

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



FEUP

Recomendação de Conteúdos: Aplicação de Agrupamento Distribuído a Conteúdos de TV

Alexandre José Monteiro Rodrigues

Mestrado Integrado em Engenharia Informática e Computação

Orientador: João Correia Lopes (Professor Doutor)

Responsável de Acompanhamento: Telma Mota (Eng^a.)

17 de Junho de 2011

Recomendação de Conteúdos: Aplicação de Agrupamento Distribuído a Conteúdos de TV

Alexandre José Monteiro Rodrigues

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo júri:

Presidente: Luís Filipe Pinto de Almeida Teixeira (Doutor)

Vogal Externo: Maria Benedita Campos Neves Malheiro (Doutor)

Orientador: João António Correia Lopes (Doutor)

13 de Julho de 2011

Resumo

Com o crescimento explosivo de conteúdos, as pessoas sentem cada vez mais dificuldade em escolher o que é mais relevante e o que mais se adequa aos seus gostos. Os sistemas de recomendação são ferramentas que sugerem itens que os utilizadores poderão gostar e desempenham um papel importante pois permitem ajudá-los a escolher com o mínimo de esforço os conteúdos ou itens a consumir.

No entanto, apesar da crescente popularidade destes sistemas, o desenvolvimento de uma aplicação agnóstica ao domínio não é trivial. Nesta dissertação considera-se a aplicação ao domínio dos filmes e reproduz-se a aplicação a um cenário de programas de televisão, que é praticamente inexplorado na literatura. Existem alguns desafios associados à evolução dos gostos dos utilizadores e aos requisitos de tempo de resposta do sistema.

O objectivo do trabalho é estudar como melhorar a recomendação de itens de conteúdo (programas de televisão e de filmes), combinando várias técnicas e assegurando a escalabilidade da aplicação.

O estudo foi realizado usando a técnica de *clustering* MinHash que associa os utilizadores a grupos de acordo com a semelhança do conjunto de itens que viram anteriormente. Esta técnica é complementada pela técnica *Probabilistic Latent Semantic Indexing* que recorre a um modelo mistura para modelar probabilisticamente a presença de sub-populações num conjunto de observações considerado, sem que as sub-populações estejam identificadas *a-priori*.

O sistema desenvolvido é tolerante à falha, composto por vários componentes que assentam numa infra-estrutura distribuída. A arquitectura contempla pedidos de recomendação em tempo real e um processamento *offline* (usando o paradigma MapReduce) de um conjunto de observações que resulta na partição de utilizadores por comunidades (*clustering*) e na modelação probabilística que determina a afinidade dos objectos de conteúdo e dos utilizadores a um número pré-definido de classes latentes, consideradas também como *clusters*. A contabilização de observações de cada item por *cluster* permite efectuar o cálculo de *score* a cada objecto candidato a recomendação e desta forma determinar a lista ordenada de objectos a recomendar.

O trabalho realizado está enquadrado num projecto de serviços baseados em contexto da PT Inovação e a combinação de técnicas estudadas será posteriormente aplicada a um conjunto de dados do serviço de IPTV (*Internet Protocol Television*) MEO. Devido ao processo burocrático de acesso a dados confidenciais, foram usados conjuntos de dados públicos no domínio de filmes, tendo em conta a natureza dos dados do conjunto da MEO.

Os resultados obtidos pela técnica MinHash são satisfatórios e a técnica permite controlar, através de parâmetros, o número de *clusters* pretendidos e a cobertura do universo de utilizadores. Os *clusters* são usados no cálculo de recomendação e determinaram-se

quais os esquemas de partição que levam a melhores resultados para o conjunto de dados utilizado. A capacidade de resposta dos componentes *online* é também estudada, sendo identificados pontos de melhoria a serem aplicadas em produção.

Abstract

With the explosive growth of multimedia content of the past few years, people are finding it increasingly more difficult to choose what is most relevant and most suited with their tastes. Recommender systems are software tools that can suggest items that users might like and play an important role because they help people choosing items to consume.

However, despite the popularity of these kind of systems, the development of an agnostic to the application domain is not trivial. This thesis considers the current applications of recommenders to the field of films and adapts it to the television programs domain, which is virtually unexplored in the literature. There are some challenges associated with the evolution of tastes over time and requirements of users of the system response time.

The study focus on the ways to improve the recommendation of content items (television programs and films) by combining various techniques and how the implementation can scale up with the increase of problem's size.

The study was conducted using the MinHash clustering technique that is a technique that links users to groups according to the similarity of the set of items they have seen before. This technique is complemented by Probabilistic Latent Semantic Indexing, which uses a mixture model to probabilistically represent the presence of sub-populations in a set of observations. The sub-populations are not identified a priori.

The system is fault tolerant, composed of several components that are on top of a distributed infrastructure. The architecture provides real-time recommendation requests and offline processing (using the MapReduce paradigm) of a set of observations that results in a set of partitions of users (communities) and in a probabilistic model which contains the affinity of content objects and users to a predefined number of latent classes, also considered as clusters. The solution also performs the observation accounting for each user-item interaction, keeping track of the observations by cluster. The statistics gathered are used in the calculation of a score for each item candidate for recommendation. The recommendation is a list of items, ordered by the calculated score.

The work is part of a context-aware services project of the company PT Inovação and the combination of techniques will be later applied to the MEO IPTV service dataset. Due to the bureaucratic process of access to confidential data, public datasets were used in the domain films, taking into account the nature of MEO's dataset.

The MinHash results are satisfactory. The parameters of this technique provide a good control of the number of clusters generated and the cover of the clusters in the universe of users.

The clusters are used in the recommendation calculation and we can conclude which kind of partition scheme leads to better results for the dataset used. The responsiveness

of the online components is also benchmarked and we can identify the points for improvement to be applied in production.

Agradecimentos

Existem alguns agradecimentos que gostaria de fazer para aqueles que me acompanharam neste percurso e, de alguma forma, me ajudaram a realizar esta dissertação.

Aos meus pais, Amélia Monteiro e Manuel Rodrigues, pelo apoio incondicional que me deram durante este período de trabalho e ausência.

À minha namorada, Alexandra Rodrigues, pela compreensão, paciência e ajuda.

Ao meu orientador, Professor João Correia Lopes, que acompanhou o meu trabalho ao longo destes meses, marcado por vários imprevistos.

À FEUP, em especial aos professores de Engenharia Informática e Computação, pelos conhecimentos transmitidos e pela incessante exigência.

Alexandre Rodrigues

“Deep experience is never peaceful.”

Henry James

Conteúdo

1	Introdução	1
1.1	Contexto e Enquadramento	1
1.2	Motivação	2
1.3	Objectivos	3
1.4	Estrutura do documento	3
2	Sistemas de Recomendação	5
2.1	Introdução	5
2.2	Principais Desafios	7
2.3	Abordagens de Recomendação	9
2.3.1	Recomendação baseada no Conteúdo	9
2.3.2	Filtragem Colaborativa	10
2.3.3	Recomendadores Híbridos	12
2.3.4	Sumário	13
2.4	Avaliação de Recomendadores	13
2.5	<i>Clustering</i> por MinHash	15
2.6	<i>Probabilistic Latent Semantic Indexing</i>	17
2.7	Sumário	18
3	Caracterização dos Dados	19
3.1	Visão Geral	19
3.2	Fontes de Dados	20
3.3	Dados do Conteúdo	22
3.4	Dados de Comportamento	23
3.5	Sumário	24
4	Tecnologia e Arquitectura	27
4.1	Introdução	27
4.2	Tecnologias e Ferramentas	28
4.2.1	Desenvolvimento de Sistemas de Recomendação	28
4.2.2	Tecnologia de Persistência	30
4.2.3	Outras Tecnologias	30
4.3	Arquitectura	31
4.3.1	Arquitectura Lógica	31
4.3.2	Arquitectura Física	33
4.4	Sumário	35

CONTEÚDO

5	Detalhes de Implementação	37
5.1	Visão Geral	37
5.2	Servidor de Metaconteúdo	37
5.3	Servidor de Estatísticas	39
5.4	Servidor de Personalização	40
5.5	Ferramenta de Aprendizagem	42
5.5.1	MinHash	42
5.5.2	PLSI	43
5.6	Ferramenta de Avaliação	46
6	Ensaio e Avaliação	49
6.1	Técnicas de <i>Clustering</i>	49
6.1.1	Técnica MinHash	49
6.1.2	<i>Probabilistic Latent Semantic Indexing</i>	53
6.2	Qualidade de Recomendação	54
6.3	Desempenho dos Componentes	56
6.4	Conclusões	60
7	Conclusão	61
7.1	Sumário	61
7.2	Trabalho Futuro	62
	Referências	63
A	Fórmulas e Cálculos de Apoio	69
A.1	Medidas de Similaridade	69
B	Artefactos	71
B.1	Implementação das Funções de Dispersão	71
B.1.1	Função de Dispersão Linear	71
B.1.2	Função de Dispersão Polinomial	71
B.1.3	Função de Dispersão Murmur	72
B.2	Esquemas XSD	74
B.2.1	Esquema XSD de Conteúdo	74
B.2.2	Esquema XSD de Recomendação	78

Lista de Figuras

3.1	Gráfico de distribuição do número de itens visualizados por utilizador no conjunto de treino.	24
3.2	Gráfico de distribuição do número de utilizadores por objecto de conteúdo no conjunto de treino.	25
3.3	Gráfico de distribuição do número de itens visualizados por utilizador no conjunto de teste.	25
3.4	Gráfico de distribuição do número de utilizadores por objecto de conteúdo no conjunto de teste.	25
4.1	Arquitectura Lógica, representando os vários sistemas <i>online</i> e <i>offline</i> . . .	31
4.2	Arquitectura Física	33
5.1	Diagrama de actividades do Colector de Itens	38
5.2	Diagrama de actividades do Servidor de Estatísticas, accionado por pedido	39
5.3	Diagrama de actividades do Servidor de Personalização, accionado por pedido	41
5.4	Diagrama de dependências dos artefactos gerados pela implementação do algoritmo EM	45
5.5	Diagrama de actividades da Ferramenta de Avaliação.	46
6.1	Número de <i>clusters</i> gerados, variando parâmetros <i>nh</i> e <i>kg</i>	50
6.2	Gráfico de número de utilizadores pertencentes aos <i>clusters</i> , variando parâmetros <i>nh</i> e <i>kg</i>	51
6.3	Gráfico representando o número médio de utilizadores por <i>cluster</i> gerado variando parâmetros <i>nh</i> e <i>kg</i>	51
6.4	Gráfico representando o desvio padrão do número de utilizadores por <i>cluster</i> gerado variando parâmetros <i>nh</i> e <i>kg</i>	51
6.5	Gráfico representando o número de <i>clusters</i> a que um utilizador pertence, variando parâmetros <i>nh</i> e <i>kg</i>	52
6.6	Gráfico representando o desvio padrão do número de <i>clusters</i> a que um utilizador pertence, variando parâmetros <i>nh</i> e <i>kg</i>	52
6.7	Gráfico com perda empírica para um número de classes latentes iguais a 10, 25 e 50	53
6.8	Gráfico com perda empírica para um número de classes latentes iguais a 12 e 13	53
6.9	Gráfico de precisão-cobertura, no conjunto de teste NF1P, usando <i>clusters</i> MinHash com $kg = 3$	55

LISTA DE FIGURAS

6.10	Gráfico representando a medida F, no conjunto de teste NF1P, usando <i>clusters</i> MinHash com $kg = 3$	55
6.11	Gráfico de precisão-cobertura, no conjunto de teste NF1P, usando <i>clusters</i> MinHash com $kg = 4$	56
6.12	Gráfico representando a medida F, no conjunto de teste NF1P, usando <i>clusters</i> MinHash com $kg = 4$	57
6.13	Tempos de resposta do ensaio SE1, ao longo de 100 s	58
6.14	Tempos de resposta do ensaio SP1 realizado com 1000 pedidos de 100 clientes concorrentes	59
6.15	Tempos de resposta do ensaio SP2 realizado com 2000 pedidos de 100 clientes concorrentes	59

Lista de Tabelas

2.1	Lista de algumas técnicas aplicadas em sistemas de recomendação	7
3.1	Caracterização das Fontes de Dados para itens	21
3.2	Principais Informações contidas em cada fonte de dados de itens	21
3.3	Elementos semânticos do item	22
4.1	<i>Frameworks</i> de desenvolvimento de SR — Comparação entre projectos	29
4.2	<i>Frameworks</i> de desenvolvimento de SR — Comparação de funcionalidades	29

LISTA DE TABELAS

Abreviaturas e Símbolos

CB	<i>Content-based</i> , ou baseados em conteúdo, é uma das categorias de recomendadores que toma em consideração a estrutura de cada item
CSV	<i>Comma Separated Value</i> , é um formato de ficheiro onde os registos são separados por linha e os valores referentes aos atributos são separados por uma vírgula. Existem formatos análogos em que o carácter de separação é outro como, por exemplo, o caractere de tabulação
EM	<i>Expectation Maximization</i> , algoritmo iterativo e não incremental usado na construção de modelos probabilísticos, determinando nos parâmetros que aumentam a sua verossimilhança
EMA	Erro Médio Absoluto
EMQ	Erro Médio Quadrático
EMAP	Erro Médio Absoluto Percentual
EPG	<i>Electronic Programming Guide</i> , ou guia de programação electrónico, é um serviço que contém a grelha com os programas que estão agendados para transmissão. Esta informação é bastante volátil, pelo que só se considera segura nas 36 horas seguintes, a cada consulta
FC	Filtragem Colaborativa é uma das categorias de recomendadores que toma em consideração as classificações que os vários utilizadores atribuem aos itens
HDFS	<i>Hadoop Distributed Filesystem</i> é um sistema de ficheiros distribuído presente na plataforma Hadoop, que armazena partições dos ficheiros por um conjunto de computadores de uma forma redundante
HTML	<i>HyperText Markup Language</i> , linguagem de anotação usada na <i>Web</i>
HTTP	<i>Hypertext Transfer Protocol</i> é o protocolo de comunicação usado na <i>Web</i>
IMDb	<i>Internet Movie Database</i> ¹
IPTV	<i>Internet Protocol Television</i> , é o nome da tecnologia que permite a visualização de televisão, através da transmissão dos conteúdos pela Internet
OWL	<i>Web Ontology Language</i> é uma linguagem de descrição de ontologias mais poderosa que o RDF
SOAP	<i>Simple Object Access Protocol</i> é um protocolo de troca de informação estruturada usada na comunicação entre computadores por <i>Web services</i>
MinHash	<i>Minwise Independent Permutation Hashing</i> , algoritmo de <i>clustering</i>
SR	Sistema de Recomendação, ou Recomendador, é uma ferramenta que gera sugestões para um dado domínio de itens a um universo de utilizadores, conforme as suas preferências

¹<http://www.imdb.com/>

ABREVIATURAS E SÍMBOLOS

STB	<i>Set-top Box</i> é o dispositivo que descodifica o sinal digital e o envia para a televisão. Tem normalmente outros serviços associados, como a navegação num menu, <i>video-on-demand</i> (VOD) e <i>Electronic Programming Guide</i> (EPG)
VOD	<i>Video on Demand</i> , serviço <i>pay-per-view</i> de televisão em que é possível alugar vídeos sob pedido, a qualquer momento
RDF	<i>Resource Description Framework</i> é uma linguagem de descrição de ontologias
REST	<i>Representational State Transfer</i> é um estilo de arquitectura de software que define um conjunto de regras que a arquitectura deve seguir. As regras definem uma interface uniforme entre componentes e a comunicação é feita com o protocolo HTTP
XML	<i>Extensible Markup Language</i> , linguagem de anotação de dados estruturados para a definição de documentos interpretáveis pelo computador
XMPP	<i>Extensible Messaging and Presence Protocol</i> é um protocolo de comunicação usado para troca de mensagens em tempo real e para troca de informação de presença
XSD	<i>XML Schema Definition</i> , linguagem de definição de esquemas XML

Capítulo 1

Introdução

1.1 Contexto e Enquadramento

Com o crescimento explosivo de conteúdos multimédia, as pessoas sentem cada vez mais dificuldade em escolher o que é mais relevante e o que mais se adequa aos seus gostos. É normal, no dia a dia, as pessoas tomarem decisões com base em informação incompleta, como a escolha de um livro para ler. Dada a limitação humana e temporal de consumir todos os conteúdos disponíveis, as pessoas recorrem muitas vezes, quando confrontadas com opções desconhecidas, a opiniões de pessoas amigas ou de críticos especialistas.

Os actuais guias de programação de televisão, disponíveis em revistas ou em formato electrónico (EPG), estão longe de fornecer ao comum visualizador a informação necessária para a tomada de decisão.

Os sistemas de recomendação (SR) aparecem, assim, com o importante papel de ajudar o espectador a escolher com o mínimo de esforço. São essencialmente ferramentas que sugerem itens de que os espectadores poderão gostar. São actualmente utilizados em diversos domínios como compras e entretenimento e são fulcrais para a actividade de alguns negócios de sucesso conhecidos como a Amazon¹ e o Netflix².

O objectivo deste trabalho de dissertação é o estudo de várias técnicas de recomendação e a sua combinação aplicada ao domínio de programas televisivos. A dissertação enquadra-se num projecto da empresa PT Inovação que tem como objectivo fornecer um conjunto de serviços inteligentes baseados no contexto do utilizador. Pretende-se munir o actual sistema de televisão IPTV “MEO” de capacidades de recomendação tanto ao nível da grelha de programação de conteúdos, como ao nível do serviço de *Video-on-Demand* (VoD), mais conhecido por “MEO Vídeoclube”.

No ponto de vista do negócio, o SR tem várias funções:

¹<http://www.amazon.com/>

²<http://www.netflix.com/>

- aumentar o número de visualizações de conteúdos e o número de vendas de filmes, resultando numa maior facturação e resultado operacional;
- aumentar a diversidade dos filmes alugados, justificando o investimento em direitos de transmissão para filmes menos populares;
- melhorar a percepção do que o utilizador realmente procura, sendo um *feedback* importante para a decisão dos conteúdos a comprar;
- fidelizar o cliente, aumentando a satisfação nos itens que consome através de um serviço mais personalizado.

No ponto de vista do utilizador, o SR tem outras funções como a selecção de bons itens de acordo com as suas preferências, a enfatização de conteúdos relevantes da grelha, a selecção de uma boa sequência de conteúdos (à semelhança das *playlists* do Youtube³), a recomendação de uma promoção na compra de dois filmes associados e ainda a possibilidade do utilizador expressar a sua opinião através de critérios bem definidos.

Pretende-se, para além da combinação de várias técnicas, agregar dados de várias fontes de forma a enriquecer a representação dos itens de conteúdos. Uma melhor representação permitirá uma melhor abordagem baseada em conteúdo, não explorada neste trabalho.

As técnicas aplicadas são agnósticas ao tipo de conteúdo por se basearem no comportamento do utilizador face aos objectos. Isso permite a aplicação do sistema a outros domínios, efectuando o respectivo ajuste de parâmetros.

1.2 Motivação

Apesar da crescente popularidade dos sistemas de recomendação, a aplicação ao domínio de programas de televisão é ainda pouco explorada. Por se tratar de um domínio muito volátil, onde estão sempre a surgir programas novos, existem desafios no desenvolvimento de um recomendador como a escalabilidade dos algoritmos para aplicação a grandes conjuntos de dados, a proactividade do SR que determina se uma sugestão deve realmente ser apresentada ou tida em conta mesmo quando não é requisitada, a privacidade do utilizador e a utilização do contexto (por exemplo, dia da semana) na melhoria da sugestão.

O ambiente de aplicação do recomendador a desenvolver é interessante pelo facto de colocar à prova todos estes factores, não sendo triviais as decisões de concepção a tomar. O desafio acresce ao ter em conta o cenário de produção de um sistema deste tipo.

³<http://www.youtube.com/>

1.3 Objectivos

O objectivo do trabalho é o desenvolvimento de um sistema de recomendação para ser aplicado nos conteúdos transmitidos pelo operador IPTV português MEO. Estes conteúdos incluem a tradicional grelha de programação, mas também serviços *on-demand* que são transmitidos quando solicitados pelo utilizador. A actual infra-estrutura permite a interacção com o utilizador, sendo possível recolher dados de visualização.

O domínio dos programas de televisão introduz requisitos específicos no desenvolvimento do recomendador, enumerados na seguinte lista:

- **R1** — capacidade de processamento de um conjunto grande de novos programas de televisão, que varia diariamente;
- **R2** — as recomendações geradas têm uma validade curta, uma vez que a maioria dos conteúdos é transmitido num horário fixo e não pode ser revisto por *video-on-demand*;
- **R3** — os dispositivos Set-top Box (STB) usados para decodificar o sinal transmitido poderão ser utilizados por várias pessoas, sendo impossível, num dado momento, determinar quem é o utilizador.

O desenvolvimento do SR implica um estudo das técnicas de recomendação a combinar, a escolha do tipo de combinação a fazer e a construção de um sistema capaz de enriquecer a representação do item recorrendo a fontes de dados externas. A eficácia das técnicas aplicadas é avaliada através de métricas descritas na literatura.

1.4 Estrutura do documento

Para além da introdução, este relatório contém mais 5 capítulos. No capítulo 2 guia-se o leitor pelo mundo dos sistemas de recomendação, explicando factores de decisão no desenho de SR, abordagens algorítmicas e os principais problemas e desafios. Neste capítulo é também referido o trabalho relacionado com o domínio de programas de televisão e de filmes. Este capítulo contém ainda uma descrição mais detalhada das técnicas escolhidas para estudo.

No capítulo 3 são caracterizados os dados disponíveis e a forma como serão preparados e processados para efeitos de recomendação. No capítulo 4 são apresentadas as tecnologias usadas na solução e a arquitectura do sistema, seguindo-se, no capítulo 5, a descrição em pormenor da implementação dos vários componentes.

No capítulo 6 são apresentados os ensaios realizados e os resultados. Finalmente, no capítulo 7 apresentam-se as conclusões e o trabalho futuro.

Introdução

Capítulo 2

Sistemas de Recomendação

Neste capítulo introduzem-se os sistemas de recomendação, descrevendo os conceitos principais e dando a conhecer como outros tentaram resolver este tipo de problema e quais os resultados alcançados. A exposição tenta guiar o leitor para que não tenha dificuldade em ler o resto do documento.

