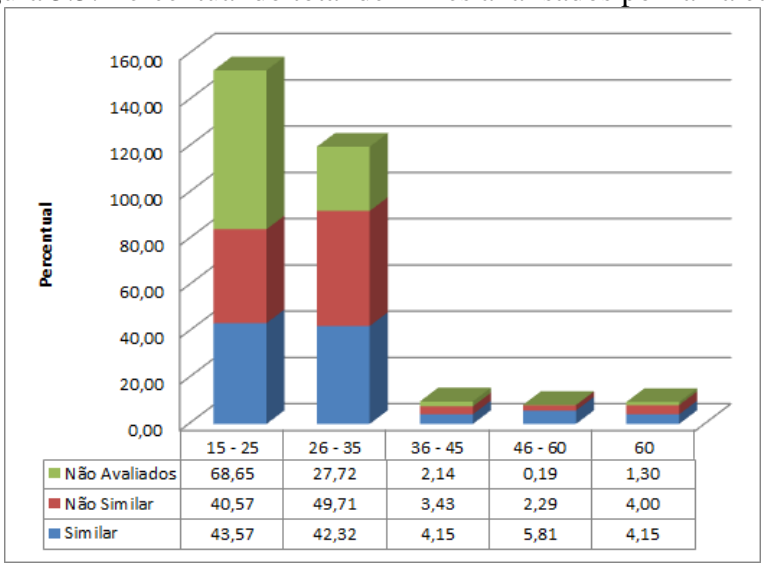
Figura 5.4: Percentual de participantes por faixa etária



Dos 615.159 registros, 1.666 filmes foram avaliados sendo 1.075 nulos. São considerados nulos os filmes em que o usuário não soube avaliar e usou o recurso de "PULAR". Assim, o montante de filmes efetivamente avaliados foi de 591, sendo 350 avaliados como NÃO SIMILAR e 241 como SIMILAR, portanto o percentual de filmes tidos como similares é de aproximados 41%. No Gráfico 5.5 está a distribuição do per-

centual dos filmes avaliados e não avaliadas por faixa etária e no Gráfico 5.4 o percentual de participantes que avaliaram os filmes em função da faixa etária.

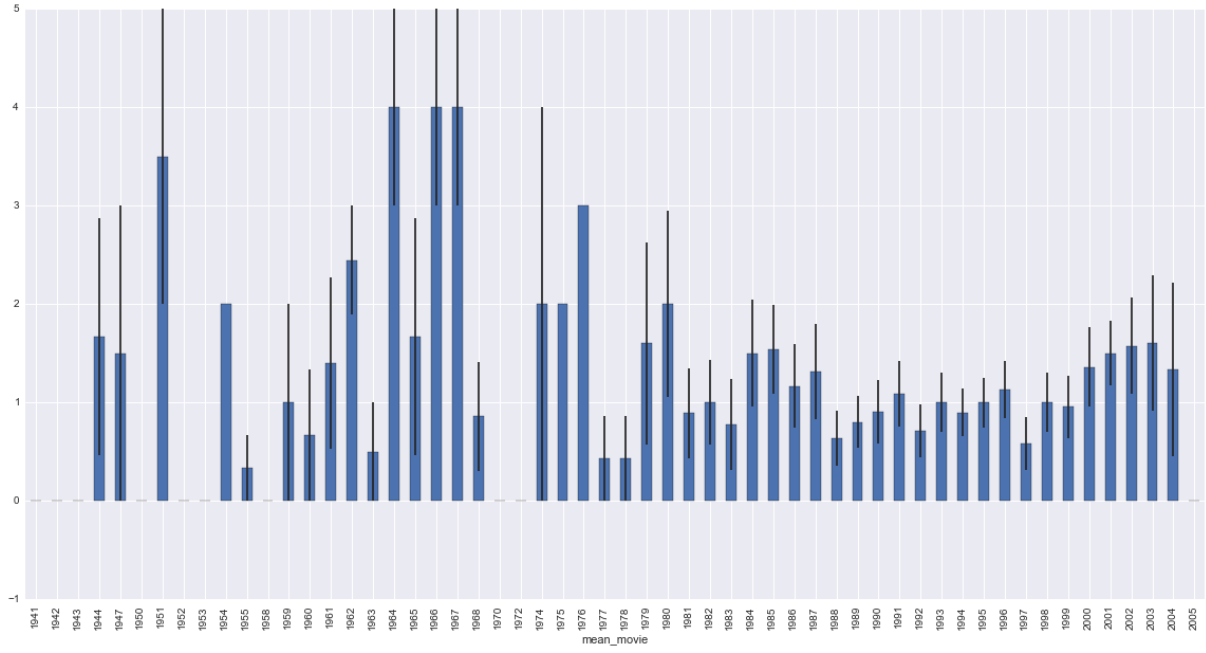
Figura 5.5: Percentual do total de filmes analisados por faixa etária



	15 - 25	26 - 35	36 - 45	46 - 60	60
Não Avaliados	68,65	27,72	2,14	0,19	1,30
Não Similar	40,57	49,71	3,43	2,29	4,00
Similar	43,57	42,32	4,15	5,81	4,15

Deste primeiro Gráfico foi possível verificar que a maioria dos participantes que não souberam avaliar o filme, i.e. usaram o recurso de PULAR, fazem parte da faixa etária de 15 a 25 anos. O que representa um comportamento já esperado para a pesquisa, visto que o filme mais atual datava de 2005. Já era esperado que pessoas dessa faixa etária não tivessem tanto conhecimento de filmes da década de 60, 70, 80 e até 90, que representa a maioria dos filmes, conforme a distribuição exibida no Gráfico presente na Figura 5.6.

Figura 5.6: Médias dos scores por ano dos filmes



Além de solicitar que o participante avaliasse um par de filmes como *similar* ou *não similar*, quando o filme era avaliado como similar, solicitou-se também que o avaliador desse uma nota dentro da escala de 1 à 5, onde **1** representava um par de filmes *pouco semelhante* e **5** um par de filmes *muito semelhante*. Esta medida chamo aqui de *score*. A distribuição das avaliações por scores são apresentadas na Figura 5.7.

Figura 5.7: Histograma dos scores correspondente aos filmes avaliados

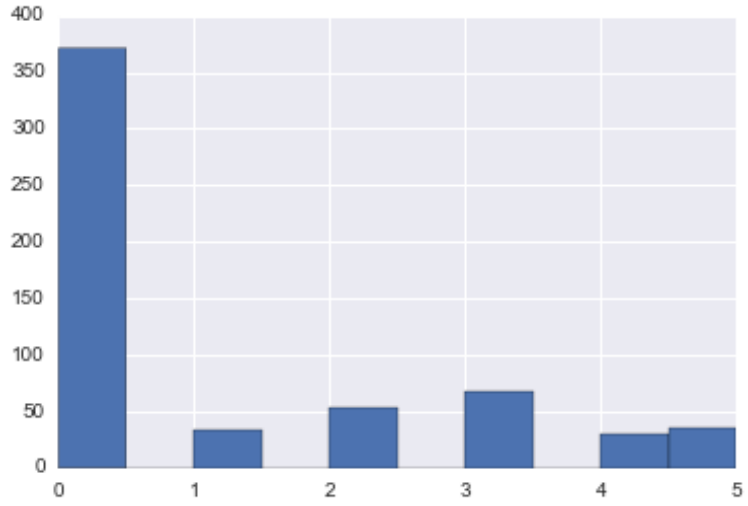
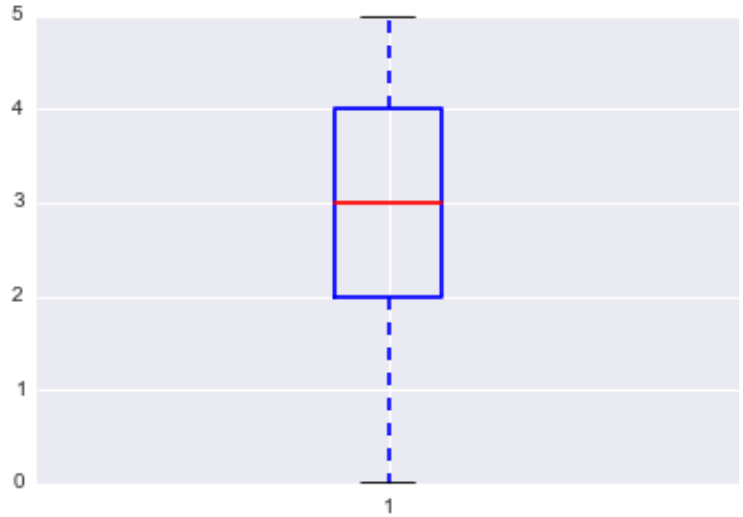


Figura 5.8: Boxplot: Scores



Na Figura 5.8 pode-se verificar que a média dos filmes avaliados como similares resultou em um score aproximado de 3. Sendo assim, dos filmes avaliados como similares pode-se dizer que, em média, eles foram considerados razoavelmente similares visto que essa média está mais próxima do limite superior do que do inferior.

5.4 Considerações Finais sobre os Resultados

Como resultado dessa análise, foi possível então verificar que 41% dos filmes foram avaliados como similares. Quanto à medida de similaridade, i.e., o quão semelhantes 2 filmes são, apresentou score médio de 3, o que pode ser visto como um resultado satisfatório pois como a variação da escala de scores foi de 1 a 5, quanto mais próximo do 5 mais similares os filmes foram considerados.