2.1 Introdução

Sistemas de Recomendação ou Recomendadores são aplicações de *software* que implementam técnicas para gerar sugestões de itens de alguma forma úteis para o utilizador. Eles funcionam como guias personalizados que vão sugerindo objectos de interesse (itens) ao utilizador, facilitando a procura e a decisão conforme o seu gosto. Os itens podem ser conteúdos como imagens, vídeos, notícias ou objectos ontologicamente mais complexos como uma casa, um carro ou um produto. Estas aplicações têm vários domínios de aplicação que vão desde o incentivo de compra de produtos alternativos num sítio *Web* de comércio electrónico à sugestão de aplicações financeiras a um cliente de um banco, sempre de acordo com o perfil do utilizador. A sua tarefa principal é estimar, matematicamente, a avaliação que o utilizador fará de itens que nunca viu ou consumiu. Esta abordagem pode ser feita com base nas avaliações que os outros utilizadores fizeram ou com base no seu próprio histórico de consumo. Uma vez obtidas as estimativas, apresentam-se os itens com uma estimativa mais alta, os que têm uma maior probabilidade do utilizador vir a gostar.

Os dois universos de dados neste tipo de sistemas são os utilizadores e os itens. Existem três grandes categorias de sistemas de recomendação [AT05], que diferem na abordagem que fazem no cálculo da classificação estimada, isto é, na fonte de dados que utilizam para o efeito:

- **Sistemas Baseados em Conteúdo (BC)** — que tomam em consideração o conteúdo dos itens e sugerem ao utilizador itens semelhantes ao que ele seleccionou no passado. A semelhança dos itens é normalmente calculada através de técnicas de *information retrieval* [MRS08].
- **Sistemas baseados em Filtragem Colaborativa (FC)** — que tomam em consideração as avaliações dos itens pelos utilizadores e sugerem itens com base em itens avaliados de forma semelhante ou em utilizadores que avaliam de forma semelhante. Este tipo de sistemas é o mais popular de todos mas requer uma matriz de avaliação com dados suficientes para extrair alguma informação capaz de traduzir a similaridade entre itens ou utilizadores. Dependendo da variável em causa, esta categoria divide-se em duas [WdVR06]:
 - **Centrado no utilizador (*user-to-user*)** — analisando a semelhança entre utilizadores com as mesmas preferências, aconselham ao utilizador itens apreciados por outros utilizadores que os tenham avaliado de forma semelhante;
 - **Centrado no item (*item-to-item*)** — sistemas parecidos com os baseados em conteúdo, que inferem a similaridade entre itens com base nas avaliações atribuídas pelos utilizadores e recomendam ao utilizador itens semelhantes aos que ele preferiu no passado.
- **Sistemas Híbridos** — combinam as duas abordagens anteriores, tentando explorar os pontos fortes e colmatar os pontos fracos, aumentando a eficácia. A combinação entre as abordagens pode ser feita de várias formas [Bur02] e será descrita mais à frente na Secção 2.3.3.

Existe ainda outro tipo de recomendadores que são derivados destas categorias principais, como os baseados em conhecimento [Bur00, BGMS05] e os recomendadores por restrições [FFJZ06, FB08]. A diferença para os referidos consiste numa especialização para o seu domínio de aplicação [Bur02]. Por exemplo, a venda de uma máquina fotográfica, ao contrário de livros, envolve tipicamente apenas uma unidade por cada utilizador, não existindo elementos de comparação para usar abordagens colaborativas.

Quanto à forma como são tratados os dados, os sistemas de recomendação podem dividir-se em duas técnicas [AT05]:

- ***memory-based* ou *heuristic-based*** — o cálculo da estimativa é feito utilizando directamente a colecção de informação (elementos, avaliação dos utilizadores ou meta-informação dos itens, dependendo da abordagem). É aplicada uma heurística aos elementos vizinhos que, tendo em conta a sua similaridade, devolve o valor estimado.

- **model-based** — esta técnica pressupõe a construção de um modelo, usando aprendizagem automática. As predições são depois geradas através deste modelo previamente construído, o que tem vantagens ao nível do tempo de resposta. É habitual fazer-se uma construção do modelo *offline* e depois, com base no modelo, fazerem-se recomendações *online*.

Uma lista dos algoritmos normalmente adoptados para cada técnica e abordagem poderá ser consultada na Tabela 2.1 (retirada de [AT05]).

Tabela 2.1: Lista de algumas técnicas aplicadas em sistemas de recomendação

Abordagem	Técnicas	
	<i>memory-based</i>	<i>model-based</i>
Baseado em Conteúdo	• Peso Term Frequency – Inverse Document Frequency (TF-IDF) [MRS08] • <i>Clustering</i>	• Classificadores Bayesianos • <i>Clustering</i> • Árvores de Decisão • Redes Neurais Artificiais
Filtragem Colaborativa	• Cálculo de Vizinhança (através de uma métrica de similaridade: distância angular ou por alguma correlação) • <i>Clustering</i> • Teoria de Grafos	• Classificadores Bayesianos • <i>Clustering</i> • Árvores de Decisão • Redes Neurais Artificiais • Regressão Linear • Modelos probabilísticos

Nas secções que se seguem são enunciados os principais desafios no desenvolvimento de recomendadores (Secção 2.2). Segue-se uma secção dedicada às várias abordagens (Secção 2.3). São ainda descritas quais as metodologias e métricas usadas para a avaliação de desempenho de SR (Secção 2.4) e, por fim, duas secções dedicadas às técnicas de *clustering* estudadas: MinHash (Secção 2.5) e *Probabilistic Latent Semantic Indexing* (Secção 2.6).

2.2 Principais Desafios

O desenvolvimento de sistemas de recomendação tem alguns desafios associados, muitos dos quais são problemas inerentes à própria abordagem. São eles:

- **Arranque a Frio** — Também conhecido por *cold-start problem*, este é um problema comum aos recomendadores e acontece quando surgem novos itens ou novos utilizadores e nada se conhece sobre eles. Este caso sente-se mais em abordagens de FC, onde não há avaliações associadas a estas novas entradas. Nos sistemas BC, apenas ocorre com o aparecimento de novos utilizadores. Como estes sistemas exploram as características dos itens, os novos itens poderão ser recomendados a alguém que tenha preferência por itens semelhantes.
- **Dispersão** — Este problema resulta da existência de poucas avaliações dos utilizadores relativamente ao número de itens existentes. Isto leva a uma cobertura reduzida, impossibilitando uma aplicação bem sucedida de estratégias de FC. A solução comum para este problema é a redução de dimensão da matriz de *ratings* [SKKR00], através de factorização de matrizes (por exemplo, *singular value decomposition* [ZWF⁺05]), análise semântica entre itens (*latent semantic analysis*) ou outras técnicas.
- **Avaliações fraudulentas** — Este problema afecta apenas os CF, sobretudo os *memory-based*, e ocorre quando os utilizadores atribuem avaliações falsas aos itens. Isto poderá ser uma acção de ataque premeditado contra o sistema ou então um comportamento reactivo para proteger a privacidade do utilizador, tentando desta forma baralhar as predições que o sistema produz. Existem vários tipos de ataque, como criar perfis de utilizador que avaliem o item a promover conjuntamente com os mais conhecidos, de forma a que estes fiquem, de alguma forma, associados (*bandwagon attack*). Em [MBS06, MBBW07, VRY09], são descritas algumas soluções para aumentar a robustez de recomendadores colaborativos.
- **Escalabilidade** — Consiste na capacidade do SR processar a grande, e sempre crescente, quantidade de dados. Este problema afecta sobretudo as técnicas *memory-based*. Uma solução será optar por uma técnica *model-based*, em que um modelo é construído previamente à geração de recomendação. A construção do modelo é independente, podendo ser realizada *offline*. Existem algumas soluções para este problema [GM05, SKKR02] que, em geral, envolvem a paralelização do processamento na construção do modelo ou envolvem a actualização incremental da representação das pontuações em sistemas colaborativos [PRPT05].
- **Distribuição** — Consiste em implementar o recomendador de forma distribuída, de modo a ser organicamente escalável à medida que o problema cresce. A grande maioria dos recomendadores adopta a tradicional arquitectura cliente-servidor, onde o cliente pede sugestões. Isto tem a falta de flexibilidade subjacente de sistemas centralizados. Outras questões, como a existência de um repositório único com

a informação dos utilizadores [BEKR07, SPTH09, WPLR06] são uma ameaça à privacidade e constituem uma fraqueza (*single point of failure*).

- **Incompletude dos Itens** — Ocorre quando não existe forma de extrair informações suficientes do conteúdo. Isto é um problema que ocorre meramente em sistemas BC e em domínios com dados multimédia, onde muitas vezes não há conteúdo textual associado. Uma solução possível é permitir aos utilizadores a atribuição de *tags* ao conteúdo [GH05].
- **Sinonímia e Polissemia** — Este problema manifesta-se nos sistemas BC e reside nos elementos usados para identificar o item. Ocorre sinonímia quando existe mais que uma palavra é usada para descrever o mesmo tipo de item e polissemia quando a mesma palavra é usada para se referir a itens diferentes. No domínio da televisão isto poderá ser um problema no seguinte cenário: o nome e a descrição do noticiário de diferentes canais poderá não ter a palavra “jornal” destacada, como é o caso do “Telejornal” da RTP1 ou o “Edição das Dez” da TVI24.
- **Utilizadores atípicos** — Este problema deve-se à existência de utilizadores que não têm uma classificação consistente. Em abordagens de FC, poderão existir recomendações com má qualidade devido à correlação feita com os restantes utilizadores.

2.3 Abordagens de Recomendação

Existem duas abordagens distintas para a geração de recomendações: a primeira, baseada em conteúdo, foca-se na constituição do item, aconselhando itens semelhantes; a segunda, conhecida por filtragem colaborativa, é baseada no comportamento dos espectadores, aconselhando itens visualizados por utilizadores com um histórico de interacção semelhante.

2.3.1 Recomendação baseada no Conteúdo

Os primeiros recomendadores, surgidos na década de 90 eram baseados em conteúdo (*content-based*) (BC). A sua tarefa é identificar itens semelhantes àqueles de que o utilizador gostou no passado, analisando-os e construindo um perfil de interesses do utilizador com base nas características dos itens. Novos objectos de conteúdo são sugeridos se corresponderem aos atributos presentes no perfil construído do utilizador [Mla99].

Por ser centrado no perfil construído, este paradigma tem algumas vantagens em relação à recomendação por filtragem colaborativa, pois não necessita das avaliações dos vários utilizadores para calcular as predições. Consegue também recomendar novos objectos de conteúdo introduzidos no sistema, ainda que nunca tenham sido pontuados. Por outro lado, não considera outros gostos para além dos anteriormente explorados. Isto

faz com que sejam, em geral, sistemas sobreespecializados e rígidos, uma vez que estão limitados às preferências anteriores e não promovem a espontaneidade (*serendipity*) das sugestões. A forma actual de ultrapassar esta limitação é a classificação por parte da comunidade com *tags* associadas aos conteúdos [CSA⁺08, DI06, GH05, dGLSB08]. Esta classificação enriquece a representação do objecto de conteúdo, facilitando o processo de selecção de itens com conteúdos semelhantes.

Sistemas de recomendação BC *model-based* são constituídos por três componentes distintos [LGS11]: um analisador de conteúdos, responsável por extrair os atributos relevantes dos novos itens; um componente de aprendizagem, que vai construindo o perfil do utilizador conforme os atributos dos itens que ele selecciona; e um componente selector de conteúdos, que agrega e sugere os novos itens introduzidos que sejam, de alguma forma, compatíveis com o perfil criado. No caso do recomendador usar técnicas *memory-based* comporta apenas o analisador de conteúdos. Este é o componente principal deste tipo de sistemas pois o seu objectivo é a extracção de características e o seu pré-processamento. No domínio da televisão e filmes, a natureza dos dados não permite a extracção de características, sendo usada meta-informação [HPK10] associada ao conteúdo, como a sinopse. Há vários casos (sistema PTV [SC00], Movies2GO [MJSS01], CinemaScreen [SA06], INTIMATE [MKP03]) que exploram a sinopse, utilizando técnicas de *Information Retrieval* (IR) [MRS08]. A sinopse e a informação textual adicional, como a lista de actores, são processadas. São eliminadas *stop-words* e é aplicado *stemming* (redução de palavras derivadas à sua raiz morfológica) para normalizar conjugações verbais e plurais. Por fim, associa-se ao conteúdo um vector de palavras, usando o peso TF-IDF [MRS08].

Em [BFPAGS⁺08], os autores recorrem a ontologias para a descrição de programas de televisão. O conhecimento é representado em OWL e são usadas técnicas de inferência para comparar as preferências dos utilizadores com os itens. O uso de ontologias enriquece a representação do item, sendo possível inferir conceitos e relações a partir de outros conhecidos.

2.3.2 Filtragem Colaborativa

Os sistemas de recomendação baseados em filtragem colaborativa têm como parâmetro de entrada uma matriz de *ratings*. A matriz de avaliações, definida por R , é uma matriz Utilizadores U x Itens I , em que $\forall i \in U, j \in I, R_{i,j} \in S, S = \{0, \dots, 5\}$. Este tipo de sistemas ultrapassam algumas limitações dos sistemas BC. Como não se focam no conteúdo em si, mas sim na perspectiva que os utilizadores têm do conteúdo, eles permitem a sugestão de itens de qualidade diferentes dos previamente consumidos pelo utilizador, se existir uma vizinhança que tenha mostrado interesse. Esta abordagem social é mais parecida com o habitual passa-palavra e é bem sucedida em muitos domínios de aplicação como, por exemplo, a Amazon [LSY03].

Existem duas classes de técnicas de filtragem colaborativa: as técnicas *memory-based* (nesta abordagem também chamadas de técnicas de vizinhança), que utilizam directamente as avaliações disponíveis para gerar previsões para itens nunca vistos; as técnicas *model-based*, que usam os dados para a construção prévia de um modelo, usando técnicas de aprendizagem ou extracção de características, que depois será usado para determinar recomendações.

As técnicas de vizinhança incluem *clustering* (agrupamento de entidades com alguma característica em comum) de utilizadores ou itens semelhantes e algoritmos de *k-Nearest-Neighbour*. O cálculo da semelhança aplica uma métrica de similaridade a dois vectores considerados. As métricas mais comuns (ver Anexo A para uma colecção de métricas), tanto a nível de resultado, quanto a nível de complexidade computacional, são a diferença angular (ver Equação A.1) e o coeficiente de correlação de Pearson (ver Equação A.2, referente à similaridade de utilizadores).

Nas técnicas centradas no utilizador, estima-se o interesse do utilizador $u \in U$ num item $i \in I$ nunca visto anteriormente usando as avaliações que outros utilizadores (vizinhança) deram a i . A vizinhança é constituída por utilizadores com padrões de classificação semelhantes aos do utilizador u ; a vizinhança de u é, então, o conjunto dos utilizadores v cujas avaliações aos itens avaliados por u e v estão mais correlacionadas com as avaliações de u . Por outro lado, as técnicas centradas no item estimam a avaliação que o utilizador u irá atribuir a um item i tendo como ponto de partida as avaliações de u a itens semelhantes a i . Nestes casos, dois itens são considerados semelhantes se vários utilizadores do sistema tiverem avaliado ambos de forma semelhante. As técnicas centradas no item são, em geral, mais eficazes do que as centradas nos utilizadores [AT05, LSY03].

As técnicas de vizinhança são bastante afectadas pela falta de avaliações, recorrendo-se a técnicas de redução de dimensão [BKV07, SKKR00] ou a grafos. A redução produz uma representação compacta das interacções entre utilizadores e itens, representando apenas as características mais significativas. As técnicas de grafos utilizam as relações transitivas dos dados, analisando as relações entre utilizadores ou itens que não estão “ligados directamente”, gerando recomendações mais espontâneas. Existem dois tipos de abordagem nas técnicas de grafos [FPRS07]: as que usam a proximidade entre os itens e os utilizadores para determinar a avaliação e as que usam a proximidade entre dois itens ou dois utilizadores para determinar a semelhança.

As técnicas *model-based* consideram as avaliações para construir um modelo preditivo que é treinado com os dados disponíveis. Este modelo é, posteriormente, usado no processo de recomendação. A construção do modelo analisa as interacções entre os utilizadores e os itens e modela-as representando características latentes como a preferência de um determinado tipo de filmes. Existem várias técnicas para a construção de modelos: classificadores bayesianos [BHK98], decomposição em factores latentes (*Latent Dirichlet Allocation* [BNJ03], *Latent Semantic Analysis* [Hof03]) e redes neuronais [SMH07]. Es-

tas técnicas poderão usar directamente a similaridade entres os itens e os utilizadores para a construção do modelo (como nas *memory-based*) ou aplicar algoritmos de factorização de matrizes ou extracção de características escondidas.

Comparando as técnicas de vizinhança com as técnicas de construção de modelo, as primeiras são mais simples, eficientes, estáveis e, embora não proporcionem resultados tão bons, promovem a espontaneidade das sugestões que apresentam [GSK⁺99]. No entanto, o uso de um modelo permite otimizar muito a recomendação em tempo real [LSY03], pré-processando os dados e incluindo no modelo a dinâmica temporal [Kor09], o que traduz as alterações de tendências dos utilizadores e a popularidade dos itens ao longo do tempo.

2.3.3 Recomendadores Híbridos

Nenhuma das abordagens anteriormente apresentadas é perfeita. De forma a ultrapassar as limitações de cada abordagem e obter as respectivas vantagens, são adoptadas abordagens híbridas. O sucesso da combinação de várias técnicas e abordagens é visível na competição de recomendadores de filmes, promovida pela empresa Netflix. O objectivo da competição é fazer um sistema de recomendação capaz de superar o CinemaScreen [SA06], recomendador da empresa Netflix, em pelo menos 10%. As equipas mais bem posicionadas combinaram múltiplas técnicas e a vencedora alega ter utilizado 107 técnicas diferentes¹, embora a combinação de 3 técnicas distintas e cuidadosamente escolhidas já gerasse resultados muito próximos do final.

A combinação (*ensemble*) de vários sistemas pode ser feita de várias formas [Bur02]:

- **Ponderação** — em que o resultado é uma lista de itens cuja avaliação prevista é uma combinação linear das avaliações previstas pelos vários sistemas combinados, usando um coeficiente de ponderação para cada um.
- **Escolha** — em que a situação e a informação condicionam qual será o motor de recomendação aplicado. Uma condicionante poderá ser o género de conteúdo que se quer recomendar (e.g. desporto).
- **Misto** — é feita uma junção dos conjuntos de itens gerados pelos vários motores de recomendação. Um exemplo é o PTV [SC00], onde os itens recomendados com base no conteúdo têm precedência sobre os gerados por filtragem colaborativa.
- **Combinação de características** — consiste em aumentar o conjunto de características do item com as avaliações dos utilizadores e aplicar motores baseados em conteúdo. Experiências deste tipo aumentam a recuperação mas não a precisão (ver métricas de avaliação na Secção 2.4).

¹<http://www.research.att.com/~volinsky/netflix/ProgressPrize2007BellKorSolution.pdf>

- **Cascata** — consiste em usar um motor para gerar uma lista de itens candidatos e usar outro motor para refinar esta lista de candidatos.
- **Metanível** — o modelo do utilizador construído por um dos motores de recomendação é usado como *input* de outro motor de recomendação *model-based*. Difere do cascata, em que se usa o resultado de um como *input* de outro motor, para efectuar refinação.

2.3.4 Sumário

São consideradas duas técnicas de filtragem colaborativa para o domínio de aplicação do trabalho. As técnicas de filtragem colaborativa são *model-based* e foram com sucesso implementadas no sistema de recomendações do Google News [DDGR07], portal de agregação de notícias. A primeira consiste na segmentação dos utilizadores em comunidades através de uma técnica de *clustering* designada *MinHash* (ver secção 2.5). A segunda consiste na modelação das preferências num modelo mistura que modela probabilisticamente a afinidade de utilizadores e itens com um conjunto de variáveis latentes de tamanho definido *a priori* (ver secção 2.6). O modelo é treinado usando o algoritmo *Expectation Maximization* que vai iterativamente ajustando os valores respectivos à afinidade com as classes até convergir.

Os grupos, também referidos como *clusters*, criados por ambas as técnicas particionam o conjunto de utilizadores em comunidades de utilizadores. A técnica *Probabilistic Latent Semantic Indexing* (PLSI) determina também a afinidade dos itens aos grupos, podendo interpretar tal relação como a afinidade dos objectos de conteúdo a um determinado tópico de interesse.

2.4 Avaliação de Recomendadores

Os sistemas de recomendação são ferramentas que tentam determinar qual a melhor recomendação para um utilizador. O conceito de boa recomendação pode ser bastante subjectivo. Nesta secção são apresentadas algumas formas de avaliar recomendadores quantitativamente.

Existem várias formas de conduzir o processo de avaliação e várias variáveis a medir como a diversidade e a novidade, que poderão, dependendo do cenário, representar melhor ou pior a satisfação do utilizador. Uma forma de avaliar é realizando testes binários em que os utilizadores indicam se gostam ou não da recomendação. As métricas mais comuns neste tipo de teste são a precisão (*precision*) e cobertura (*recall*) [MRS08], que são também aplicadas em IR a resultados de pesquisa. Numa lista de recomendações, a precisão (Equação 2.1) é a percentagem de itens recomendados no universo de todos os

itens que os utilizadores gostaram e a cobertura (Equação 2.2) é o rácio dos itens considerados relevantes pelo utilizador que foram apresentados como recomendação. Existem outras métricas de desempenho, como a medida F (Equação 2.3).

$$\text{Precisão} = \frac{|\{\text{itens interessantes}\} \cap \{\text{itens recomendados}\}|}{|\{\text{itens recomendados}\}|} \quad (2.1)$$

$$\text{Cobertura} = \frac{|\{\text{itens interessantes}\} \cap \{\text{itens recomendados}\}|}{|\{\text{itens interessantes}\}|} \quad (2.2)$$

$$F = 2 \cdot \frac{\text{Precisão} \cdot \text{Cobertura}}{\text{Precisão} + \text{Cobertura}} \quad (2.3)$$

Estas métricas têm alguns problemas em experiências *offline* pois os utilizadores só vão dar *feedback* explícito para um conjunto de itens que conhecem e que são da sua preferência. Estas métricas falham também em representar factores como a novidade e a espontaneidade. Para melhor representar a diversidade sugere-se em [SG11] o índice de Gini, que representa quão desigualmente é que itens diferentes são escolhidos (i.e. vistos, no caso da televisão) pelos utilizadores, quando recomendados. Se cada item i teve uma proporção de escolhas $p(i)$ quando apresentado como recomendação, o índice de Gini é dado pela Equação 2.4, onde $i_1, \dots, i_n$ é a lista ordenada com $p(i)$ crescente. O índice de Gini igual a 0 significa que é escolhido sempre com a mesma frequência por todos os utilizadores; quando igual a 1 significa que é sempre escolhido.

$$\text{Gini} = \frac{1}{n-1} \cdot \sum_{j=1}^n (2j-n-1)p(i_j) \quad (2.4)$$

$$\text{Shannon} = \sum_{i=1}^n p(i) \log p(i) \quad (2.5)$$

Outra métrica que representa a distribuição da desigualdade é a entropia de Shannon [SG11] (Equação 2.5), que é igual a 0 se existe um item que é sempre recomendado ou igual a $\log n$ se existirem n itens que são sempre recomendados ou escolhidos de forma igual.

É plausível afirmar que um recomendador perfeito é aquele capaz de adivinhar exactamente o grau de afinidade entre um utilizador e um item e materializar isso num valor de avaliação. Outra forma de avaliar o desempenho do sistema de recomendação é medir a precisão com que o recomendador estima os *ratings*. Este processo é normalmente feito dividindo o conjunto de itens em dois conjuntos menores: o de teste e o de validação. O recomendador calcula estimativas de avaliação $\hat{r}_{ui}$ para o conjunto de validação Υ de pares utilizador-item (u, i) para os quais r_{ui} é conhecido (mas omitido), usando os dados do conjunto de teste τ . Com o conjunto de validação calcula-se, através de uma métrica,

a distância da estimativa calculada ao valor conhecido.

Existem várias métricas usadas. As mais populares são o Erro Médio Absoluto (EMA) (Equação 2.6), Erro Médio Quadrático (EMQ) (Equação 2.7) e Erro Médio Absoluto Percentual (EMAP) (Equação 2.8) [HKTR04].

$$\text{EMA} = \sqrt{\frac{1}{|\Upsilon|} \sum_{(u,i) \in \Upsilon} |\hat{r}_{ui} - r_{ui}|} \quad (2.6)$$

$$\text{EMQ} = \sqrt{\frac{1}{|\Upsilon|} \sum_{(u,i) \in \Upsilon} (\hat{r}_{ui} - r_{ui})^2} \quad (2.7)$$

$$\text{EMAP} = \sqrt{\frac{1}{|\Upsilon|} \sum_{(u,i) \in \Upsilon} \left| \frac{\hat{r}_{ui} - r_{ui}}{r_{ui}} \right|} \quad (2.8)$$

O EMQ evidencia mais a diferença do erro quando comparado com o EMA. O EMAP é a percentagem de erro do valor estimado face ao valor de validação conhecido.

2.5 Clustering por MinHash

O MinHash [Bro97, Coh97], abreviatura para *Minwise Independent Permutation Hashing*, é uma técnica de *clustering* probabilístico usada para a detecção de sequências quase duplicadas. É uma técnica de *hashing* sensível à localização dos elementos, assegurando uma maior colisão da função de dispersão quando aplicada a conjuntos de entrada semelhantes.

Esta técnica é usada no domínio da filtragem colaborativa para detectar quais os utilizadores que consumiram conjuntos de itens semelhantes, associando quaisquer dois utilizadores u_i, u_j ao mesmo *cluster* com uma probabilidade proporcional à intersecção dos seus históricos de consumo (I_i, I_j) .

Esta técnica apresenta vantagens face a outras métricas de similaridade de conjuntos como, por exemplo, o coeficiente de similaridade de Jaccard (equação A.3), pois tem execução de ordem linear e não tem de comparar todos os pares de conjuntos.

Ordenando aleatoriamente os índices de cada conjunto (permutação), define-se para cada um uma assinatura que consiste no índice do primeiro elemento original na nova ordem. A probabilidade de dois conjuntos terem a mesma assinatura é proporcional à sua similaridade (Equação A.3). Para reduzir a probabilidade de falsos positivos e de falsos negativos, calculam-se várias assinaturas de várias permutações diferentes, resultando numa quantidade de falsos positivos muito baixo e de falsos negativos nulo. Assim, sendo h uma função de dispersão, define-se $h_{\min}(A)$ como sendo o elemento $a \in A$ com o valor mínimo de $h(a)$. Quando aplicado várias vezes para permutações aleatórias escolhidas de forma uniforme, $Pr[h_{\min}(A) = h_{\min}(B)] = \text{Jaccard}(A, B)$.

Desta forma, se A for um conjunto de elementos binários, define-se $h(A)$ como o conjunto das *hashes* produzidas pela aplicação de h a cada elemento de A , i.e. $\{h(a_1), h(a_2), \dots, h(a_n)\}$. Para determinar se $\min(h(A))$, valor mínimo da função aplicada aos vários elementos de A , é um descritor estável do conjunto, consideramos o seu valor quando um elemento x é inserido e removido de A .

Se x for removido então $\min(h(A))$ apenas irá mudar se $h(x) = \min(h(A))$. A probabilidade de cada elemento de A ter o valor de *hash* mínima é igual a $\frac{1}{|A|}$.

De igual modo, se x for acrescentado a A , então $\min(h(A))$ será diferente apenas se $h(x) < \min(h(A))$. Neste caso a probabilidade de cada elemento ter o valor mínimo da função de dispersão é igual a $\frac{1}{|A|+1}$. Assim pode-se afirmar que $\min(h(A))$ é um descritor estável de um qualquer conjunto A .

Considerando agora dois conjuntos distintos, A e B , se um elemento produzir o valor mínimo de h em ambos os conjuntos, então produzirá o valor mínimo na sua união, conforme a premissa da Equação 2.9.

$$\min(h(A)) = \min(h(B)) \text{ iif } \min(h(A \cup B)) = \min(h(A)) = \min(h(B)) \quad (2.9)$$

Seja x o elemento de $A \cup B$ que produz o valor mínimo da função de dispersão. A probabilidade de A e B terem o mesmo mínimo de h é equivalente à probabilidade de x estar simultaneamente em A e em B . Como a probabilidade de qualquer elemento de $A \cup B$ ser o mínimo é igual, a probabilidade de $x \in A \wedge x \in B$ é igual a $\frac{|A \cap B|}{|A \cup B|}$ (Equação A.3).

Segue-se um exemplo da aplicação do algoritmo MinHash usando o histórico de visualização representado pela matriz da Equação 2.10, onde as colunas correspondem a itens e as colunas a três utilizadores distintos.

$$\begin{array}{c} \text{utilizador A} \quad \text{utilizador B} \quad \text{utilizador C} \\ \begin{array}{l} \text{item 1} \\ \text{item 2} \\ \text{item 3} \\ \text{item 4} \\ \text{item 5} \end{array} \left(\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array} \right) \end{array} \quad (2.10)$$

Definindo duas funções de dispersão distintas, $h(x) = x \bmod 5$ e $g(x) = 2x + 1 \bmod 5$, obtemos os seguintes valores de *hash* para o conjunto $W = \{1, 2, 3, 4, 5\}$: $h(W) = \{1, 2, 3, 4, 0\}$ e $g(W) = \{3, 0, 2, 4, 1\}$. Assim, sendo a assinatura de cada utilizador dado pelo valor mínimo de *hash* h e g dos itens visualizados, respectivamente, obtemos seguintes assinaturas: $\text{sig}(A) = 1, 2$, $\text{sig}(B) = 0, 0$ e $\text{sig}(C) = 2, 0$. Como podemos observar o utilizador B e C fazem parte de um grupo em comum, dada por uma das funções (g). A implementação do algoritmo, descrita na Secção 5.5.1, tem como parâmetros de entrada nh , que define o número de funções de dispersão a aplicar, e o parâmetro kg , que define o número de

hashes presentes na assinatura única de cada *cluster*. Neste caso, com $kg = 2$ obtemos três grupos distintos: 1-2, 0-0 e 2-0; com $kg = 1$ obtemos igualmente três *clusters* (0, 1 e 2), embora o utilizador B e C façam parte do mesmo grupo (0). Assim, o parâmetro kg influencia a especificidade de cada grupo, definindo se um grupo tem utilizadores mais parecidos ou menos parecidos.

2.6 Probabilistic Latent Semantic Indexing

A *Probabilistic Latent Semantic Indexing* (PLSI) é uma técnica criada por Hofmann [Hof04] que consiste na construção de um modelo mistura que modela utilizadores e objectos de conteúdo como variáveis aleatórias, tomando valores de cada universo respectivo. A sua relação é então capturada construindo um modelo mistura a partir da sua distribuição conjunta. Introduce-se um conjunto de L variáveis Z que retêm a relação entre itens e objectos de conteúdo, que podem ser interpretadas como comunidades de utilizadores ou tópicos de interesse (comunidades de itens).

Definindo formalmente, seja $u \in U$ um utilizador e $s \in S$ um objecto de conteúdo. A Equação 2.11 representa a afinidade de um item com um utilizador, de acordo com um conjunto de parâmetros θ e é dada pela aplicação da regra da cadeia das probabilidades condicionadas, para toda a extensão de classes latentes z .

$$p(s|u; \theta) = \sum_{z=1}^L p(s|z)p(z|u) \quad (2.11)$$

É construído um modelo de forma a determinar o θ que aumente a verossimilhança total do modelo, representando o conjunto de probabilidades $p(s|z)$ e $p(z|u)$ correctas que aumentam a afinidade total para todos os pares observados usados no processo de treino do modelo.

O modelo é treinado a partir do algoritmo *Expectation Maximization* (EM) [DLR77]. Este método para estimar os parâmetros θ de um modelo baseia-se na criação de estimativas de afinidade $p(s|u)$ para todos os pares utilizador-item observados. Este é o passo de expectativa ou passo E. A criação das estimativas é feita calculando as probabilidades suavizadas para os pares observados.

Após o cálculo das estimativas, dá-se início ao passo de maximização ou passo M. Neste passo é calculado o conjunto θ , contendo os valores que maximizam a verossimilhança do modelo. Assim, as distribuições de probabilidade condicional $p(s|z)$ e $p(z|u)$ são substituídas pelas probabilidades suavizadas do passo E.

O algoritmo EM é um algoritmo iterativo, não incremental, obrigando a um treino para objectos de conteúdo novos ou novos utilizadores. Desta forma, a sua aplicação é

feita *offline* periodicamente e os parâmetros θ são usados *online* representando o grau de pertença de um item ou um utilizador a uma classe latente z (*cluster*).

2.7 Sumário

Ao longo deste capítulo caracterizaram-se os sistemas de recomendação, focando os problemas comuns e as alternativas que podem ser tomadas para a sua resolução. São apresentadas duas formas de abordar o problema consoante os dados disponíveis: as que exploram a representação dos itens para recomendar outros itens semanticamente semelhantes e as que tratam as interações (visualizações) para agrupar itens e utilizadores em grupos de interesse (filtragem colaborativa).

Quanto à técnica aplicada, pode ser *memory-based*, se os dados forem tratados directamente no cálculo de recomendação, ou *model-based*, se os dados forem usados para construir um modelo, que será posteriormente usado no cálculo. Optou-se pelas técnicas *model-based* pelo facto dos modelos serem uma representação mais simples dos dados brutos e pelos desafios de escalabilidade sentidos quando a quantidade de dados a processar é grande.

São dados vários exemplos da literatura para os diferentes tipos de sistema, sendo a maioria aplicados ao domínio de filmes. Registam-se poucas aplicações ao domínio deste trabalho: a televisão.

Detalham-se duas técnicas de agrupamento (*clustering*), MinHash e PLSI, ambas agnósticas face ao domínio de aplicação, para particionar o universo de utilizadores em comunidades. Estas técnicas são as eleitas para a solução adoptada neste trabalho. A escolha foi baseada na escalabilidade destas técnicas, no seu agnosticismo e na sua aplicação bem sucedida no portal agregador de notícias *Google News* [DDGR07].

Capítulo 3

Caracterização dos Dados

Neste capítulo é feita a caracterização dos dados disponíveis para os ensaios e dos dados alvo para os quais o sistema será aplicado. Distinguem-se duas naturezas distintas de dados: os dados de conteúdos e os dados de comportamento dos utilizadores.

3.1 Visão Geral

O sistema desenvolvido estava inicialmente planeado para tratar dados da plataforma de IPTV MEO. No entanto, devido ao processo burocrático na obtenção de acesso, não foi possível realizar os ensaios com este conjunto. Esta lacuna foi colmatada recorrendo a um conjunto de dados do domínio de filmes, tentando reproduzir com este a experiência da televisão.

A experiência de televisão difere do domínio de filmes por limitar a visualização dos conteúdos à oferta presente numa grelha de programação. Esta dinâmica não permite ao utilizador a liberdade de visualizar tudo aquilo que quer a qualquer momento. Normalmente, os conteúdos na televisão não se repetem ou repetem-se com uma periodicidade baixa, pelo que o ciclo de vida de cada conteúdo é pequeno e as recomendações têm um prazo de validade de apenas alguns minutos, consoante a duração dos conteúdos. A isto acresce o facto de o número de itens diferentes ser alto em relação ao número de itens.

O conjunto de dados MEO, ao qual não foi possível ter acesso, é constituído por pares utilizador-conteúdo para conteúdos visualizados durante mais de 30s. O conjunto contém cerca de dez mil utilizadores distintos e contém o histórico de quatro meses.

Apesar das técnicas adoptadas no sistema serem agnósticas face à natureza dos dados por considerarem apenas o comportamento dos utilizadores, o conjunto de dados usado nos ensaios reproduz um cenário de televisão simulado e é descrito em pormenor na Secção 3.4.

O conjunto de dados do domínio de filmes considerado foi o *dataset* do Netflix por ser um dos maiores a nível de dimensão, totalizando cem milhões de observações, e por ser o mais utilizado na literatura para validação das várias técnicas.

Apesar do formato dos dados ser completamente compatível com o serviço MEO Vídeoclube, sistema de vídeo por pedido (*video-on-demand*), o comportamento dos utilizadores poderá não ser análogo pelo facto do sistema Netflix ser *flat rate*, onde os utilizadores pagam uma subscrição e têm liberdade total de ver os conteúdos que desejam, e o MEO Vídeoclube ser *pay-per-view*, onde os utilizadores pagam pela visualização de cada conteúdo.

O domínio da televisão é semelhante ao *flat rate* limitando a visualização à oferta da grelha. Desta forma, reproduz-se a experiência considerando um subconjunto de registos de um pequeno grupo de utilizadores. Eles são divididos em dois conjuntos: o de treino, consistindo no histórico de visualização de cada utilizador, e o de teste, contendo registos de itens anteriormente vistos e de itens novos. Os itens que constam no conjunto de teste são considerados itens candidatos a recomendação. Destes, os que se encontram também no conjunto de treino, dizem respeito aos itens visualizados há pelo menos 30s pelo respectivo utilizador. Os novos, por sua vez, dizem respeito a programas de televisão que ainda não tiveram início ou que não foram visualizados, pelo menos, durante 30s.

3.2 Fontes de Dados

Para o cenário MEO considerou-se como fontes de informação sobre os conteúdos do serviço SAPO EPG (Electronic Programing Guide), que disponibiliza publicamente a grelha de programação dos vários canais durante o período de aproximadamente uma semana. Esta informação é obtida através da invocação de *Web services* por *Simple Object Access Protocol* (SOAP).

Muitos dos itens disponíveis na grelha têm uma descrição muito limitada do conteúdo — problema da incompletude dos itens. À semelhança de [Per10], e de forma a enriquecer a metainformação do item, são consideradas outras fontes de informação externas como a Internet Movie Database¹ (IMDb), a TV.com², a Living Social³ e a Wikipedia⁴.

As fontes externas são uma mais-valia para as estratégias BC. No entanto, estas fontes também poderão ser usadas em estratégias de FC desde que contenham qualquer tipo de pontuação ou comentário ao conteúdo.

A agregação das várias fontes externas aos itens obtidos do EPG dependerá da sua natureza: existem itens para os quais é de facto possível obter informação adicional como, por exemplo, séries de produção estrangeira e filmes. Na Tabela 3.1 são caracterizadas

¹<http://www.imdb.com/>

²<http://www.tv.com/>

³<http://tv.livingsocial.com/>

⁴<http://en.wikipedia.org/> e <http://pt.wikipedia.org/>

Caracterização dos Dados

Tabela 3.1: Caracterização das Fontes de Dados para itens

Fonte Externa	TV.com	IMDb	LivingSocial	Wikipedia
Domínio	Programas TV	Programas TV e Filmes	Programas TV	Programas TV e Filmes
Classificações	107.942	19.292	15.569	5
Itens Considerados	1.000	1.000	1.000	20
Utilizadores	46.032	14.299	5.862	
Intervalo das Classific.	1.0, 1.5, ..., 9.5, 10.0	1, ..., 10	1.0, 1.5, ..., 4.5, 5.0	—

as várias fontes de dados externas (informação parcialmente reunida em [Luc10]) e na Tabela 3.2 estão as discriminadas as principais informações contidas em cada fonte. Esta é a lista de informações existente para a maioria dos itens procurados.

Numa análise sintética pode notar-se que, a nível de classificações por parte dos utilizadores, a TV.com é a fonte mais rica e que, para caracterização do item e extracção de metainformação, o IMDb e a Wikipedia são as fontes que fornecem dados mais variados. O caso específico da Wikipedia tem vantagem face às outras fontes quando aplicada a itens de conteúdos de produção nacional, pois os seus conteúdos podem ser editados livremente pelos utilizadores.

Como fonte de informação de comportamento do cenário MEO coloca-se a hipótese de utilizar um componente desenvolvido na PT Inovação apelidado de *Broker* de Contexto. Este componente é consiste num servidor de presença e utiliza *Extensible Messaging and Presence Protocol* (XMPP)⁶ que permite a publicação e subscrição de nós (interpretados como tópicos) numa estrutura hierárquica. A escolha deve-se ao facto deste componente ser transversal a vários serviços existentes e de facilitar a aquisição de infor-

⁵Não existem classificações na Wikipedia. No entanto, poderão existir ligações à rede social Facebook onde é possível extrair o número de *Likes* (preferências)

⁶<http://xmpp.org/>

Tabela 3.2: Principais Informações contidas em cada fonte de dados de itens

Principais Informações	TV.com	IMDb	LivingSocial	Wikipedia
Título	x	x	x	x
Géneros	x	x	x	x
Descrição	x	x	x	x
Tema	x			x
Autores	x	x		x
Realizador	x	x		x
Produtor		x		x
Empresas		x		x
Actores / Apresentadores	x	x		x
Língua		x		x
País		x		x
Duração		x		x

Tabela 3.3: Elementos semânticos do item

Grupo	Elemento Semântico
Transmissão	Duração Hora de início Hora de fim Canal
Classificação	Faixa etária Categoria (e.g. Desporto, Filme, Informativo)
Pessoas	Actores/Apresentadores Argumentista Produtor Realizador
Genérico	Título Sinopse Título Episódio Número Episódio Número Temporada

mação de consumo de itens de várias naturezas como vídeos do Youtube ou notícias lidas no Portal SAPO.

No cenário simulado de televisão, as observações estão presentes no ficheiro do conjunto de dados.

3.3 Dados do Conteúdo

Os dados referentes ao conteúdo têm várias origens, dependendo do domínio. No cenário MEO, a fonte dos itens a considerar é o EPG da SAPO. Isto é um desafio para o desenvolvimento do recomendador porque, ao contrário dos filmes, é uma lista bastante volátil: os itens passados não podem voltar a ser consumidos a não ser que repitam; itens como o “Telejornal” poderá ter o mesmo nome mas conteúdos completamente diferentes, o que significa que pode ser uma boa recomendação para um utilizador se passar uma notícia sobre a ida do Homem à lua, mas uma péssima recomendação se a maioria do conteúdo se resumir à exposição de episódios de violência.

De forma a colmatar a volatilidade, o modelo de representação do item terá que dar mais relevância, no caso dos conteúdos classificados como série, ao título da série e não ao título do episódio. Para solucionar a falta de informação disponível no EPG, terá de ser feito o cruzamento com outras fontes apresentadas anteriormente.

A agregação deverá ser feita extraindo informações das diferentes fontes através da análise de conteúdos *Web* ou de metainformação disponível em formatos semânticos (como *Resource Description Framework*, RDF, ou *Web Ontology Language*, OWL), quando

disponível. As informações serão guardadas numa representação *Extensible Markup Language* (XML) definida por um esquema *XML Schema Definition* (XSD) (ver no Anexo B.2.1).

A Tabela 3.3 contém os elementos semânticos presentes na representação de um item de conteúdo. O título do episódio deve ser o elemento unívoco, devendo ser, sempre que necessário, adaptado para a categoria. Por exemplo, a transmissão de um jogo de futebol deverá ter no título as equipas que se defrontam, bem como o nome do campeonato.

No cenário simulado de televisão, os dados obtidos são o subconjunto disponível em fontes externas como IMDb, a partir do número de identificação IMDb presente no *dataset* da Netflix.

3.4 Dados de Comportamento

Os dados de comportamento poderão ser explícitos quando existe um valor numérico associado à observação (*rating*) ou implícitos quando apenas é conhecida a observação. É normal cada registo ser acompanhado de um campo que indica a data de consumo do item.

Para o cenário MEO, a informação disponível é implícita. Para o cenário simulado de TV, o *dataset* original Netflix é transformado em formato binário, truncando todos os registos e considerando-os observações. Assim, é criado o conjunto NF1 contendo aproximadamente 2,4 milhões registos de 7266 utilizadores distintos para um total de 17 146 itens. Os utilizadores visualizaram entre 316 e 338 itens, valores normais para um consumo de 2,75 horas médias diárias de televisão ao longo de um período de 4 meses.

O NF1 é dividido num conjunto de treino e num conjunto de teste, denominados NF1T e NF1P respectivamente. A divisão foi feita aleatoriamente numa proporção 80%-20%.

O NF1T contém 1 900 050 registos com observações de 7266 utilizadores para 16 877 itens. A Figura 3.1 indica a distribuição ordenada do número de itens visualizados para cada utilizador. Cada utilizador visualizou em média aproximadamente 262 objectos de conteúdo, com um desvio padrão de aproximadamente 9 itens. Por sua vez, a Figura 3.2 contém a distribuição ordenada do número de utilizadores que visualizaram cada um dos 16 877 itens, destacando bem a diferente popularidade dos vários. Em média cada um foi visto por aproximadamente 112 utilizadores, com um desvio padrão de 369 utilizadores.

O conjunto de teste, NF1P, contém 475 551 registos com observações de 7266 utilizadores para 13 295 itens. Estas observações não são consideradas na fase de treino dos vários algoritmos, sendo o objectivo da sua utilização a validação dos mesmos. Este conjunto simulará a dinâmica de televisão por conter cerca de 21% (3582) novos itens, que serão apresentados aos novos utilizadores, retratando assim a alta taxa de novos itens face ao domínio dos filmes.