Acredito que o algoritmo LAP pode ainda ser aprimorado, principalmente na questão do tempo de execução, pois atualmente a versão otimizada só dá bons resultados ao ser executada em um cluster de computadores com vários processadores em paralelo, o que comercialmente pode ser uma barreira. Ainda, sobre a qualidade do LAP, é possível dizer que se mostrou razoável e competitivo comparado com as demais plataformas.

6 CONCLUSÃO

Dentro do contexto de sistemas de recomendação, em geral, percebe-se um cenário bastante promissor para pesquisa e principalmente para o desenvolvimento de soluções inteligentes e aplicáveis, i.e, algoritmos ou plataformas que possam ser utilizadas em benefício de pessoas e negócios, não apenas no âmbito acadêmico e experimental. Durante a execução deste trabalho, pode-se perceber uma variedade de domínios a explorar e, em especial, um grande potencial a aplicabilidade dos motores de recomendação. Acredita-se que novas tecnologias de recomendação deverão melhorar aspectos como flexibilidade para uso em diferentes domínios e a escalabilidade. Estudos nesta última área devem ser intensificados, dado o crescente aumento no volume de dados produzidos e disponibilizados diariamente na Web.

Dada então a importância dos SRs, neste estudo desejou-se ter um conhecimento adicional sobre algumas plataformas conhecidas que fazem recomendação e verificar por meios de métricas o desempenho e acurácia de cada uma delas. O objetivo deste trabalho, de análise e verificação das plataformas de recomendação por meio de um processo de benchmarking e também por um experimento com usuários, foi validar a qualidade do algoritmo de recomendação LAP. Apesar de outros trabalhos e pesquisas relacionados a este algoritmo terem apresentado bons resultados, a ideia central era analisá-lo comparativamente com plataformas mais robustas e existentes no mercado e que têm propósito de recomendação similar.

Sobre as plataformas de recomendação estudadas, pelos resultados apresentados, ficou bastante evidente que GraphLab Create tem um desempenho muito bom, além da qualidade das recomendações ser bastante relevante. Acredita-se que a arquitetura desenvolvida para esse framework contribuiu bastante para sua performance em tempos de resposta.

Já sobre o algoritmo LAP, objeto foco deste trabalho e o que motivou a pesquisa e o benchmarking, no contexto estudado, ele mostrou-se com desempenho bastante aquém do esperado. Obviamente é preciso considerar que ainda é uma tecnologia em seu estado da arte, que ainda não atingiu sua maturidade e está distante de ser usada para fins comerciais, por exemplo, principalmente pela necessidade de uso de um hardware (um cluster) com altos recursos de processador e memória RAM. Além, é claro, da necessidade adicional de torná-la integrável via API, por exemplo.

Quanto aos resultados da pesquisa, embora a avaliação tenha sido positiva, no

sentido de que a quantidade de filmes avaliados como similares foi expressiva em comparação com o conjunto total de filmes avaliados, ela não foi muito conclusiva. Acredita-se que, pelo fato da base de dados de filmes ser composta por filmes bem antigos, ocasionou que boa parte dos avaliadores não tinham conhecimento dos mesmos e necessitaram pesquisar informações na internet, o que pode tê-los conduzido a avaliações subjetivas ou mesmo superficiais.

Como possibilidade de extensões deste estudo em trabalhos futuros, cita-se o uso de bancos de dados maiores do que o utilizado nesta pesquisa, por exemplo as versões do Netflix que possuem 100 mil e 1 milhão de usuários, para se obter uma visão de como ambas plataformas trabalham a questão da escalabilidade. Sobre o algoritmo LAP, a sugestão é que a técnica do cálculo de coocorrência fosse substituída pelo cálculo de similaridade Jaccard, que com base neste estudo comparativo indicou, por meio dos valores de PPV, como sendo a melhor técnica de predição.

REFERÊNCIAS

BRUGGER, C. et al. Exploiting phase transitions for the efficient sampling of the fixed degree sequence model. In: ACM. **Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015**. [S.l.], 2015. p. 308–313.

HU, Y.; KOREN, Y.; VOLINSKY, C. Collaborative filtering for implicit feedback datasets. In: IEEE. **2008 Eighth IEEE International Conference on Data Mining**. [S.l.], 2008. p. 263–272.

JOHN, A. F. Cluster implementation for the link assessment problem finding a heuristic to estimate the number of swaps. 2015.

LINDEN, G.; SMITH, B.; YORK, J. Amazon. com recommendations: Item-to-item collaborative filtering. **IEEE Internet computing**, IEEE, v. 7, n. 1, p. 76–80, 2003.