Caracterizando NF1P com a distribuição da Figura 3.3 verifica-se que cada utilizador visualizou em média aproximadamente 65 objectos de conteúdo, com um desvio padrão

Caracterização dos Dados

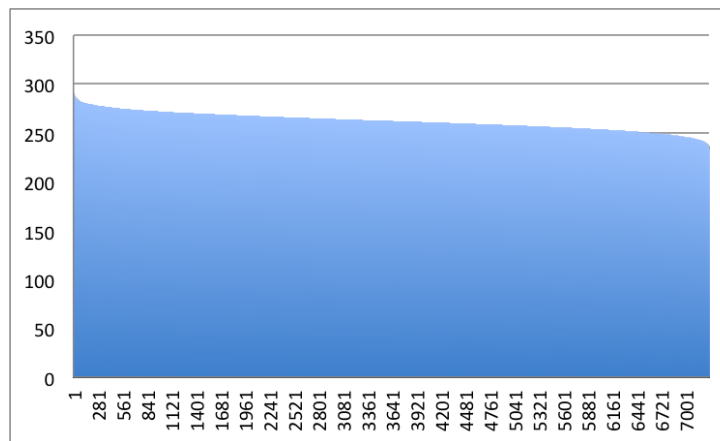


Figura 3.1: Gráfico de distribuição do número de itens visualizados por utilizador no conjunto de treino.

de 7 itens. Por sua vez, a Figura 3.4 contém dados da distribuição de utilizadores por item, tendo cada um em média 36 utilizadores, com um desvio padrão de 103 utilizadores.

3.5 Sumário

Neste capítulo caracterizou-se o tipo de dados usados em cada uma das duas abordagens apresentadas no capítulo anterior — dados de conteúdo para abordagens baseadas em conteúdo e dados de comportamento para técnicas de filtragem colaborativa.

São analisadas algumas fontes de metainformação de objectos de conteúdo quanto à qualidade da informação disponível. A combinação de várias fontes enriquece a representação de cada item, podendo melhorar uma abordagem baseada em conteúdo.

Descreve-se o conjunto de dados para o qual o sistema desenvolvido foi pensado (dados do operador de IPTV MEO). Descreve-se o conjunto de dados usado nos ensaios (Netflix), como alternativa ao conjunto de dados MEO. É descrita a metodologia de tratamento de dados adoptada para a simulação de um cenário de TV com dados de filmes e fundamenta-se os critérios de selecção da porção do *dataset* Netflix.

No capítulo seguinte apresenta-se a arquitectura adoptada para o tratamento dos dados disponíveis.

Caracterização dos Dados

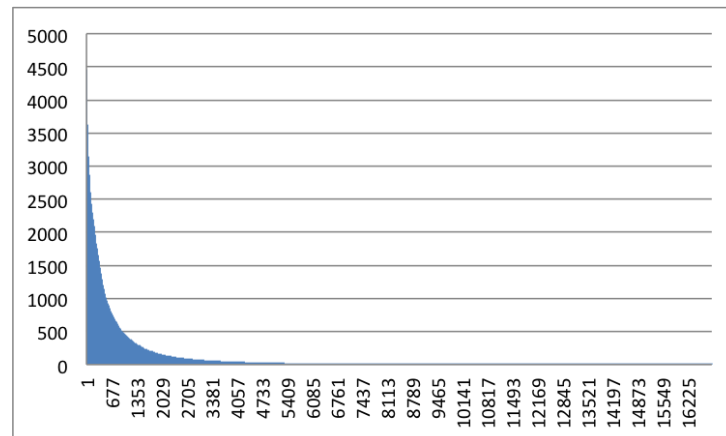


Figura 3.2: Gráfico de distribuição do número de utilizadores por objecto de conteúdo no conjunto de treino.

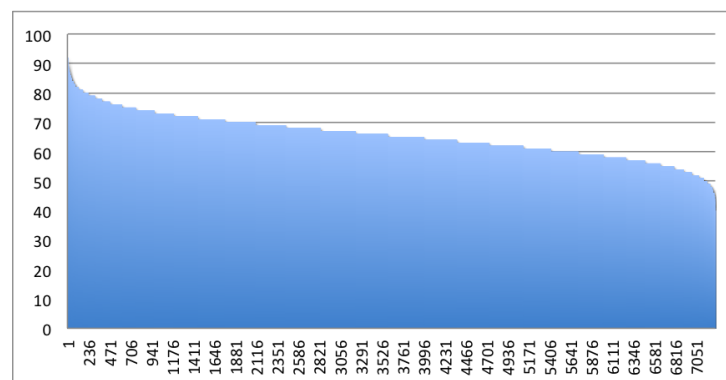


Figura 3.3: Gráfico de distribuição do número de itens visualizados por utilizador no conjunto de teste.

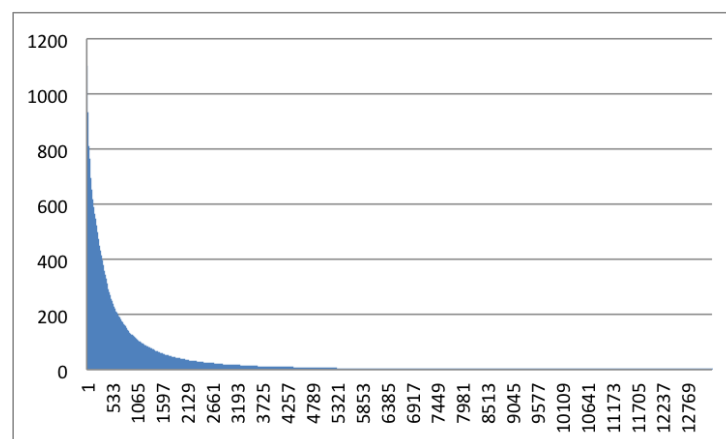


Figura 3.4: Gráfico de distribuição do número de utilizadores por objecto de conteúdo no conjunto de teste.

Caracterização dos Dados

Capítulo 4

Tecnologia e Arquitectura

Neste capítulo é descrita a solução implementada para responder aos requisitos definidos na Secção 1.3 e a arquitectura adoptada. Para um melhor esclarecimento das decisões introduzem-se também, de uma forma breve, as tecnologias usadas.

A Secção 4.2 descreve algumas das tecnologias usadas para o tratamento dos dados e para o desenvolvimento dos motores de recomendação. É feito algum estudo das capacidades das várias ferramentas e *frameworks* para o desenvolvimento do recomendador. De seguida, são apresentadas as tecnologias utilizadas na arquitectura da solução.

4.1 Introdução

A solução proposta baseia-se em técnicas *model-based* que pressupõem a criação de um modelo simples que será usado para determinar a afinidade entre utilizadores e itens de conteúdo. A construção de modelos facilita também a agregação de outra informação relacionada com as preferências vinda de fontes terceiras.

A solução, baseada em [DDGR07], foi pensada para um cenário de produção real tanto a nível de arquitectura como a nível de tecnologia. Valorizou-se a capacidade de resposta a pedidos em tempo real, a escalabilidade praticamente linear com o aumento da dimensão do problema e a tolerância à falha dos vários componentes presentes na arquitectura.

Foram analisadas várias soluções tecnológicas para o desenvolvimento dos componentes e optou-se por utilizar aquelas com mais capacidade de escalabilidade.

4.2 Tecnologias e Ferramentas

Nesta secção são apresentadas algumas tecnologias e ferramentas que facilitem e suportem o desenvolvimento de um sistema de recomendação. São também apresentados algumas ferramentas que permitam fazer outras tarefas, como a preparação de dados.

4.2.1 Desenvolvimento de Sistemas de Recomendação

Recentemente tem-se assistido a um esforço por parte de comunidades de código livre para se construir ambientes de trabalho que facilitem o desenvolvimento de recomendadores. Segue-se uma lista das bibliotecas e *frameworks* analisadas que têm este objectivo:

- **Apache Mahout**¹ — biblioteca com uma vasta colecção de técnicas de *machine learning*, escalável e com suporte para grandes conjuntos de dados. A vasta colecção inclui filtragem colaborativa, através da biblioteca *Taste*, e tem vários algoritmos de *clustering* (e.g. *k*-Means, Canopy, MinHash) e de classificação (e.g. classificadores bayesianos, Restricted Boltzmann Machines, Support Vector Machines). Todas as rotinas são escritas em Java e a escalabilidade é feita através da plataforma Apache Hadoop, que implementa, entre muitas outras coisas, a orquestração de processos utilizando o paradigma MapReduce [DG04].
- **ColFi**² (*Recommender System for a Dating Service*) — sistema de recomendação colaborativo, criado pela Universidade de Charles (República Checa), para um sítio *Web* de encontros que implementa algoritmos *user-to-user* e *item-to-item*. O seu desenho é flexível e permite acrescentar novos algoritmos com alguma facilidade. O estado de desenvolvimento nunca chegou a sair de *alpha* e já não é mantido desde Dezembro de 2009.
- **Cofi**³ — desenvolvido pela Universidade de Quebec, contém uma boa colecção de algoritmos colaborativos tanto *memory-based*, como *learning-free* (i.e. que não usam qualquer informação do utilizador actual). Não foi encontrada qualquer tipo de documentação e o projecto não é mantido desde 2005.
- **Duine Framework**⁴ — conjunto de bibliotecas na linguagem Java que permitem a criação de componentes de predição e a combinação de vários componentes. Para a construção dos componentes, a *framework* contém vários algoritmos de filtragem colaborativa e baseados em conteúdos, sendo relativamente fácil expandir os existentes ou adicionar outras implementações. Para a combinação de componentes,

¹<http://mahout.apache.org/>

²<http://colfi.wz.cz/>

³<http://savannah.nongnu.org/projects/cofi/>

⁴<http://www.duineframework.org/>

Tabela 4.1: *Frameworks* de desenvolvimento de SR — Comparação entre projectos

	Apache Mahout	ColFi	Cofi	Duine Framework
Linguagem Programação	Java	Java	Java	Java
Qualidade Documentação	Boa	Boa	Elementar	Boa
Suporte da Comunidade	Bom	—	—	Bom
Última actualização	Fevereiro 2011	Dezembro 2009	2005	Janeiro 2009
Tipo de Utilização	<i>Framework</i>	<i>Framework</i>	<i>Framework</i>	Algoritmos de Predição

permite definir uma estratégia de decisão da aplicação dos vários componentes de predição conforme a situação. Não há actualizações desde Janeiro de 2009.

Foram encontradas outras opções como MyMediaLite⁵, mas por se encontrar numa fase inicial de desenvolvimento e por ser construído em tecnologia Microsoft .NET de portabilidade limitada não foi considerada. A Tabela 4.1 apresenta o estudo das várias opções, de uma forma mais criteriosa, quanto à informação conhecida sobre cada projecto. Na Tabela 4.2 são comparadas quanto às funcionalidade implementadas até Fevereiro de 2011.

Por ser a plataforma com uma maior comunidade e actividade, optou-se pelo Apache Mahout. Esta tecnologia tem vantagens como a integração com o Apache Solr/Lucene, plataforma usada para indexar documentos, e a integração com o Apache Hadoop, que possibilita acompanhar o problema à medida que a sua dimensão aumenta através do paradigma MapReduce. Quanto às técnicas implementadas, existe um maior suporte no Mahout do que na Druine Framework, o que permite explorar resultados obtidos com várias técnicas.

⁵<http://www.ismll.uni-hildesheim.de/mymedialite/index.html>

Tabela 4.2: *Frameworks* de desenvolvimento de SR — Comparação de funcionalidades

	Apache Mahout	ColFi	Cofi	Duine Framework
Baseado em Conteúdo	○	—	—	◇
FC centrada no item	◇	◇	◇	◇
FC centrada no utilizador	◇	—	◇	—
Hibridação	○	◇	—	◇
Técnicas <i>memory-based</i>	◇	◇	◇	◇
Técnicas <i>model-based</i>	○	◇	◇	◇
Módulo de Avaliação	◇	—	—	◇

(Legenda: ◇ — funcionalidade completa; ○ — funcionalidade não completa, com solução conhecida)

4.2.2 Tecnologia de Persistência

Como soluções de persistência opta-se pelo Apache HBase, que é a implementação *open-source* do Google BigTable [CDG⁺08]. O Apache HBase é uma base de dados não relacional que, apesar de suportar vários sistemas de ficheiros, tem uma boa integração do *Hadoop Distributed Filesystem* (HDFS), sistema de ficheiros distribuído usado na plataforma Hadoop. Desta forma, é possível alimentar processos MapReduce com os dados guardados em tabelas e actualizá-las com os artefactos gerados por esses processos. As grandes vantagens do HBase usado em conjunto com o HDFS é a tolerância à falha inerente ao HDFS, que divide os ficheiros em blocos e os replica redundantemente pelos vários computadores.

O HBase é uma base de dados orientada à coluna e funciona como uma estrutura associativa distribuída em que a cada linha (definida por uma chave primária) estão associadas várias colunas organizadas por família. Assim como no BigTable, cada nó do *cluster* de computadores gere um intervalo de chaves e as pesquisas são feitas em paralelo através de filtros de *Bloom*.

4.2.3 Outras Tecnologias

Para a agregação de dados dos itens, provenientes do EPG e de fontes externas que enriquecem a qualidade da meta-informação e complementam a existente, foram analisadas tecnologias usadas para *scrapping*. As bibliotecas Mechanize⁶ e BeautifulSoup⁷, escritas na linguagem Python⁸, permitem respectivamente obter o conteúdo HTML (*HyperText Markup Language*) de uma página e filtrar o mesmo através de várias formas, como expressões regulares, de forma a extrair conteúdo da página. O processo de extracção e de agregação tem como saída um ficheiro XML contendo a informação reunida sobre os itens.

Para tratar a informação reunida sobre os itens, é necessário fazer um pré-processamento, que envolve a separação do texto em *tokens*, a eliminação de *tokens* sem significado semântico para a mensagem (*stop words*), a redução dos termos à sua raiz morfológica de forma a normalizar plurais e formas verbais (*stemming*) e, finalmente, a indexação dos termos característicos do item, usando técnicas de *Information Retrieval* (IR).

As ferramentas consideradas para a eliminação de *stop-words* e para a operação de *stemming* foram o Snowball⁹, por conter um *stemmer* para a língua portuguesa, implementada por investigadores da área e usado noutros projectos. Uma vantagem é que é

⁶<http://wwwsearch.sourceforge.net/mechanize/>

⁷<http://www.crummy.com/software/BeautifulSoup/>

⁸<http://www.python.org/>

⁹<http://snowball.tartarus.org/>

possível aplicar o *stemmer* na ferramenta Apache Solr¹⁰, que será considerada para a tarefa de indexação.

A tarefa de indexação usará Apache Solr, que analisa o texto e permite escolher qual o tipo de indexação que podemos fazer nos documentos (objectos de conteúdo). O Apache Solr usa a conhecida plataforma Apache Lucene¹¹ para efectuar os cálculos dos pesos dos termos para a indexação dos itens. O Apache Solr contém ainda uma interface por *Web services* que permite efectuar pesquisas nos itens.

4.3 Arquitectura

É distinguida a arquitectura lógica, onde se identificam os sistemas presentes na solução e se define a função que cada um desempenha, e arquitectura física, onde se definem os componentes que implementam os pacotes da arquitectura lógica, detalhando o seu papel objectivo, as interacções com os outros elementos e que tecnologias estão associadas à sua concepção.

4.3.1 Arquitectura Lógica

A arquitectura lógica inclui vários pacotes representando os sistemas e subsistemas da solução. Os pacotes presentes na Figura 4.1 são o **Repositório de Metaconteúdos** (*meta-content repository*), o **Sistema de Aquisição de Observações** (*observations acquisition*), o **Sistema de Cálculo de Recomendação** (*recommender*), o **Sistema de Persistência** (*persistence*), o **Sistema de Aprendizagem** (*machine learning*), o **Módulo de Avaliação** (*evaluation module*) e o **Colector de Itens** (*item data collector*). Os vários sistemas enunciados podem funcionar *online*, se a sua execução é continuada e permanente, e *offline*, se tiverem um ciclo de vida curto e forem executados periodicamente.

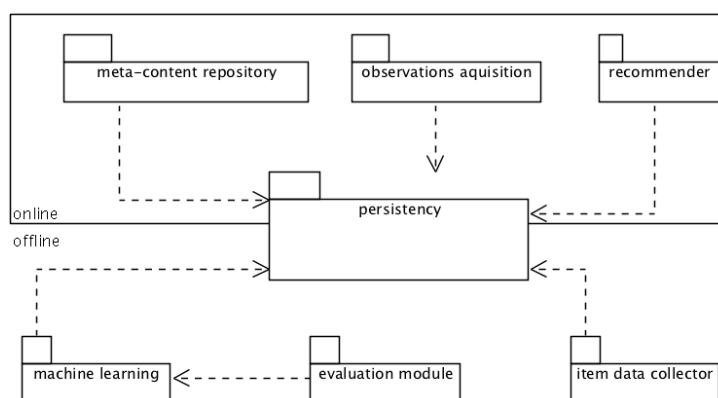


Figura 4.1: Arquitectura Lógica, representando os vários sistemas *online* e *offline*.

¹⁰<http://lucene.apache.org/solr/>

¹¹<http://lucene.apache.org/>

Numa descrição mais detalhada, os vários pacotes que constituem a arquitectura lógica são:

- **Repositório de Metaconteúdos** — tem como função disponibilizar a informação existente sobre os itens a outros sistemas. Este sistema depende de um motor de persistência onde são guardadas as representações semânticas dos objectos de conteúdo. Conta com o apoio do sistema **Colector de Itens** (*item data collector*), que é responsável pela agregação da informação vinda das fontes anterior referidas (i.e., EPG, IMDb, Wikipedia) e pela construção da representação final do item. A função principal do repositório é a disponibilização dos itens candidatos a recomendação num determinado momento.
- **Sistema de Aquisição de Observações** — é responsável por processar informação sobre visualizações em tempo real, contabilizando o número de utilizadores a ver um determinado item. Este sistema contabiliza também os itens mais vistos por cada comunidade de utilizadores (*cluster*) e a co-visitação de dois itens por elementos de um determinado grupo. Os dados adquiridos são os necessários para o cálculo da recomendação em tempo real.
- **Sistema de Cálculo de Recomendação** — é o sistema que tem como papel a geração de recomendações para um utilizador. Este sistema obtém a lista de itens disponíveis para recomendação mantida pelo Repositório e usa as estatísticas recolhidas pelo Sistema de Aquisição de Observações para efectuar o cálculo da recomendação em tempo real.
- **Sistema de Persistência** — é um sistema que tem como função o armazenamento de dados, a pedido de vários sistemas. A centralização deste sistema permite uma interacção entre os componentes assíncrona, facilitando alguns dos processos descritos no Capítulo 5. A sua natureza é crítica uma vez que há vários sistemas dependem dele.
- **Sistema de Aprendizagem** — é o sistema que se alimenta dos dados estatísticos recolhidos e os processa aplicando técnicas de *machine learning* para geração de *clusters* que representam comunidades de utilizadores. Tem como artefactos de saída os modelos que consistem na lista de *clusters* a que cada utilizador pertence e o seu nível de comprometimento (valor entre 0 e 1, definindo a afinidade do utilizador com o *cluster*). Gera também outro artefacto secundário que contém métricas sobre a execução dos algoritmos, que será analisado posteriormente pelo Sistema de Avaliação.
- **Sistema de Avaliação** — trata de monitorar a qualidade dos vários algoritmos de *machine learning*, fornecendo ao Sistema de Aprendizagem um conjunto de dados

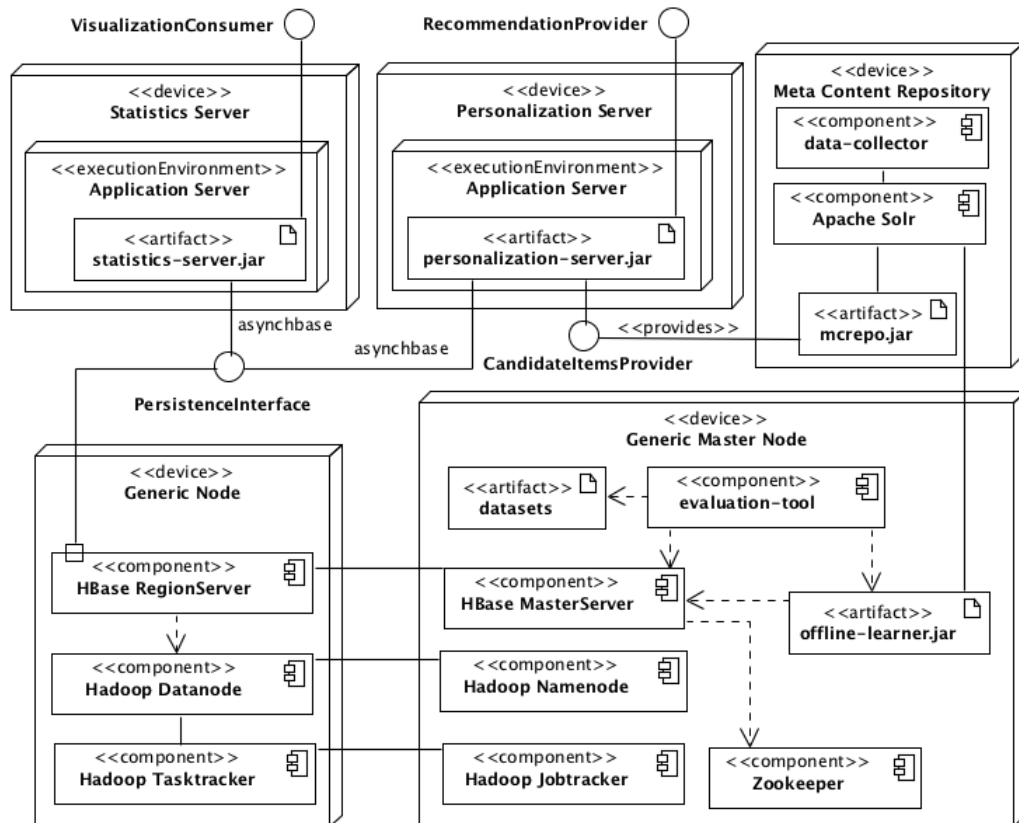


Figura 4.2: Arquitectura Física

de treino. Os vários artefactos produzidos pelo Sistema de Aprendizagem são processados e comparados com um conjunto de dados de teste. As várias métricas que advêm do tratamento destes artefactos são usadas para definir os parâmetros da solução em produção. Este é um sistema de uso regular e de manutenção dos vários sistemas.

4.3.2 Arquitectura Física

A arquitectura física, ilustrada no Diagrama de Distribuição da Figura 4.2, define quais os componentes, as ferramentas e as tecnologias usadas neste trabalho para implementar cada sistema definido na arquitectura lógica.