RICCI, F.; ROKACH, L.; SHAPIRA, B. **Introduction to recommender systems handbook**. [S.l.]: Springer, 2011.

SPITZ, A. et al. Assessing low-intensity relationships in complex networks. **PloS one**, Public Library of Science, v. 11, n. 4, p. e0152536, 2016.

ZWEIG, K. A. How to forget the second side of the story: A new method for the one-mode projection of bipartite graphs. In: IEEE. **Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on**. [S.l.], 2010. p. 200–207.

APÊNDICE A — ALGORITMO PARA O CÁLCULO DO PPV

```
1
2  #include <iostream>
3  #include <fstream>
4  #include <sstream>
5  #include <string>
6  #include <cstdlib>
7  #include <stdlib.h>
8  #include <vector>
9  #include <algorithm>
10 #include <algorithm>
11
12 using namespace std;
13
14
15 int main () {
16
17   int i=0,j=0,x=0;
18   int encontrouParNoGT=0;
19
20   int indiceNumRegs=0;
21   int indiceNumRegsNoGT=0;
22
23   int *vectorMovieMain      = new int[12051];
24   int *vectorMovie          = new int[12051];
25   double *vectorScore       = new double[12051];
26
27   int *vectorMovieMain_GT   = new int[12051];
28   int *vectorMovie_GT       = new int[12051];
29
30   for (j = 0; j < 12051; j++)
31   {
32     vectorMovieMain[j]     = 0;
33     vectorMovie[j]         = 0;
34     vectorScore[j]         = 0.0;
35     vectorMovieMain_GT[j] = 0;
36     vectorMovie_GT[j]      = 0;
37   }
38
39   string line,lineGT;
40
41   ifstream myfile ("/home/tagline/datasets/output/training_data_ordenado     .csv");
42   ifstream myfileGT ("/home/tagline/datasets/big_gt_20k_pairs.csv");
43
44    // carrega movies da framework num array
45    if (myfile.is_open())
46    {
47     while (! myfile.eof() )
48     {
49
```

```
50          if(indiceNumRegsNoGT == 12051)
51          break;
52
53          getline (myfile,line);
54      std::string s(line);
55      std::string str = line;
56      std::replace( str.begin(), str.end(), ',', ' ');
57        std::istringstream ss(str);
58
59        int movieA, movieB;
60        double score;
61
62        ss >> movieA >> movieB >> score;
63
64      //cout << movieA << " - " << movieB << " : " << score << endl;
65
66      if (ss.fail())
67        {
68            // Error
69        }
70        else
71        {
72
73            vectorMovieMain[indiceNumRegs] = movieA;
74              vectorMovie[indiceNumRegs]     = movieB;
75              vectorScore[indiceNumRegs]     = score;
76
77        indiceNumRegs++;
78
79        }
80
81      }
82      myfile.close();
83
84    }
85    else cout << "Unable to open file" << endl;
86
87
88
89    // carrega movies do GT num array
90    if(myfileGT.is_open()) {
91
92     while (! myfileGT.eof() )
93     {
94
95          getline (myfileGT,lineGT);
96
97          std::string s(lineGT);
98          std::string strGT = lineGT;
99          std::replace( strGT.begin(), strGT.end(), ',', ' ');
100         std::istringstream ss_GT(strGT);
101
```

```
102         int movieA_GT, movieB_GT;
103
104         ss_GT >> movieA_GT >> movieB_GT;
105
106         if (ss_GT.fail())
107         {
108           // Error
109         }
110         else
111         {
112
113           vectorMovieMain_GT[indiceNumRegsNoGT] = movieA_GT;
114           vectorMovie_GT[indiceNumRegsNoGT]     = movieB_GT;
115
116           indiceNumRegsNoGT++;
117
118         }
119     }
120
121   }
122   else cout << "Unable to open file GT" << endl;
123
124    // para cada filme presente no arquivo de recomendação, compara-o com todos os filmes
        existentes do GT
125    for(j=0;j<=indiceNumRegs;j++) {
126
127      if(vectorMovieMain[j] > 0 && vectorMovie[j] > 0) {
128
129        for(x=0;x<=indiceNumRegsNoGT;x++) {
130
131         if ( (vectorMovieMain[j] == vectorMovieMain_GT[x] && vectorMovie[j] ==
       vectorMovie_GT[x]) ||
132              (vectorMovieMain[j] == vectorMovie_GT[x] && vectorMovie[j] ==
       vectorMovieMain_GT[x]) )
133           encontrouParNoGT++;
134        }
135      }
136
137   }
138
139   cout << "\n TOTAL DE REGS: " << indiceNumRegsNoGT << "\n";
140   cout << "\n PPV " << encontrouParNoGT ;
141   cout << "\n FIM " ;
142     return 0;
143
144 }
```