Os sistemas de Aprendizagem e de Avaliação dão origem, respectivamente, à **Ferramenta de Aprendizagem** (*offline-learner*) e à **Ferramenta de Avaliação** (*evaluation-tool*). O Sistema de Persistência é dividido em dois componentes: um **Indexador** (Apache Solr), que processa a representação dos itens com técnicas de *Information Retrieval*, para que se possa no futuro explorar abordagens BC e um motor de persistência (HBase), descrito mais à frente, disponível para todos os componentes. Os sistemas de Aquisição de

Observações e Cálculo de Recomendação dão origem a dois componentes *online*, respectivamente o **Servidor de Estatísticas** (*statistics-server*) e o **Servidor de Personalização** (*personalization-server*).

O facto de dividir as acções do sistema pelos vários componentes irá aumentar a tolerância a falhas. Desta forma, o servidor de personalização poderá continuar a servir recomendações se o servidor de estatísticas estiver a falhar. O único ponto de falha única é o motor de persistência que poderá ser replicado ou distribuído. As ferramentas *offline* como a Ferramenta de Aprendizagem e de Avaliação correm periodicamente e não constituem um ponto crítico do sistema em produção.

A plataforma Hadoop, composta pelos componentes de HDFS (armazenamento em sistema de ficheiros distribuído) *Datanode* e *Namenode*, e pelos componentes de MapReduce (processamento) *TaskTracker* e *JobTracker*, é a eleita para o processamento *offline* dos *clusters* MinHash e a construção do modelo PLSI.

O HBase posiciona-se acima da plataforma Hadoop, servindo-se do HDFS para persistir. Ele é composto por vários componentes *RegionServer* que serão responsáveis por um conjunto de chaves e por um conjunto de *HMaster*s concorrentes que tentam competir pelo controlo da infra-estrutura. Os *HMaster* mantêm o registo das linhas à responsabilidade de cada *RegionServer*. A competição evita o *single point of failure* de um único *HMaster* e assegura a consequente tolerância à falha. O *Zookeeper* é o componente responsável por eleger o *HMaster* e a eleição é feita com um número ímpar de instâncias.

A Ferramenta de Aprendizagem usa os dados de visualizações directamente do HBase ou de um ficheiro no HDFS e calcula os *clusters*. Os *clusters* poderão ser armazenados num ficheiro ou no HBase. A Ferramenta de Avaliação, por sua vez, alimenta o HBase com um conjunto de dados (*dataset* de treino) e invoca a ferramenta de aprendizagem. Os artefactos por ela gerados são comparados com o *dataset* de teste.

O Servidor de Estatísticas e o Servidor de Personalização comunicam com o HBase por intermédio da *PersistenceInterface*, usando a biblioteca ‘*asynchbase*’ (descrita mais à frente, na Secção 5.3)

O Colector de Itens constrói a representação do item a partir de informações obtidas de várias fontes externas e insere a informação no Indexador. O Indexador consiste numa instância Apache Solr, que persiste e indexa uma colecção de entradas referentes a programas de televisão e a filmes. Este componente armazena os dados em vectores dispersos TF-IDF que poderão ser posteriormente analisados pela Ferramenta de Aprendizagem para cálculo de similaridade com os itens mais recentes. Esta abordagem não foi explorada e é referida como trabalho futuro.

Por sua vez, o Repositório de Metaconteúdos explora as funcionalidade de pesquisa em texto do Apache Solr para obter os itens mais recentes ou referentes a um certo tópico. O Servidor de Personalização solicita a lista de itens com transmissão agendada para o presente intervalo de tempo e efectua recomendações nesta lista de itens candidatos.

Por fim, os serviços *online* oferecem os seus serviços usando REST sobre HTTP. Como servidor de aplicações opta-se pelo Jetty¹², escrito em Java.

4.4 Sumário

Neste capítulo analisaram-se as tecnologias utilizadas e definiu-se a arquitectura da solução ao problema.

As tecnologias adoptadas são de código fonte livre, com capacidade de escalar numa infra-estrutura distribuída. A tolerância à falha foi também considerada como um factor importante para o desenvolvimento de um sistema de recomendação robusto para ambiente de produção.

A arquitectura descrita baseada em [DDGR07] permite responder aos requisitos de tempo e expandir com o aumento da dimensão do problema.

Identificaram-se, por último, os vários componentes constituintes do sistema, descrevendo o seu papel e as tecnologias envolvidas no seu desenvolvimento. No capítulo seguinte é descrita em pormenor a implementação de cada componente.

¹²<http://jetty.codehaus.org/jetty/>

Capítulo 5

Detalhes de Implementação

Este capítulo apresenta os pormenores mais relevantes da implementação de cada componente identificado na Arquitectura Física.

5.1 Visão Geral

O desenvolvimento de todos os componentes implementados foi feito com o apoio da ferramenta Apache Maven¹, que permitiu uma organização intuitiva das directorias do código-fonte, gestão de dependências e execução automatizada dos testes desenvolvidos. Para definição de testes unitários foi usada a biblioteca JUnit² para Java e a biblioteca *unittest* para a linguagem Python.

5.2 Servidor de Metaconteúdo

O Servidor de Metaconteúdo é responsável por manter uma lista actualizada de informações relativas aos itens de conteúdo. Usa o Apache Solr para gerir e pesquisar na informação, podendo assim filtrar os itens por período temporal ou área temática. O componente é feito em Java e usa *Jersey* para servir a colecção de itens candidatos a recomendação por REST.

O motor de persistência e indexação utilizado é o Apache Solr e este é alimentado pelo Colector de Itens, que está implementado em Python. Este colector contém vários módulos que processam a representação do item desde a sua aquisição à sua persistência. Os módulos executam cada uma das actividades presentes no diagrama da Figura 5.1, consistindo cada um deles numa classe Python com uma interface comum.

¹<http://maven.apache.org/>

²<http://junit.sourceforge.net/>

Detalhes de Implementação

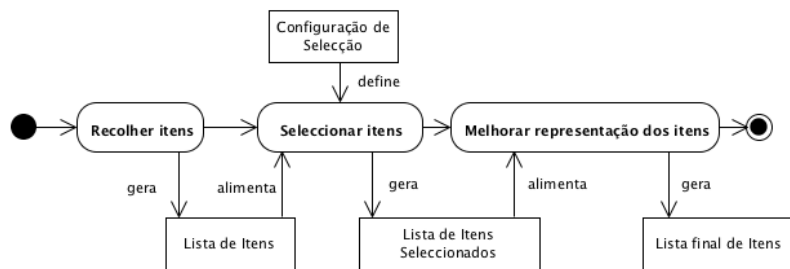


Figura 5.1: Diagrama de actividades do Colector de Itens

Desta forma é possível ter vários colectores, cada um específico para cada fonte de itens, e vários *improvers*, que são responsáveis por melhorar a representação através de outra fonte específica, interagindo com eles de modo semelhante. Foram implementadas duas classes colectoras: uma para os *Web services* SOAP da SAPO, que contém informação do EPG, usando a biblioteca *suds*³, e outra para o *dataset* do Netflix. O seleccionador de itens implementado apenas filtra os itens por título, eliminando todos aqueles que tenham um dos padrões especificados na configuração. O papel desta actividade é recusar itens desinteressantes para recomendação, como é o caso de programas de Televidas no domínio da televisão. Foram também implementadas duas classes de melhoramento: uma que procura o item no IMDb e, se houver um e um só resultado, adiciona os campos obtidos à representação; e outra que pesquisa na Wikipedia e extrai os campos presentes na caracterização geral (secção que se apresenta do lado direito da página, contendo informação estruturada). Das fontes analisadas na Secção 3.2 optou-se pelas que continham mais metainformação.

Todas as actividades são executadas na plataforma Hadoop através do módulo *hadoop-streaming* que permite processar a informação passando pares chave-valor em formato CSV (*Comma Separated Value*) a um qualquer programa pelos canais habituais de entrada/saída (neste caso, através do *stdin* e *stdout* do *script*). Desta forma, é possível fazer cada fase de recolha e melhoria de forma distribuída, processando os itens em paralelo em cada fase.

O formato de representação do item é inicialmente em JSON, pois diminui o tamanho da transferência necessária entre os vários passos do processo. No final realiza-se um pedido de indexação por REST ao servidor *Web* do Apache Solr, dando como parâmetro a representação XML do item, respeitando um esquema XSD especificado (ver Anexo B.2.1).

³<https://fedorahosted.org/suds/>

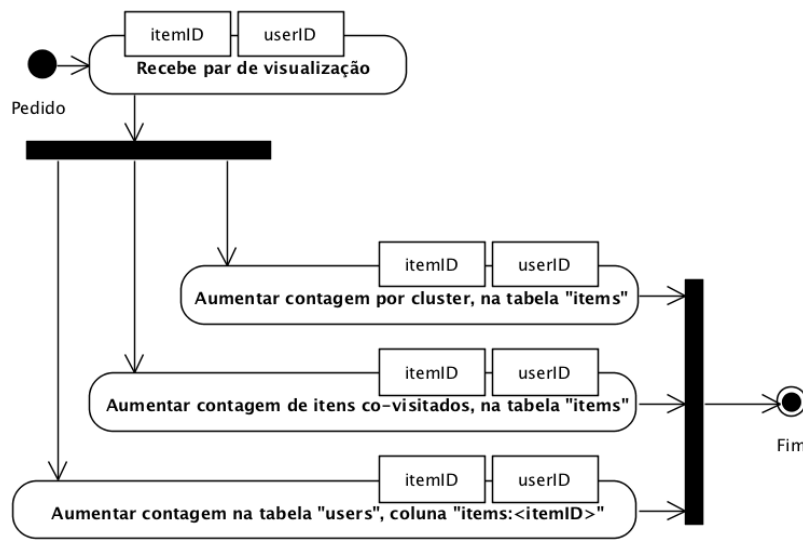


Figura 5.2: Diagrama de actividades do Servidor de Estatísticas, accionado por pedido

5.3 Servidor de Estatísticas

O Servidor de Estatísticas, implementado em Java, contém um *singleton*, que interage com o motor de persistência e um recurso REST, desenvolvido com a biblioteca *Jersey*⁴. O motor de persistência utilizado é o HBase e a interface usada é o *asynchbase*⁵, que permite realizar pedidos não bloqueantes e *thread-safe*. Esta opção veio aumentar o número de pedidos por segundo a este componente (ver Secção 6.3) face à API nativa do HBase.

O diagrama da Figura 5.2 representa a sequência de actividades realizadas aquando da recepção de um pedido.

O componente recebe um pedido REST, por HTTP, com os parâmetros *userID* e *itemID*, referentes aos identificadores únicos do utilizador e do item visualizado. São desencadeadas três actividades assíncronas, em paralelo:

- uma para incrementar a contagem de observações do par utilizador-item, representado pelo valor presente na coluna ‘items:<itemID>’, na tabela *users*;
- outra para incrementar a contagem de observações para cada *cluster* (MinHash e PLSI) na coluna ‘cluster_mh:<clusterID>’ e ‘cluster_plsi:<clusterID>’ da tabela *items*. A lista de *clusters* é obtida a partir do utilizador e o incremento é na proporção da afinidade do utilizador com o *cluster* — 1, no caso do MinHash e $p(z|u)$, no caso do PLSI. É também incrementado um valor de normalização T , na tabela *clusters*, que consiste no total de visualizações efecutadas por membros do *cluster*;

⁴<http://jersey.java.net/>

⁵<https://github.com/stumbleupon/asynchbase>

- uma última, para associar o item visto com todos aqueles já presentes no histórico recente do utilizador. O valor na coluna ‘co_item:<coItemID>’ da tabela *items* é incrementado, onde ‘coItemID’ é o identificador de um objecto de conteúdo presente no histórico. À semelhança dos *clusters*, é mantido um valor de normalização, na tabela *items*, que consiste no número de itens que foram co-visitados com o presente (*itemID*).

Todas as actividades são assíncronas, executando acções desencadeadas por eventos. Desta forma, não é necessário cada actividade bloquear nas operações de I/O na comunicação com a base de dados. Assim que os resultados de um pedido ao HBase estejam disponíveis, o *asynchbase* assegura que uma função é chamada para continuar a execução.

No fim da execução das três actividades, as tabelas ficam com os valores necessários para executar o cálculo de afinidade dos utilizadores aos itens candidatos para recomendação. Este cálculo é efectuado pelo Servidor de Personalização.

5.4 Servidor de Personalização

O Servidor de Personalização, implementado em Java, tem como função o cálculo de recomendações em tempo real a partir das informações guardadas no HBase. Este componente pede periodicamente a lista de itens candidatos a recomendação ao Repositório de Metaconteúdo, através de chamadas REST e adoptando XML como formato de mensagem. Esta lista, no domínio de programas de televisão, consiste nos itens que vão ser transmitidos num intervalo próximo definido (e.g. 1 hora) ou a colecção de filmes disponíveis para a aplicação ao cenário MEO Vídeoclube.

Quando recebe um pedido de recomendações destinadas a um utilizador u , para cada item c da lista de itens candidatos são calculados três valores de avaliação S (ver Figura 5.3): duas avaliações *cluster*-candidato (S_M para MinHash e S_P PLSI), uma avaliação de co-visitação S_I (considerando os itens vistos recentemente por u).

Para o cálculo de S_P são obtidos os *clusters* de u e os *clusters* cujos membros observaram c recentemente. Para os *clusters* z presentes na intersecção destes dois conjuntos, denominada por P , calcula-se $S_P(z)$, dada pela Equação 5.1, onde o numerador é a contagem de observações recentes do item c por membros de z ponderada com o grau de compromisso do utilizador u ao *cluster* em questão e o denominador é a contagem total de observações por membros do *cluster* z . O grau de compromisso é igual a $p(z|u)$. O cálculo final de S_P é dado pela Equação 5.2.

$$S_P(z) = \frac{\text{membership}(u, z) * \text{observations}(z, c)}{\text{observations}(z)} \quad (5.1)$$

Detalhes de Implementação

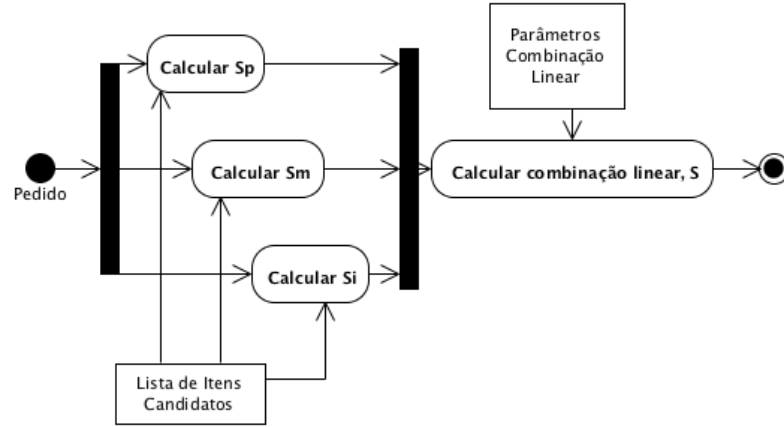


Figura 5.3: Diagrama de actividades do Servidor de Personalização, accionado por pedido

O cálculo de S_M é feito de forma análoga, embora o factor de ligação do utilizador ao *cluster* ($membership(u, p)$) seja sempre igual a 1.

$$\frac{1}{|P|} \sum_{z \in P} S_P(z) \quad (5.2)$$

Por último, o cálculo de S_I é dado pela Equação 5.3, onde S é o conjunto de itens recentes vistos por u , T é o número de observações co-visitadas com c e $covisitation(s, c)$ é o número de observações co-visitadas por s e por c .

$$\frac{1}{T} \sum_{s \in S} covisitation(s, c) \quad (5.3)$$

Os valores de co-visitação e de observações são obtidos do HBase e o cálculo dos *scores* é feito paralela e assincronamente à semelhança do Servidor de Estatísticas através da biblioteca *asynchbase*. Foi ainda implementada uma *cache* de tamanho pré-definido para evitar que sejam feitos muitos pedidos à base de dados. A *cache* foi implementada com tabelas de dispersão (*HashMap*) e é limpa quando se esgota o seu espaço, podendo este valor ser configurado.

As avaliações calculadas são combinadas linearmente resultando num *score* final, S_F , igual a $\alpha_I S_I + \alpha_M S_M + \alpha_P S_P$. No entanto existem alguns casos especiais como a recomendação para um utilizador recente, cujo histórico de visualizações pouco se conhece. Neste caso, são retornados os itens mais populares, isto é, com mais visualizações. No cenário de televisão isto equivale aos itens mais vistos no momento, uma vez que o ciclo de vida da lista de candidatos é curta.

À semelhança do Servidor de Estatísticas, este componente usa o Jersey para expor os seus serviços através de REST, recebendo como argumento o identificador do utilizador (*userID*). Os pedidos ao HBase usam de igual modo o *asynchbase* e os cálculos são feitos em paralelo e de forma assíncrona.

O pedido retorna em formato XML uma lista de itens ordenada por *score*. O formato da mensagem está de acordo com o XSD utilizado pelo *Colector de Itens* (ver Anexo B.2.1).

5.5 Ferramenta de Aprendizagem

A ferramenta de aprendizagem consiste numa colecção de rotinas MapReduce referentes aos vários passos dos algoritmos aplicados (PLSI e MinHash). MapReduce é um paradigma de computação distribuída que potencia o processamento de grandes volumes de dados numa infra-estrutura heterogénea, com tolerância a falha. Este paradigma tem dois tipos de tarefa: a Map que emite conjuntos chave-valor para um dado conjunto de entrada, e a Reduce ao qual são passados os pares gerados pela Map agrupados por chave.

As rotinas estão implementadas em Java e preparadas para correr na plataforma Apache Hadoop. Foram também implementadas classes serializáveis para representar algumas chaves dos pares de valores trocados pelas rotinas, uma vez que as chaves complexas recorrem a tipos não primitivos do Hadoop para representarem os vários valores. Por fim, foram implementadas classes Driver que orquestram a execução das rotinas MapReduce.

Algumas estruturas de dados existentes no Apache Mahout foram utilizadas na implementação das rotinas.

As rotinas iniciais e finais de cada algoritmo foram desenhadas para ler e escrever dados para ficheiros e para tabelas da base de dados HBase. Isto permite alimentar o motor de persistência com os *clusters* gerados, actualizando directamente a informação de cada utilizador e de cada item sem ter de recorrer a outra ferramenta.

O sistema de ficheiros (HDFS) contém uma região dedicada à persistência de artefactos de métricas. As métricas são posteriormente processadas pela Ferramenta de Avaliação.

5.5.1 MinHash

A implementação do algoritmo MinHash (descrito em pormenor na Secção 2.5), em Hadoop, é feita através de uma tarefa MapReduce que processa linearmente o histórico de visualização de cada utilizador. Os dados de entrada provêm de um ficheiro contendo pares chave-valor referentes a ‘*userID*-[lista de *itemID*]’ ou de um *Scan* à tabela *users* presente no HBase.

A operação Map recebe cada par e, para cada identificador de item (4 bytes) na lista calcula *nh hashes*. Cada uma das *hashes* é calculada por uma função de dispersão diferente que tem na génese factores aleatórios definidos na sua criação. Foram implementadas três funções: uma linear (com dois parâmetros), uma polinomial (com três parâmetros) e uma função de Murmur (com um parâmetro). A criação das *nh* funções

é feita sequencialmente a partir da mesma *seed*, para assegurar que em todos os nós onde a operação de Map corre, as nh funções aplicadas são as mesmas, criadas com os mesmos parâmetros iniciais.

O valor mínimo da passagem de cada função é guardado, resultando no fim nh valores referentes às *hashes* das permutações ordenadas do conjunto, dadas pelo factor uniformemente aleatório passado na criação da própria função. Assim, sendo y o número de funções das nh às quais $h_{\min}(A) = h_{\min}(B)$, y/nh é o estimador de $Jaccard(A, B)$ com um erro esperado de $O(1/\sqrt{nh})$ (cf. [BCFM98]).

Para a criação dos *clusters* são concatenados os valores mínimos de *hash* calculados em kg grupos separados pelo símbolo ‘-’. Cada operação Map emite nh *clusters*. Um identificador de *clusters* com mais grupos (kg mais elevado) representa um *cluster* mais refinado. Uma vez que é composto pelos n -ésimo *cluster*, contém os $[n...n + kg]$ valores mínimos de *hash*.

A operação Reduce recebe os pares ‘*userID-clusterID*’ emitidos pela operação Map e contabiliza a quantos *clusters* é que cada utilizador pertence. O artefacto de saída é um artefacto com métricas cuja posterior análise servirá para refinar os parâmetros de entrada de forma a que os utilizadores, em média, não pertençam a demasiados *clusters*.

5.5.2 PLSI

A implementação da técnica PLSI contém várias operações MapReduce e toma como argumento o número de classes latentes z e o número de iterações it .

Começa com três operações para contar o número de itens distintos observados $|I|$, o número de itens que cada utilizador observou $|I_j|$ e o número de utilizadores observado por cada item $|U_v|$. Estas três operações foram implementadas para tomarem como fonte um ficheiro ou uma tabela do HBase, definindo uma coluna específica para a contagem.

De seguida, são inicializados dois ficheiros: ‘*pzu_0*’ que contém os pares ‘*userID-[p(z|u)∀z ∈ 1..L]*’ e o ficheiro ‘*psz_0*’ que contém os pares ‘*itemID-[p(s|z)∀z ∈ 1..L]*’. Os valores $p(s|z)$ iniciais são iguais a $1/|I|$ e os valores $p(z|u)$ iniciais são aleatórios e a sua soma é igual a 1. Estes ficheiros são gerados por duas operações MapReduce.

Depois, são executadas as it iterações do algoritmo EM, explicado na Secção 2.6, começando pelo passo E onde é calculado o valor esperado da função de verossimilhança logarítmica com os parâmetros $\hat{\theta}$, gerados anteriormente (‘*psz_0*’ e ‘*pzu_0*’). São calculados os valores $Q(z; u, s)$ para cada par observado, contendo a probabilidade da ocorrência de cada par (u, s) , isto é, a selecção do item s pelo utilizador u é atribuída ao facto de u pertencer à comunidade (*cluster*) z .

Para cada par utilizador-item observado, é calculado $p(s|z) * p(z|u)$ e armazenado no ficheiro ‘*pszPzu_i*’ registos com a chave composta (*userID, itemID, z*), onde $i ∈ \{1, ..., it\}$

representa a iteração actual. Posteriormente, é gerado o ficheiro ‘sumZPszPzu_i’ contendo pares com chave composta (*userID*, *itemID*) e valores iguais ao respectivo cálculo $\sum_z p(s|z) * p(z|u)$. De seguida, é calculado o valor de Q^* dividindo os valores de ‘pszPzu_i’ pelos valores em ‘sumZPszPzu_i’. Estes cálculos são guardados no ficheiro ‘qStar_it’ com o formato ‘(*userID*, *itemID*, *z*)-valor’, onde valor é o resultado da Equação 5.4.

$$\frac{p(s|z) * p(z|u)}{\sum_{z'=1}^L p(s|z') * p(z'|u)} \quad (5.4)$$

Passa-se ao passo M onde é determinado o parâmetro θ que maximiza a verossimilhança do modelo.

O passo de maximização começa por processar os registos em ‘qStar_i’ agrupados por item e a somá-los, escrevendo o resultado no ficheiro ‘sumSQStarPath_i’ com pares com chave ‘(*userID*, *z*)’ e com valor igual ao resultado da Equação 5.5.

$$\sum_{s \in I} \frac{p(s|z) * p(z|u)}{\sum_{z'=1}^L p(s|z') * p(z'|u)} \quad (5.5)$$

O ficheiro anterior é processado somando os valores agrupados por utilizador, resultando num ficheiro ‘sumZSQStar_i’ com registos com chave igual a ‘*userID*’ e valores iguais ao resultado da Equação 5.6.

$$\sum_z \sum_{s \in I} \frac{p(s|z) * p(z|u)}{\sum_{z'=1}^L p(s|z') * p(z'|u)} \quad (5.6)$$

O primeiro artefacto a ser usado na iteração seguinte é o ‘pzu_i’ que contém pares com chave *userID* e uma lista de valores com os L valores de z , dados pela Equação 5.7, com os registos de ‘sumSQStarPath_i’ em numerador e os registos de ‘sumZSQStar_i’ em denominador.

$$\frac{\sum_{s \in I} \frac{p(s|z) * p(z|u)}{\sum_{z'=1}^L p(s|z') * p(z'|u)}}{\sum_z \sum_{s \in I} \frac{p(s|z) * p(z|u)}{\sum_{z'=1}^L p(s|z') * p(z'|u)}} \quad (5.7)$$

Da mesma forma, os registos contidos em ‘qStar_i’ agrupados por utilizador são somados, escrevendo o resultado no ficheiro ‘sumUQStar_i’ com pares de chave ‘(*itemID*, *z*)’ e valor igual ao resultado da Equação 5.8.

$$\sum_{u \in U} \frac{p(s|z) * p(z|u)}{\sum_{z'=1}^L p(s|z') * p(z'|u)} \quad (5.8)$$

O ficheiro anterior é processado somando os valores agrupados por item, resultando num ficheiro ‘sumSUQStar_i’ com registos com chave z .

$$\sum_{s \in I} \sum_{u \in U} \frac{p(s|z) * p(z|u)}{\sum_{z'=1}^L p(s|z') * p(z'|u)} \quad (5.9)$$

O segundo e último artefacto que passa para a iteração seguinte é o ficheiro ‘psz_i’ e nele estão pares que a cada chave *itemID* associam uma lista de L valores, respectivos a cada z , dados pela Equação 5.10, onde o numerador provém do artefacto ‘sumUQStar_i’ e o denominador do artefacto ‘sumSUQStar_i’.

$$\frac{\sum_{u \in U} \frac{p(s|z) * p(z|u)}{\sum_{z'=1}^L p(s|z') * p(z'|u)}}{\sum_{s \in I} \sum_{u \in U} \frac{p(s|z) * p(z|u)}{\sum_{z'=1}^L p(s|z') * p(z'|u)}} \quad (5.10)$$

A implementação do PLSI tem a desvantagem de manipular vários ficheiros intermédios de dimensão considerável. Para colmatar este facto, ficheiros intermédios são eliminados assim que não sejam necessários por tarefas futuras. Na Figura 5.4 estão representadas os vários artefactos gerados pelas operações e a sua dependência. Os artefactos com fundo alaranjado são originados no passo E do algoritmo e os artefactos com fundo verde são originados no passo M.

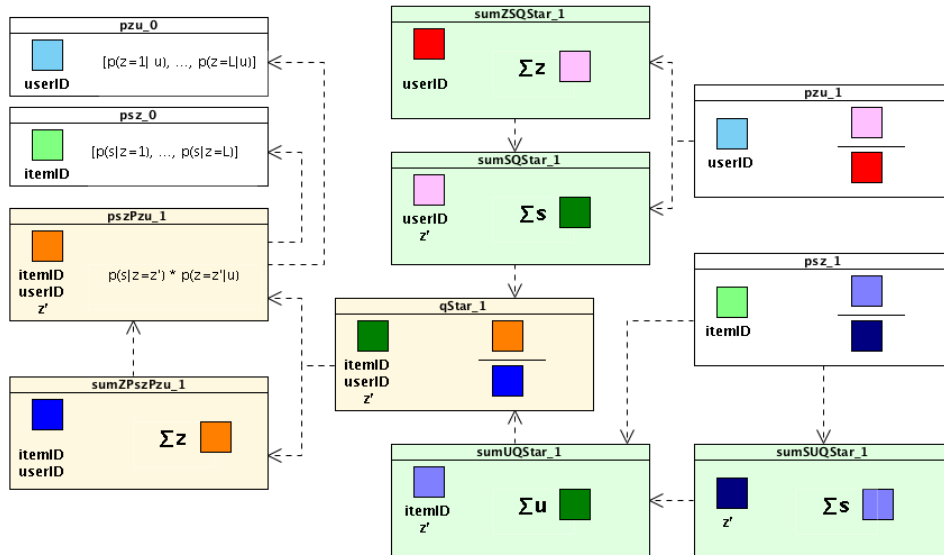


Figura 5.4: Diagrama de dependências dos artefactos gerados pela implementação do algoritmo EM

Por fim, é calculado o valor da perda logarítmica empírica $L(\theta)$ para cada par de ficheiros ‘pzu_i’ e ‘psz_i’, dado pela Equação 5.11, onde T é o conjunto de observações.

A lista de valores é transformada pela Ferramenta de Avaliação para o formato CSV.

$$L(\theta) = -\frac{1}{T} \sum_{t=1}^T \log \left(\sum_{z \in \{1..L\}} p(s_t|z) * p(z|u_t) \right) \quad (5.11)$$

Todas as operações são feitas com a classe `BigDecimal` do J2SE, que permite cálculos de precisão arbitrária, evitando desta forma os *overflows* que ocorriam nas grandes somas usando a primitiva `double`. A serialização dos números para a passagem dos valores entre as operações é feita por texto, imprimindo o número e reinterpretando-o na leitura.

5.6 Ferramenta de Avaliação

A Ferramenta de Avaliação percorre os vários pares de conjuntos treino+teste, invoca a ferramenta de aprendizagem e processa os artefactos por ela gerados (*clusters* e métricas). As métricas relacionadas com os algoritmos de *clustering* são transformadas em ficheiros CSV e a análise destas é feita recorrendo a uma folha de cálculo (ver Figura 5.5).

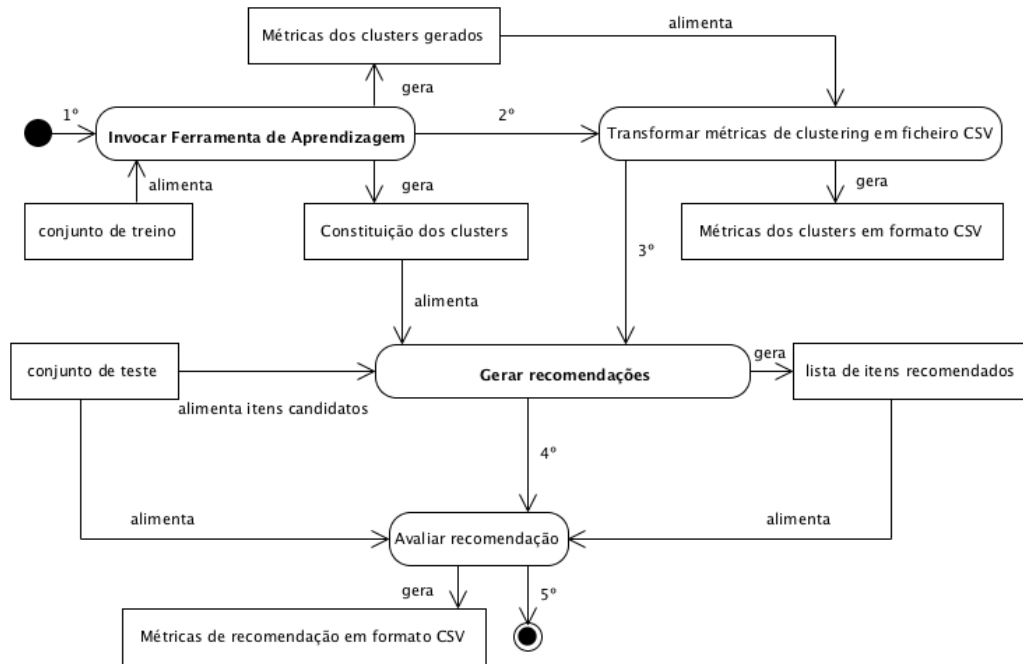


Figura 5.5: Diagrama de actividades da Ferramenta de Avaliação.

As métricas dos *clusters* estão divididas em dois artefactos: para *clusters* MinHash é retido o número de *clusters* gerados, o número de utilizadores em cada e para cada utilizador, o número médio de *clusters* a que pertence. Para os *clusters* PLSI é retido o valor da verossimilhança total do modelo, a iteração e o número de classes latentes.

Na Secção 6.1 encontra-se a análise dos valores que resultaram dos ensaios efectuados.

De seguida, geram-se recomendações para os utilizadores que constam do conjunto de teste. A geração de recomendações recorre directamente ao motor de personalização presente no Servidor de Personalização, evitando o *overhead* do pedido por REST ao servidor. O motor de personalização encontra-se embutido no artefacto *jar* da ferramenta de avaliação e não usa a *cache* descrita no Servidor de Personalização. Esta acção é feita considerando apenas os itens presentes no conjunto de teste como itens candidatos a recomendação.

A lista de recomendações gerada está ordenada por *score*. A acção de avaliação considera esta lista e calcula a precisão, a cobertura e a medida F (ver métricas na Secção 2.4). Estas métricas são armazenadas num ficheiro CSV de forma a poderem ser analisadas numa folha de cálculo. A avaliação é feita através de duas operações MapReduce:

- *GenerateCurrentState*, que considera os dados de treino e os *clusters* dele obtidos e recria um ficheiro com estrutura análoga às tabelas HTable, contendo as contagens de visualização por *cluster* e por utilizador bem como a co-visitação de itens.
- *GenerateRecommendations*, que usa os ficheiros criados na operação anterior e o conjunto de teste com o motor de recomendação, calculando o *score* de cada item candidato presente no conjunto de teste para cada utilizador, também presente no conjunto, tendo como base a informação de contagens resultante da operação anterior.

Na Secção 6.2 encontram-se a análise dos valores que resultaram dos ensaios efectuados à qualidade de recomendação.

Esta ferramenta é invocada com parâmetros que indicam o caminho para o conjunto de treino e de teste, os parâmetros de ambos algoritmos de *clustering* (ver Ferramenta de Aprendizagem na Secção 5.5) e com o peso a atribuir a cada *score* calculado na personalização.

Detalhes de Implementação

Capítulo 6

Ensaaios e Avaliação

Neste capítulo são apresentados os resultados dos ensaios realizados ao desempenho do algoritmo MinHash, ensaios à qualidade de recomendação e à capacidade de resposta dos vários componentes. Para cada teste realizado descreve-se sucintamente a metodologia e avaliam-se os resultados.

6.1 Técnicas de *Clustering*

As técnicas de *clustering* aplicadas têm um papel importante na qualidade da recomendação e, consoante os parâmetros, poderão dar origem a grupos mais refinados ou menos refinados. Realizaram-se ensaios para determinar qual o impacto dos parâmetros na geração destes grupos.

Os ensaios foram realizados num *cluster* computacional criado com alguns dos computadores de uma sala de trabalho da FEUP. Por este motivo, o *cluster* não estava permanentemente disponível. O *cluster* era constituído por um mínimo de 2 nós, chegando algumas vezes a um total de 50. Cada nó tem 3,4 GB de memória RAM e um processador Intel Core2 Quad Q9300 a 2,5 GHz, contendo 4 *cores*. Cada nó tinha como sistema operativo um Linux 2.6.32 e tinha aproximadamente 17 GB de armazenamento disponível. Cada bloco de ficheiro presente no HDFS estava redundantemente replicado duas vezes e todos os nós eram ligados por uma rede Ethernet de 1 Gb/s.

6.1.1 Técnica MinHash

A técnica MinHash foi aplicada ao conjunto de dados NF1T, variando o número de funções de dispersão (*nh*) e o número de grupos chave que identificavam unicamente cada *cluster* (*kg*).

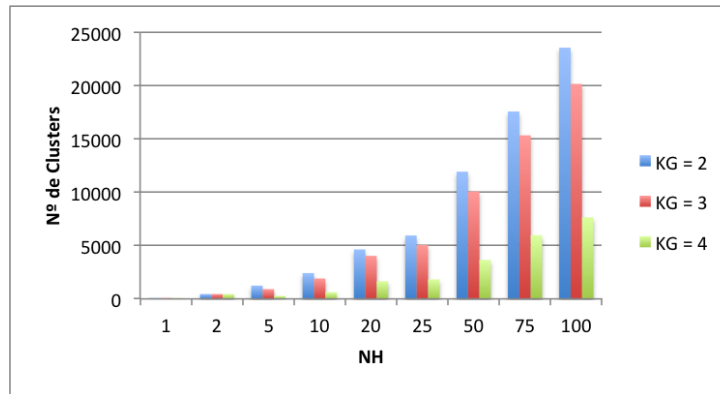


Figura 6.1: Número de *clusters* gerados, variando parâmetros *nh* e *kg*

A Figura 6.1 contém o número de grupos gerados pela execução, variando os parâmetros. Verifica-se que um número baixo de grupos de chave de identificação (*kg*) não aumentam a colisão de identificadores quando o número de funções de dispersão aplicadas é baixo (1 e 2). À medida que *nh* aumenta, evidencia-se uma maior colisão, crescente com *kg*. Em todas as execuções só era considerado um *cluster* se contivesse no mínimo 5 utilizadores. Esta restrição permite a geração de *clusters* minimamente coesos e evita a geração de muitos grupos pequenos, que não é favorável na complexidade de computação de recomendação em tempo real.

Na Figura 6.2 ilustra-se a evolução da cobertura da comunidade de utilizadores pelos *clusters* gerados, representando o número de utilizadores que pertencem a um ou mais *clusters*. Apenas foram considerados os utilizadores que tenham visualizado no mínimo dois itens diferentes, critério que é sempre verdadeiro em toda a extensão do conjunto NF1T.

Verifica-se que a cobertura total só se dá quando $nh \geq 10$ e que, comparando com a Figura 6.1, *clusters* identificados por mais do que um grupo de chaves têm um maior rácio entre a cobertura do conjunto de utilizadores e a sua quantidade.

Na Figura 6.3 apresenta-se o número médio de utilizadores por *cluster*. Pelo gráfico é possível identificar que *kg* é o principal parâmetro de influência desta métrica. Os resultados de $kg = 4$ dão uma cobertura aceitável com relativamente poucos grupos, constituídos por relativamente poucos utilizadores, tornando-se mais refinados quanto à relação que representam.

A Figura 6.4 mostra que o desvio padrão do número de utilizadores por *cluster* é mais baixo para $kg = 4$, significando que a constituição dos vários grupos é mais homogênea.

A Figura 6.5 expõe os valores do número de grupos a que um utilizador pertence. Destaca-se uma diferença de declive das rectas para os vários parâmetros *kg*, que evidencia a maior colisão da função de dispersão para *kg* maiores, dando origem a *clusters* mais refinados.

Ensaio e Avaliação

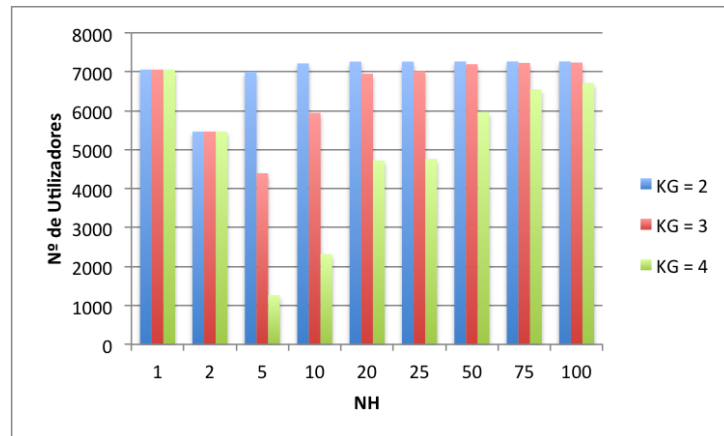


Figura 6.2: Gráfico de número de utilizadores pertencentes aos *clusters*, variando parâmetros nh e kg

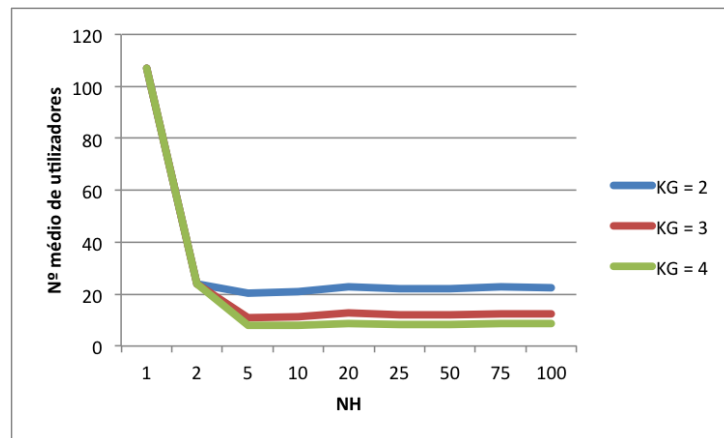


Figura 6.3: Gráfico representando o número médio de utilizadores por *cluster* gerado variando parâmetros nh e kg

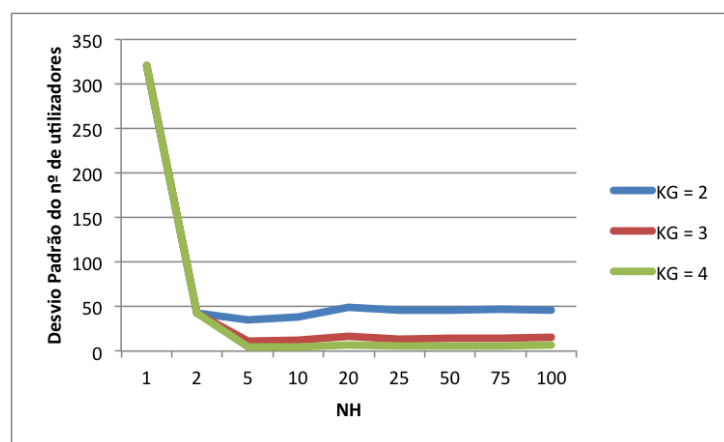


Figura 6.4: Gráfico representando o desvio padrão do número de utilizadores por *cluster* gerado variando parâmetros nh e kg

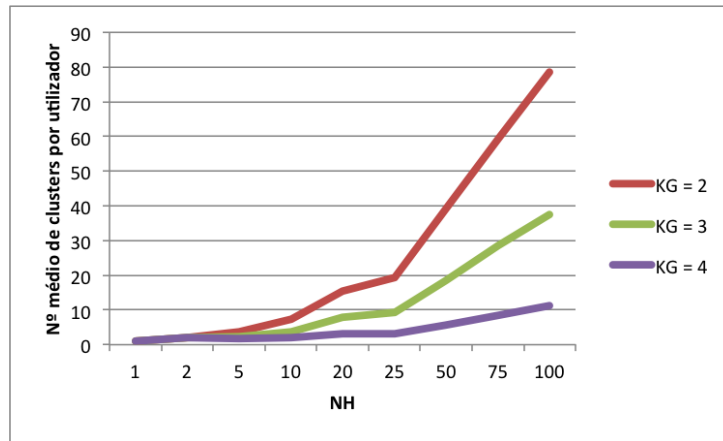


Figura 6.5: Gráfico representando o número de *clusters* a que um utilizador pertence, variando parâmetros *nh* e *kg*

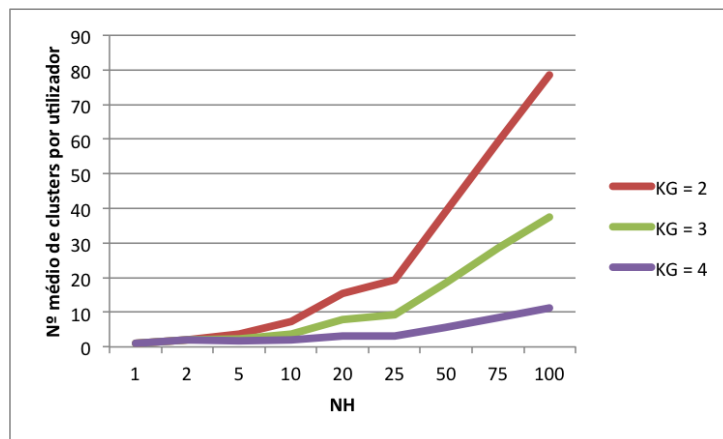


Figura 6.6: Gráfico representando o desvio padrão do número de *clusters* a que um utilizador pertence, variando parâmetros *nh* e *kg*

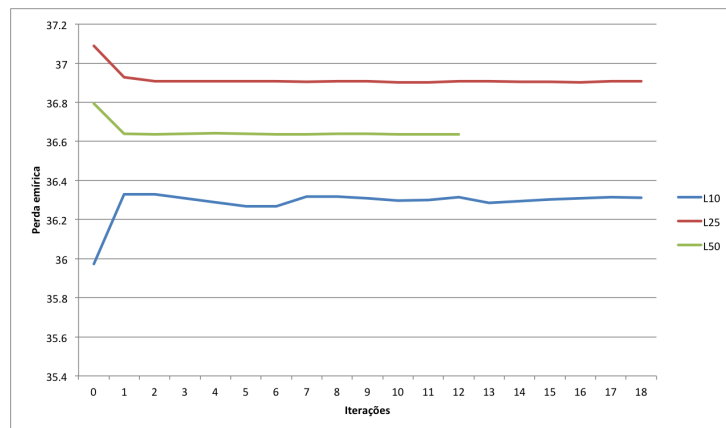


Figura 6.7: Gráfico com perda empírica para um número de classes latentes iguais a 10, 25 e 50

Complementando a análise, a Figura 6.6 demonstra que o número de *clusters* de cada utilizador não é tão uniforme, significando que existem alguns utilizadores inseridos em vários grupos e outros que se identificam com um conjunto menor.

6.1.2 Probabilistic Latent Semantic Indexing

Esta técnica foi aplicada várias vezes ao conjunto de dados NF1, resultando uma malha de pontos que variam com o número de classes latentes do modelo e com o número de iterações, ilustradas nas Figuras 6.7 e 6.8.

Os ensaios realizados com a técnica PLSI não foram conclusivos, pelo facto da perda empírica estabilizar logo nas primeiras iterações. A disparidade de valores entre as duas figuras revelam também que o número de classes latentes usadas (L) não influencia os valores da perda de uma forma estável.

Ensaaios com um número maior de classes latentes não foram conduzidos devido às

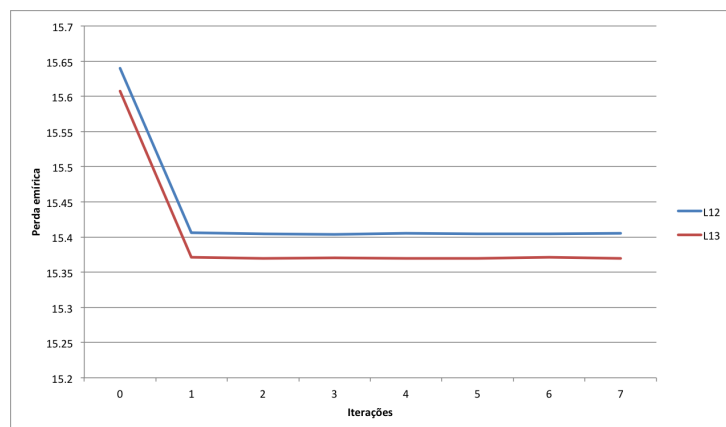


Figura 6.8: Gráfico com perda empírica para um número de classes latentes iguais a 12 e 13

limitações da infra-estrutura, nomeadamente nós disponíveis e capacidade de armazenamento em cada um.

6.2 Qualidade de Recomendação

As experiências realizadas com os algoritmos de filtragem colaborativa consideraram apenas as observações, ignorando qualquer tipo de informação adicional como a hora da observação, dados demográficos do utilizador ou meta-informação relacionada com o conteúdo.

As métricas aplicadas foram a precisão, a cobertura e a medida F (descritas na Secção 2.4), que combina as duas. O cálculo de recomendação é orientado a um dado utilizador. Assim, a precisão é a fracção dos objectos recomendados que consta na lista de itens visualizados pelo utilizador no conjunto de teste. Por sua vez, a cobertura (*recall*) é a fracção das visualizações do utilizador em questão, presentes no conjunto de teste, que foram recomendados.

Foram realizados ensaios para cada algoritmo, variando alguns parâmetros seleccionados dos ensaios individuais (ver Secção 6.1).

Começa-se por comparar a recomendação originada pelas várias partições dos utilizadores, utilizando o algoritmo MinHash. A metodologia seguida simula o cálculo efectuado pelo Servidor de Personalização através de um processo MapReduce. Para um dado utilizador u presente no conjunto NF1P é calculada a sua afinidade com um item candidato c por $\sum_{u_i} w(u, u_i)$, onde u_i são todos os utilizadores que visualizaram o item c , $w(u, u_i)$ é o número de *clusters* MinHash que u e u_i têm em comum, normalizado para que a soma seja igual a 1 para todos os utilizadores u_i .

O processo MapReduce começa por mapear o conjunto NF1P para pares utilizador-item e por mapear os *clusters* para o formato utilizador-*cluster*. A operação Reduce receberá listas contendo, para cada utilizador, os objectos de conteúdo que ele visualizou e os identificadores dos *clusters* a que o mesmo pertence. Para cada item emite um par cujo valor é a lista concatenada de ‘clusterID-userID’.

O resultado é passado novamente a uma operação Reduce que terá, para cada item, um conjunto de listas de *clusters*, cada uma delas representando um utilizador que viu aquele mesmo item. Para cada par do conjunto, é calculada a intersecção dos *clusters*, que se pode traduzir como sendo o número de *clusters* que os utilizadores respectivos têm em comum. A contagem é feita para todos os utilizadores e, por fim, é emitido o par ‘(utilizador,item)- $\frac{\text{contagem}}{\#\text{contagens}}$ ’ contendo como valor a normalização de contagens de *clusters* em comum com todos os outros utilizadores que viram aquele item.

Com base nos valores obtidos, ordenaram-se os itens candidatos por *score* e calculou-se a precisão e a cobertura (*recall*). A selecção foi feita com um limiar de $10^{-1.5}$, sendo que os objectos de conteúdo pontuados acima eram recomendados. A variação do limiar

influencia a precisão-recuperação. O limiar foi escolhido empiricamente, de forma a obter uma precisão alta, comportamento desejável de um recomendador *top-N*.

A Figura 6.9 mostra o gráfico de precisão-recuperação, numa escala 0-100, para os *clusters* obtidos com 3 grupos chave como identificador ($kg = 3$). Apenas foram considerados os ensaios corridos com o número de funções de dispersão (nh) altas, pois eram aqueles que cobriam a totalidade dos utilizadores, ou seja, não havia nenhum utilizador sem *cluster* associado. A Figura 6.10 contém a respectiva medida F.

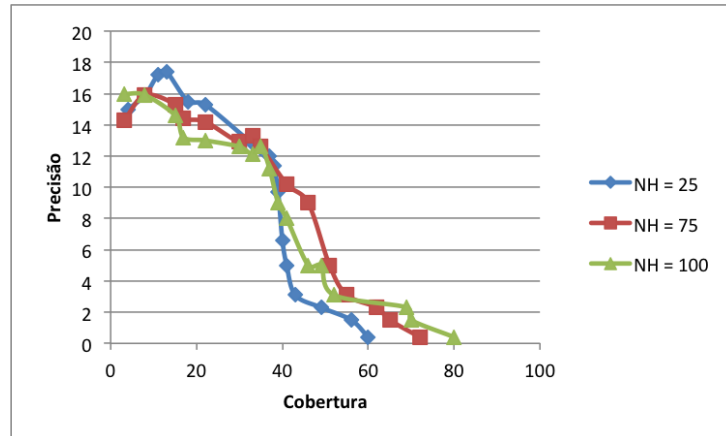


Figura 6.9: Gráfico de precisão-cobertura, no conjunto de teste NF1P, usando *clusters* MinHash com $kg = 3$

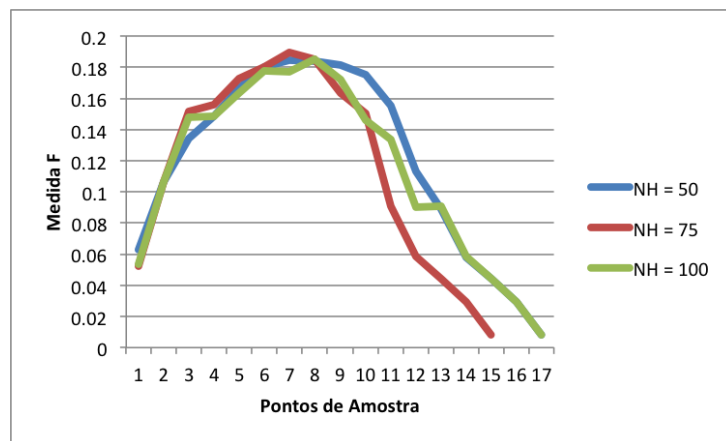


Figura 6.10: Gráfico representando a medida F, no conjunto de teste NF1P, usando *clusters* MinHash com $kg = 3$

Por sua vez, a Figura 6.11 mostra o gráfico de precisão-recuperação para $kg = 4$. Apesar de existirem utilizadores sem *cluster* associados, a precisão é bastante melhor. A Figura 6.12, que contém a respectiva medida F, evidencia o facto desta partição de utilizadores ser menos extensa, mas mais precisa na qualidade de recomendação.

Como vimos na Secção 6.1, estes *clusters* são mais semelhantes em tamanho, o que pode levar a um *score* mais alto para todos os itens considerados e, desta forma, aumentar

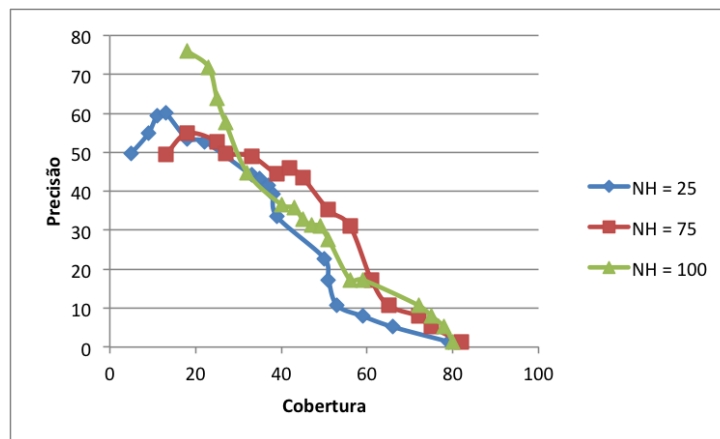


Figura 6.11: Gráfico de precisão-cobertura, no conjunto de teste NF1P, usando *clusters* MinHash com $k_g = 4$

a precisão. Por outras palavras, os *clusters* gerados pelo MinHash com estes parâmetros representam gostos mais concretos.

Uma comparação directa não pode ser feita com os resultados apresentados em [DDGR07] pelo facto de não usarem o mesmo conjunto de dados e de não revelarem quais os parâmetros usados em cada técnica, nomeadamente o *threshold* aplicado e o número de funções de dispersão e de grupos de chave.

6.3 Desempenho dos Componentes

Foram realizados testes de carga aos componentes *online* de forma a identificar quais as limitações. Os componentes *offline*, nomeadamente a Ferramenta de Aprendizagem, não tiveram qualquer tipo de experiência de desempenho uma vez que a infra-estrutura usada para a sua execução não estava permanentemente disponível.

No entanto, observou-se uma computação intensiva nos ficheiros intermédios do PLSI, devido ao seu tamanho e à sua densidade pelos vários nós. Verificou-se que o número de nós na operação Map influencia bastante o tempo total de execução da tarefa, devido ao acesso ao disco em paralelo e a uma distribuição menos pesada.

Os testes de carga ao Servidor de Estatísticas e ao Servidor de Personalização foram feitos recorrendo à ferramenta ApacheBench¹. Esta ferramenta permite gerar tráfego HTTP, simulando pedidos simultâneos. Existe, no entanto, a limitação de invocar apenas um endereço. Foi usado um endereço genérico, que por sua vez accionava um pedido com parâmetros aleatórios, dentro do intervalo de utilizadores e do intervalo de objectos de conteúdo. O *overhead* introduzido pela geração uniforme dos parâmetros do pedido é constante, pelo que poderá ser desprezado.

¹<http://httpd.apache.org/docs/2.0/programs/ab.html>

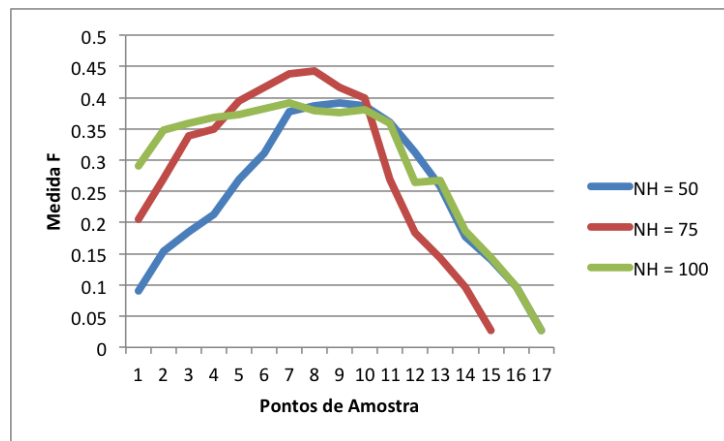


Figura 6.12: Gráfico representando a medida F, no conjunto de teste NF1P, usando *clusters* MinHash com $kg = 4$

Todos os componentes *online* usam o servidor de HTTP *Jetty*².

Os testes realizados ao Servidor de Estatísticas foram:

- SE1, pedidos simultâneos de vários utilizadores que vêem um conjunto reduzido de itens;
- SE2, pedidos simultâneos de alguns utilizadores que vêem vários itens diferentes.

O ensaio SE1 pressupõe uma visualização de televisão menos impulsiva, com pouca pesquisa pela grelha de canais. Por outro lado, o ensaio SE2 pretende provar como se comporta uma instância deste componente perante um conjunto de utilizadores a fazer *zapping*.

O tempo do pedido HTTP ronda os 3 ms, tanto para o teste SE1 como para o SE2, numa infra-estrutura em que o HBase tenha as chaves bem divididas pelos vários *RegionServers*. Más experiências ocorreram no ensaio SE2, pois a divisão das chaves dos itens não explorava os vários nós disponíveis e sobrecarregava um único nó (que tinha a seu cargo todas as chaves). A Figura 6.13 contém o número de pedidos por segundo feitos e o respectivo tempo de resposta. O ensaio foi conduzido ao longo de um período de 100 s, usando 10 geradores de tráfego concorrentes. Nota-se num tempo de resposta baixo e constante até ao pedido 35 000, aproximadamente. Depois existe um aumento abrupto do tempo de pedido, devido à estagnação dos vários *RegionServers*. Isto acontece devido aos *Locks* de linha no momento da escrita.

Uma vez que em cada pedido são realizados vários *locks* e o processamento é assíncrono, o Servidor de Estatísticas retorna com o código HTTP 200 OK ao fim de pouco tempo. Todos os pedidos de escrita ficam armazenados numa fila que é executada ao fim de 1 s. Ao fim de alguns pedidos, a fila a executar a cada segundo vai aumentando. No

²<http://jetty.codehaus.org/jetty/>

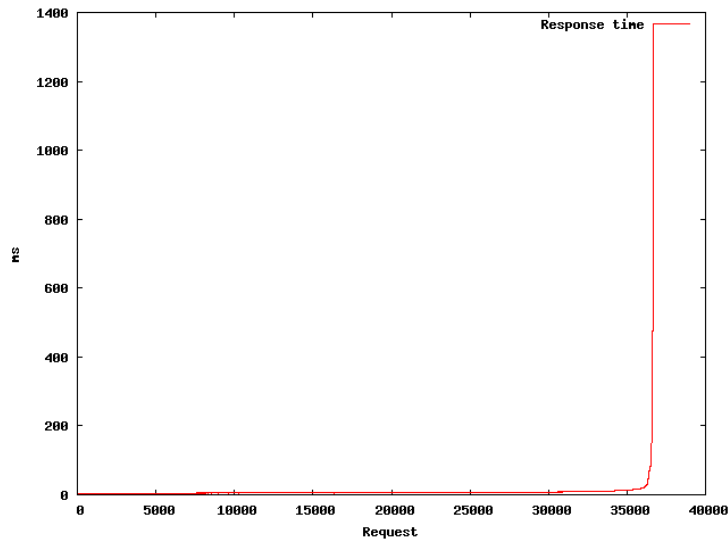


Figura 6.13: Tempos de resposta do ensaio SE1, ao longo de 100 s

caso limite de execução, o processamento total de uma fila faz com que existam demasiados *Locks* em cada *RegionServer*, levando a um *denial-of-service*.

Uma solução, não adoptada, seria limitar o número de pedidos de cada instância deste componente. Isto faria com que a fila a executar em cada um não superasse um determinado tamanho. Usando um balanceador de pedidos que faria a distribuição do tráfego por parâmetro de pedido, faria com que a execução das filas das várias instâncias não colidisse com *Locks*. Outra opção seria colocar outro tipo de componente de persistência.

Foram também realizados os seguintes testes ao Servidor de Personalização:

- SP1, pedidos de recomendação para vários utilizadores sem recorrer a *cache*.
- SP2, pedidos de recomendação para vários utilizadores usando *cache* para os valores obtidos do HBase;

Os resultados do ensaio SP1 e SP2 poderão determinar se compensa ter várias instâncias deste componente, cada um responsável por um intervalo de utilizadores, ou se compensa ter várias instâncias indiferenciadas, onde vão parar os vários pedidos por balanceamento de pedidos.

Na Figura 6.14 estão os resultados do ensaio SP1. O resultado é o esperado: o número de pedidos concorrentes leva ao rápido esgotamento dos recursos resultando numa menor capacidade de resposta.

Foram realizados testes de forma a medir o impacto da cache implementada. A *cache* foi definida com um factor de armazenamento de 4 MB. Os resultados estão na Figura 6.15. É visível uma evolução no tempo de resposta e pequenos períodos em que a *cache* parece durar mais. Neste cenário é possível servir mais clientes por unidade de tempo, uma vez que tem um tempo de resposta significativamente mais baixo do que os

do ensaio SP1 para uma maior extensão de pedidos concorrentes (2000 pedidos de 100 clientes face aos 1000 de 100 clientes).

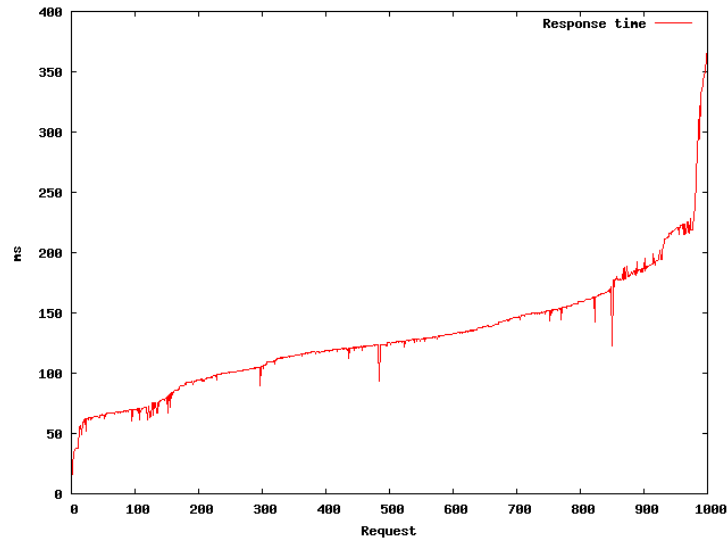


Figura 6.14: Tempos de resposta do ensaio SP1 realizado com 1000 pedidos de 100 clientes concorrentes

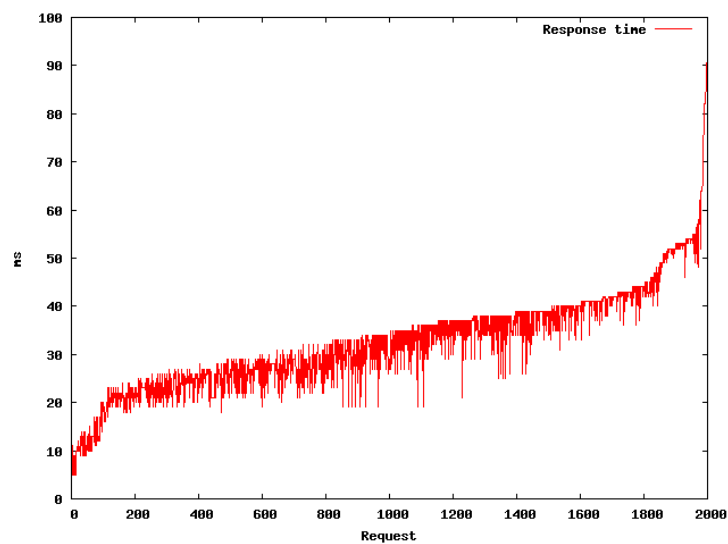


Figura 6.15: Tempos de resposta do ensaio SP2 realizado com 2000 pedidos de 100 clientes concorrentes

No entanto, uma actualização da *cache* por unidade de tempo poderá surtir melhor efeito do que uma baseada no seu tamanho, uma vez que poderá variar com o pedido (concretamente, os *clusters* usados no cálculo).

6.4 Conclusões

Conclui-se que a técnica MinHash tem um comportamento aceitável para o conjunto de dados NF1T. Na técnica MinHash os *clusters* originados com um parâmetro *kg* maior dão origem a grupos mais pequenos, com menos elementos mas mais homogêneos (mesmo número de utilizadores). Apesar de não terem uma cobertura tão grande da população de utilizadores, particionam melhor os utilizadores pelas várias comunidades, aumentando a qualidade de recomendação e diminuindo o tempo de processamento dos componentes *online*.

Conclui-se que a técnica PLSI requer a realização de ensaios mais extensos para o estudo do comportamento dos parâmetros. Para ensaios com um número de classes baixa (≤ 50), os resultados não indicam qual a melhor escolha de parâmetros a aplicar.

A recomendação usando o MinHash é bastante aceitável para o conjunto de teste NF1P, pois tem uma precisão alta, significando que a lista dos 3 melhores itens recomendados incide claramente nos gostos do utilizador. Este é um comportamento desejável pelo facto do utilizador não ter muito espaço no televisor para visualizar a lista de recomendações. Outras acções que considerem essa lista, como por exemplo a gravação prévia de um programa, apenas são viáveis com uma alta precisão devido à limitação de espaço de armazenamento para o efeito.

Nos componentes verifica-se um desempenho razoável, contabilizando alguns problemas no Servidor de Estatísticas. Este obstáculo poderá ser ultrapassado num sistema em produção através de um motor de persistência distribuído e tolerante a falha. Identifica-se ainda uma limitação clara do Servidor de Personalização que poderá ser facilmente ultrapassada através de um balanceador de pedidos (*load balancer*) e pela execução de várias instâncias desse mesmo componente.

Por último, apesar das técnicas terem sido aplicadas a um conjunto de dados com características seleccionadas para simular o cenário de televisão e a obtenção dos resultados ser satisfatória, é necessário aplicar a um conjunto de dados real para validar o seu desempenho num sistema em produção. Será também necessário adaptar os parâmetros ao conjunto de dados, através de ensaios semelhantes aos realizados.

Capítulo 7

Conclusão

7.1 Sumário

A recomendação de objectos de conteúdo não é uma operação trivial, podendo ser adoptadas diferentes abordagens. As abordagens tomadas consideraram apenas o comportamento passado de cada utilizador, embora fossem preparados dados de metac conteúdo para fazer um estudo de técnicas baseadas na composição do conteúdo.

Das técnicas aplicadas, podemos apenas concluir resultados satisfatórios para o MinHash. Este algoritmo particionou o universo dos utilizadores de forma fiel à similaridade de cada par de utilizadores, de forma a que o número de *clusters* em comum entre dois utilizadores distintos fosse proporcional ao coeficiente de Jaccard aplicado aos conjuntos de itens do histórico de visualização de cada um.

Da implementação de ambas as técnicas, conclui-se que escalam bem com a implementação descrita, usando o paradigma distribuído MapReduce.

A nível de arquitectura existe uma boa orquestração entre os vários componentes. A arquitectura foi feita separando os papéis de cada componente de forma a que, quando um componente falha, outro continua a realizar a sua tarefa. Foi ainda separado em dois tipos de componentes: os *online*, responsáveis por responder a pedidos num curto espaço de tempo, e os *offline* que são accionados para processar dados e posteriormente alimentar as ferramentas *online*.

A nível de tecnologia, conclui-se que o motor de persistência poderá não ser o mais indicado para um sistema em produção devido ao *bottleneck* sentido na escrita de valores pelo componente de aquisição de estatísticas.

Deixa-se ainda em aberto a aplicação do sistema a outros domínios, uma vez que as técnicas são agnósticas à natureza dos dados, tratando-os como observações utilizador-item. A versatilidade das técnicas poderá tratar interacções entre utilizadores e outro tipo de conteúdos, embora seja necessário ajustar os parâmetros.

Pode-se concluir que, no geral, foi um trabalho ambicioso por aliar uma forte componente de aprendizagem automática a uma arquitectura distribuída com processos *online* e *offline*. A escolha de tecnologia foi vital para a concretização dos objectivos propostos, nomeadamente o tempo de resposta do sistema, a tolerância à falha e a taxa de novos conteúdos inerente ao domínio de aplicação.

7.2 Trabalho Futuro

Quanto ao processo de recomendação, identifica-se como trabalho futuro a aplicação de técnicas baseadas em árvore de decisão que permitam a combinação dos *scores* calculados conforme o grau de desenvolvimento do perfil do utilizador. O actual sistema, baseado em filtragem colaborativa, pode ainda ser combinado com um sistema baseado em conteúdo, aproveitando os vectores TF-IDF criados pela ferramenta de indexação. Esta combinação permitiria recomendar melhor uma determinada característica do objecto de conteúdo aos utilizadores fieis como, por exemplo, a equipa que está a jogar determinado jogo.

Na implementação do sistema *offline* dever-se-á utilizar um sistema de monitorização que permita recolher métricas de execução dos vários nós da infra-estrutura Hadoop. Estes dados poderão levar a uma configuração mais adequada ao *hardware* existente e tirar o máximo partido de todos os recursos disponíveis.

Nos processos MapReduce pode-se ainda utilizar um algoritmo de compressão que permita a compressão e a descompressão em paralelo, como o LZ0, aplicado no Twitter¹. Desta forma, o volume de dados a transferir entre cada máquina reduziria e as unidades de armazenamento seriam melhor aproveitadas, introduzindo em contrapartida algum *overhead* na compressão.

Na implementação dos sistemas *online*, o uso de servidores HTTP assíncronos e baseados em eventos como o Netty² poderia aumentar a capacidade individual de cada instância. Deixa-se como trabalho futuro o estudo do comportamento destes componentes numa infra-estrutura com um balanceador de pedidos. Uma melhoria a introduzir no Servidor de Personalização, é a actualização dos parâmetros de combinação de *scores* dinamicamente.

¹<http://www.cloudera.com/blog/2009/06/parallel-lzo-splittable-compression-for-hadoop/>

²<http://www.jboss.org/netty>

Referências

- [AT05] Gediminas Adomavicius e Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17:734–749, June 2005.
- [BCFM98] Andrei Z. Broder, Moses Charikar, Alan M. Frieze e Michael Mitzenmacher. Min-wise independent permutations (extended abstract). In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 327–336, New York, NY, USA, 1998. ACM.
- [BEKR07] Shlomo Berkovsky, Yaniv Eytani, Tsvi Kuflik e Francesco Ricci. Enhancing privacy and preserving accuracy of a distributed collaborative filtering. In *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, pages 9–16, New York, NY, USA, 2007. ACM.
- [BFPAGS⁺08] Yolanda Blanco-Fernandez, Jose J Pazos-Arias, Alberto Gil-Solla, Manuel Ramos-Cabrer e Martin Lopez-Nores. Providing entertainment by content-based filtering and semantic reasoning in intelligent recommender systems. *Consumer Electronics IEEE Transactions on*, 54(2):727–735, 2008.
- [BGMS05] Derek Bridge, Mehmet H. Göker, Lorraine McGinty e Barry Smyth. Case-based recommender systems. *Knowl. Eng. Rev.*, 20:315–320, September 2005.
- [BHK98] John S. Breese, David Heckerman e Carl Myers Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G. F. Cooper e S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.
- [BKV07] Robert Bell, Yehuda Koren e Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 95–104, New York, NY, USA, 2007. ACM.
- [BNJ03] David M. Blei, Andrew Y. Ng e Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

REFERÊNCIAS

- [Bro97] A. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES '97, pages 21–, Washington, DC, USA, 1997. IEEE Computer Society.
- [Bur00] Robin Burke. Knowledge-based Recommender Systems. In *Encyclopedia of Library and Information Systems*, volume 69, 2000.
- [Bur02] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, November 2002.
- [CDG⁺08] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes e Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.*, 26(2):1–26, June 2008.
- [Coh97] Edith Cohen. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. Syst. Sci.*, 55:441–453, December 1997.
- [CSA⁺08] Ivan Cantador, Martin Szomszor, Harith Alani, Miriam Fernandez e Pablo Castells. Enriching ontological user profiles with tagging history for multi-domain recommendations. In *1st International Workshop on Collective Semantics: Collective Intelligence & the Semantic Web (CISWeb 2008)*, June 2008.
- [DDGR07] Abhinandan S. Das, Mayur Datar, Ashutosh Garg e Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA, 2007. ACM.
- [DG04] Jeffrey Dean e Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *OSDI*, page 13, 2004.
- [dGLSB08] Marco de Gemmis, Pasquale Lops, Giovanni Semeraro e Pierpaolo Basile. Integrating tags in a semantic content-based recommender. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 163–170, New York, NY, USA, 2008. ACM.
- [DI06] Jorg Diederich e Tereza Iofciu. Finding communities of practice from user profiles based on folksonomies. 2006.
- [DLR77] A. P. Dempster, N. M. Laird e D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.
- [FB08] A. Felfernig e R. Burke. Constraint-based recommender systems: technologies and research issues. In *Proceedings of the 10th international conference on Electronic commerce*, ICEC '08, pages 3:1–3:10, New York, NY, USA, 2008. ACM.

REFERÊNCIAS

- [FFJZ06] Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach e Markus Zanker. An integrated environment for the development of knowledge-based recommender applications. *Int. J. Electron. Commerce*, 11:11–34, December 2006.
- [FPRS07] Francois Fouss, Alain Pirotte, Jean-Michel Renders e Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. on Knowl. and Data Eng.*, 19:355–369, March 2007.
- [GH05] Scott Golder e Bernardo A. Huberman. The Structure of Collaborative Tagging Systems. *Journal of Information Science*, 2005.
- [GM05] Thomas George e Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the Fifth IEEE International Conference on Data Mining, ICDM '05*, pages 625–628, Washington, DC, USA, 2005. IEEE Computer Society.
- [GSK⁺99] Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker e John Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence, AAAI '99/IAAI '99*, pages 439–446, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.
- [HKTR04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen e John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22:5–53, January 2004.
- [Hof03] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, SIGIR '03*, pages 259–266, New York, NY, USA, 2003. ACM.
- [Hof04] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22:89–115, January 2004.
- [HPK10] Günther Höbling, Michael Pleschgattertnig e Harald Kosch. Personal tv a tv recommendation system using program metadata for content filtering. *Multimedia Tools and Applications*, 46(2):259–288, 01 2010.
- [Kor09] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '09*, pages 447–456, New York, NY, USA, 2009. ACM.
- [LGS11] Pasquale Lops, Marco Gemmis e Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Francesco Ricci, Lior Rokach, Bracha Shapira e Paul B Kantor, editors, *Recommender*

REFERÊNCIAS

- Systems Handbook*, pages 73–105. Springer US, 2011. 10.1007/978-0-387-85820-3_3.
- [LSY03] G. Linden, B. Smith e J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76 – 80, 2003.
- [Luc10] André Lucas. Recomendação de Programas de Televisão. Master’s thesis, Instituto Superior Técnico, Lisboa, 2010.
- [MBBW07] Bamshad Mobasher, Robin Burke, Runa Bhaumik e Chad Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Trans. Internet Technol.*, 7, October 2007.
- [MBS06] Bamshad Mobasher, Robin Burke e J. J. Sandvig. Model-based collaborative filtering as a defense against profile injection attacks. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, pages 1388–1393. AAAI Press, 2006.
- [MJSS01] Rajatish Mukherjee, Gerdur Jonsdottir, Sandip Sen e Partha Sarathi. Movies2go: an online voting based movie recommender system. In *Proceedings of the fifth international conference on Autonomous agents, AGENTS ’01*, pages 114–115, New York, NY, USA, 2001. ACM.
- [MKP03] Harry Mak, Irena Koprinska e Josiah Poon. Intimate: A web-based movie recommender using text categorization. In *Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence, WI ’03*, pages 602–605, Washington, DC, USA, 2003. IEEE Computer Society.
- [Mla99] D. Mladenic. Text-learning and related intelligent agents: a survey. *Intelligent Systems and their Applications, IEEE*, 14(4):44 –54, 1999.
- [MRS08] Christopher D. Manning, Prabhakar Raghavan e Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.
- [Per10] João Pereira. Sistema de recomendação TV. Master’s thesis, Faculdade de Engenharia da Universidade do Porto, 2010.
- [PRPT05] Manos Papagelis, Ioannis Rousidis, Dimitris Plexousakis e Elias Theoharopoulos. Incremental collaborative filtering for highly-scalable recommendation algorithms. In *Proceedings of the 15th International Symposium on Methodologies of Intelligent Systems (ISMIS’05)*, 2005.
- [SA06] James Salter e Nick Antonopoulos. Cinemascreen recommender agent: Combining collaborative and content-based filtering. *IEEE Intelligent Systems*, 21:35–41, January 2006.
- [SC00] B. Smyth e P. Cotter. A Personalized TV Listings Service for the Digital TV Age. *Journal of Knowledge-Based Systems*, 13(2-3):53–59, 2000.

REFERÊNCIAS

- [SG11] Guy Shani e Asela Gunawardana. Evaluating Recommendation Systems. In Francesco Ricci, Lior Rokach, Bracha Shapira e Paul B. Kantor, editors, *Recommender Systems Handbook*, chapter 8, pages 257–297. Springer US, Boston, MA, 2011.
- [SKKR00] B. Sarwar, G. Karypis, J. Konstan e J. Riedl. Application of dimensionality reduction in recommender systems—a case study, 2000.
- [SKKR02] Badrul Sarwar, George Karypis, Joseph Konstan e John Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28, 2002.
- [SMH07] Ruslan Salakhutdinov, Andriy Mnih e Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning, ICML '07*, pages 791–798, New York, NY, USA, 2007. ACM.
- [SPTH09] Reza Shokri, Pedram Pedarsani, George Theodorakopoulos e Jean-Pierre Hubaux. Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 157–164, New York, NY, USA, 2009. ACM.
- [VRY09] Benjamin Van Roy e Xiang Yan. Manipulation-resistant collaborative filtering systems. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 165–172, New York, NY, USA, 2009. ACM.
- [WdVR06] Jun Wang, Arjen P. de Vries e Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 501–508, New York, NY, USA, 2006. ACM.
- [WPLR06] Jun Wang, Johan Pouwelse, Reginald L. Lagendijk e Marcel J. T. Reinders. Distributed collaborative filtering for peer-to-peer file sharing systems. In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pages 1026–1030, New York, NY, USA, 2006. ACM.
- [ZWF⁺05] Sheng Zhang, Weihong Wang, James Ford, Fillia Makedon e Justin Pearlman. Using singular value decomposition approximation for collaborative filtering. In *Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*, pages 257–264, Washington, DC, USA, 2005. IEEE Computer Society.

REFERÊNCIAS

Anexo A

Fórmulas e Cálculos de Apoio

Este anexo contém algumas fórmulas que suportam cálculos realizados nas operações de predição de itens.

A.1 Medidas de Similaridade

Existem várias medidas de similaridade, que tomam dois vectores em consideração. Apesar de algumas traduzirem melhor a similaridade entre itens ou utilizadores, num determinado cenário de aplicação, normalmente opta-se pela que tem resultados satisfatórios face à sua complexidade de execução.

Distância angular ou medida do cosseno

$$\cos(x, y) = \frac{(x \bullet y)}{\|x\| \|y\|} \quad (\text{A.1})$$

Correlação de Pearson

$$Pearson(x, y) = \frac{\sum_{i \in I} (R_{x,i} - \bar{R}_x)(R_{y,i} - \bar{R}_y)}{\sqrt{\sum_{i \in I} (R_{x,i} - \bar{R}_x)^2} \sqrt{\sum_{i \in I} (R_{y,i} - \bar{R}_y)^2}} \quad (\text{A.2})$$

Coeficiente de Jaccard, que traduz a percentagem de elementos comuns em dois conjuntos X e Y

$$Jaccard(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \quad (\text{A.3})$$

Coeficiente de Tanimoto, que é a versão extendida do Coeficiente de Jaccard, incluindo no cálculo, para além da quantidade relativa de elementos diferentes, a diferença angular.

$$Tanimoto(X, Y) = \frac{X \cdot Y}{\|X\|^2 + \|Y\|^2 - X \cdot Y} \quad (\text{A.4})$$

Anexo B

Artefactos

B.1 Implementação das Funções de Dispersão

Foram implementadas três funções de dispersão: uma linear, uma polinomial e outra de Murmur. Segue-se o código da sua implementação, na linguagem Java.

B.1.1 Função de Dispersão Linear

```
static class LinearHash implements HashFunction {
    private final int seedA;
    private final int seedB;

    LinearHash(int seedA, int seedB) {
        this.seedA = seedA;
        this.seedB = seedB;
    }

    @Override
    public int hash(byte[] bytes) {
        long hashValue = 31;
        for (long byteVal : bytes) {
            hashValue *= seedA * byteVal;
            hashValue += seedB;
        }
        return Math.abs((int) (hashValue %
RandomUtils.MAX_INT_SMALLER_TWIN_PRIME));
    }
}
```

B.1.2 Função de Dispersão Polinomial

```
static class PolynomialHash implements HashFunction {
    private final int seedA;
    private final int seedB;
    private final int seedC;
```

```

PolynomialHash(int seedA, int seedB, int seedC) {
    this.seedA = seedA;
    this.seedB = seedB;
    this.seedC = seedC;
}

@Override
public int hash(byte[] bytes) {
    long hashValue = 31;
    for (long byteVal : bytes) {
        hashValue *= seedA * (byteVal >> 4);
        hashValue += seedB * byteVal + seedC;
    }
    return Math
        .abs((int) (hashValue %
RandomUtils.MAX_INT_SMALLER_TWIN_PRIME));
}
}

```

B.1.3 Função de Dispersão Murmur

```

static class MurmurHashWrapper implements HashFunction {
    private final int seed;

    MurmurHashWrapper(int seed) {
        this.seed = seed;
    }

    @Override
    public int hash(byte[] bytes) {
        long hashValue = MurmurHash.hash64A(bytes, seed);
        return Math.abs((int) (hashValue %
RandomUtils.MAX_INT_SMALLER_TWIN_PRIME));
    }
}

public final class MurmurHash {
    private MurmurHash() {
    }

    public static int hash(byte[] data, int seed) {
        return hash(ByteBuffer.wrap(data), seed);
    }

    public static int hash(byte[] data, int offset, int length, int seed) {
        return hash(ByteBuffer.wrap(data, offset, length), seed);
    }
}

```

```

}

public static int hash(ByteBuffer buf, int seed) {
    // save byte order for later restoration
    ByteOrder byteOrder = buf.order();
    buf.order(ByteOrder.LITTLE_ENDIAN);

    int m = 0x5bd1e995;
    int r = 24;
    int h = seed ^ buf.remaining();

    while (buf.remaining() >= 4) {
        int k = buf.getInt();

        k *= m;
        k ^= k >>> r;
        k *= m;
        h *= m;
        h ^= k;
    }

    if (buf.remaining() > 0) {
        ByteBuffer finish = ByteBuffer.allocate(4)
.order(ByteOrder.LITTLE_ENDIAN);
        finish.put(buf).rewind();
        h ^= finish.getInt();
        h *= m;
    }

    h ^= h >>> 13;
    h *= m;
    h ^= h >>> 15;

    buf.order(byteOrder);
    return h;
}

public static long hash64A(byte[] data, int seed) {
    return hash64A(ByteBuffer.wrap(data), seed);
}

public static long hash64A(byte[] data, int offset, int length, int seed) {
    return hash64A(ByteBuffer.wrap(data, offset, length), seed);
}

public static long hash64A(ByteBuffer buf, int seed) {

```

```

ByteOrder byteOrder = buf.order();
buf.order(ByteOrder.LITTLE_ENDIAN);

long m = 0xc6a4a7935bd1e995L;
int r = 47;

long h = seed ^ (buf.remaining() * m);

while (buf.remaining() >= 8) {
    long k = buf.getLong();
    k *= m;
    k ^= k >>> r;
    k *= m;
    h ^= k;
    h *= m;
}

if (buf.remaining() > 0) {
    ByteBuffer finish = ByteBuffer.allocate(8)
.order(ByteOrder.LITTLE_ENDIAN);
    finish.put(buf).rewind();
    h ^= finish.getLong();
    h *= m;
}

h ^= h >>> r;
h *= m;
h ^= h >>> r;

buf.order(byteOrder);
return h;
}
}

```

B.2 Esquemas XSD

Existem dois esquemas XSD utilizados no sistema: o primeiro é referente a uma lista de recomendações de objectos de conteúdo, a segunda é referente à representação de um item.

B.2.1 Esquema XSD de Conteúdo

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://iex.ptin.pt/ctx/content"
    xmlns:tns="http://iex.ptin.pt/ctx/content">

```

```

<xsd:element name="content" type="tns:ContentType" />

<xsd:complexType name="SourceType">
  <!-- URI fonte do proprio conteudo -->
  <xsd:attribute name="src" type="xsd:string"
    use="required" />
  <!-- qualidade do conteudo e' opcional
    e ajuda a decidir que URI escolher em cenario
    de streaming com base na largura de banda
    disponivel -->
  <xsd:attribute name="quality" type="xsd:string"
    use="optional" />
</xsd:complexType>

<xsd:complexType name="ContentType">
  <xsd:sequence>
    <xsd:element name="tags" minOccurs="0" maxOccurs="1">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="tag"
            type="tns:TagType"
            minOccurs="1"
            maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="people" minOccurs="0" maxOccurs="1">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="person"
            type="tns:PersonType"
            minOccurs="1"
            maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="sources"
      minOccurs="0"
      maxOccurs="1">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="source"
            type="tns:SourceType"
            minOccurs="1"

```

```

        maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>

<!-- descricao / sinopse -->
<xsd:element name="description"
    type="xsd:string" minOccurs="0"
    maxOccurs="1" />
<!-- titulo do programa -->
<xsd:element name="title" type="xsd:string"
minOccurs="1" maxOccurs="1" />
<!-- titulo do episodio -->
<xsd:element name="episode_title"
    type="xsd:string" minOccurs="0"
    maxOccurs="1" />
<!-- numero do episodio -->
<xsd:element name="episode_number"
    type="xsd:integer" minOccurs="0"
    maxOccurs="1" />
<!-- numero de temporada -->
<xsd:element name="season_number"
    type="xsd:integer" minOccurs="0"
    maxOccurs="1" />
<!-- duracao em minutos -->
<xsd:element name="duration"
    type="xsd:integer" minOccurs="0"
    maxOccurs="1" />
<!-- hora de inicio -->
<xsd:element name="start_at"
    type="xsd:dateTime" minOccurs="0"
    maxOccurs="1" />
<!-- hora de fim -->
<xsd:element name="end_at"
    type="xsd:dateTime" minOccurs="0"
    maxOccurs="1" />
<!-- canal -->
<xsd:element name="channel"
    type="xsd:string" minOccurs="0"
    maxOccurs="1" />
<!-- faixa etaria -->
<xsd:element name="age_classification"
    type="xsd:string" minOccurs="0"
    maxOccurs="1" />
<!-- categorias (e.g. drama, comedia, etc) -->
<xsd:element name="categories" minOccurs="0" maxOccurs="1">
    <xsd:complexType>

```

```

        <xsd:sequence>
            <xsd:element name="category"
                type="xsd:string"
                minOccurs="1"
                maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
</xsd:sequence>

<xsd:attribute name="type" type="tns:ContentMediaType"
    use="required" />
<xsd:attribute name="purpose" type="tns:ContentPurposeType"
    use="required" />
</xsd:complexType>

<xsd:complexType name="PersonType">
    <xsd:attribute name="role" type="tns:PeopleRoleType"
        use="required" />
    <xsd:attribute name="name" type="xsd:string"
        use="required" />
</xsd:complexType>

<xsd:simpleType name='TagType'>
    <xsd:restriction base="xsd:string" />
</xsd:simpleType>

<xsd:simpleType name='PeopleRoleType'>
    <xsd:restriction base='xsd:string'>
        <!-- produtor -->
        <xsd:enumeration value='producer' />
        <!-- realizador -->
        <xsd:enumeration value='director' />
        <!-- argumentista -->
        <xsd:enumeration value='argument' />
        <!-- actor / atriz -->
        <xsd:enumeration value='actor' />
        <!-- apresentador -->
        <xsd:enumeration value='hoster' />
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name='ContentMediaType'>
    <xsd:restriction base='xsd:string'>
        <xsd:enumeration value='video' />
        <xsd:enumeration value='audio' />
        <xsd:enumeration value='picture' />
    </xsd:restriction>
</xsd:simpleType>

```

```

    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name='ContentPurposeType'>
  <xsd:restriction base='xsd:string'>
    <xsd:enumeration value='ad' />
    <xsd:enumeration value='content' />
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

B.2.2 Esquema XSD de Recomendação

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://iex.ptin.pt/ctx/recommendation"
  xmlns:tns="http://iex.ptin.pt/ctx/recommendation"
  xmlns:content="http://iex.ptin.pt/ctx/content">

  <xsd:import namespace="http://iex.ptin.pt/ctx/content"
    schemaLocation="content.xsd"/>

  <xsd:complexType name="RecommendationType">
    <xsd:sequence>
      <xsd:element name="content"
        type="content:ContentType"
        maxOccurs="unbounded" minOccurs="1" />
    </xsd:sequence>

    <!-- date when the recommendation was calculated -->
    <xsd:attribute name="issueDate" type="xsd:date"
      use="optional" />
    <!-- recommendation's target user -->
    <xsd:attribute name="for" type="xsd:string"
      use="required" />
  </xsd:complexType>

  <xsd:element name="recommendations"
    type="tns:RecommendationType" />
</xsd:schema>

```