UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA ABORDAGEM DE SISTEMA DE RECOMENDAÇÃO ORIENTADA PELO APRENDIZADO SEM FIM

REGINALDO APARECIDO GOTARDO

ORIENTADOR: PROF. DR. ESTEVAM RAFAEL HRUSCHKA JUNIOR

São Carlos - SP
Fevereiro/2014

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA ABORDAGEM DE SISTEMA DE RECOMENDAÇÃO ORIENTADA PELO APRENDIZADO SEM FIM

REGINALDO APARECIDO GOTARDO

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação, área de concentração: Inteligência Artificial
Orientador: Dr. Estevam Rafael Hruschka Junior

São Carlos - SP
Fevereiro/2014

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária/UFSCar

G683as

Gotardo, Reginaldo Aparecido.
Uma abordagem de sistema de recomendação orientada pelo aprendizado sem fim / Reginaldo Aparecido Gotardo. -- São Carlos : UFSCar, 2014.
90 f.

Tese (Doutorado) -- Universidade Federal de São Carlos, 2014.

1. Sistemas de recomendação. 2. Inteligência artificial. 3. Aprendizado de computador. 4. Personalização. I. Título.

CDD: 003.7 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Uma Abordagem de Sistema de Recomendação Orientada pelo Aprendizado Sem Fim"

Reginaldo Aparecido Gotardo

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Membros da Banca:

Prof. Dr. Estevam Rafael Hruschka Junior
(Orientador - DC/UFSCar)

Prof. Dr. Sergio Donizetti Zorzo
(DC/UFSCar)

Profa. Dra. Vânia Paula de Almeida Neris
(DC/ UFSCar)

Prof. Dr. Nelson Francisco Favilla Ebecken
(UFRJ)

Prof. Dr. Emerson Cabrera Paraíso
(PUC-PR)

São Carlos
Fevereiro/2014

À Érica, um amor de muitas vidas e de toda a eternidade.

Aos meus anjos Nietzche, Eve, Bellinha e Eugênio..

AGRADECIMENTO

Agradeço ao meu orientador, Professor Dr. Estevam pela sua dedicação, pela sua paciência e sabedoria. Com toda a certeza, este trabalho tem grande potencial devido à sua ávida paixão pelo que faz e como demonstra isto.

Agradeço também à SEaD-UFSCar pelo fornecimento de dados importantes para experimentação e às Professoras Dra. Vânia Neris, Dra. Joice Otsuka e Dra. Sandra Abib pelo auxílio no acesso a estes dados.

Agradeço ao Professor Dr. Zorzo pelas palavras de apoio em momentos importantes e por me apresentar a área de sistemas de recomendação e me apresentar ao Dr. Estevam.

Agradeço também aos meus familiares, por compreenderem minha ausência e, assim, me fornecem apoio e solidariedade nos dias mais difíceis.

Agradeço aos colegas do MaLL, particularmente Maísa e Laurence pelo apoio. E também ao meu amigo Paulo, pela ajuda, pela motivação e atenção.

Agradeço a Deus por me dar, generosamente, um trabalho árduo, mas gratificante a ser feito e por colocar no meu caminho pessoas maravilhosas e gentis.

Imagine que sua mente é uma esfera. Seu conhecimento, aquilo que você sabe, está contido na esfera. A esfera, através de sua superficie, entra em contato com o mundo exterior, ou seja, aquilo que você não sabe. Quanto mais você aprende, maior se torna sua esfera e maior é a superficie da mesma. A área de contato com o exterior aumenta de maneira exponencial e então vem a sensação que o tamanho do desconhecido aumentou. Quanto mais sua esfera de conhecimento cresce, mais aumenta a sensação de que há um universo desconhecido ai fora.

Eu ouvi isto do Prof. Estevam na primeira aula da disciplina de Mineração de Dados em 2010. É a mais bela e brilhante explicação que já ouvi acerca do que disse o Filósofo Sócrates uma vez:

"- Só sei que nada sei! "

Obrigado Professor, nunca vou esquecer disto.

RESUMO

Os Sistemas de Recomendação possuem uma função muito bem definida: recomendar algo a alguém. Através de técnicas de Inteligência Artificial, mais particularmente de áreas como a Mineração de Dados e o Aprendizado de Máquina é possível construir Sistemas de Recomendação que analisem grandes volumes de dados e consigam predizer aos usuários algo que provavelmente irá lhes interessar. No entanto, algumas limitações dos Sistemas de Recomendações, causadas as vezes pelos próprios modelos de Mineração ou Aprendizado utilizados ou pela escassez dos dados disponíveis, os tornam computacionalmente caros ou imprecisos. Além disto, Sistemas de Recomendação em ambientes reais são dinâmicos, ou seja, os dados mudam com o passar do tempo seja com novas avaliações, novos usuários, novos itens ou mesmo atualizações de avaliações anteriores. A abordagem de Aprendizado Sem-Fim (SASF) visa um aprendizado autossupervisionado e autorreflexivo para, sobretudo, maximizar o aprendizado de um sistema com base em dados de fontes diversas, algoritmos que cooperem entre si para melhor modelar uma base de conhecimento e considerar a dinamicidade de problemas reais de aprendizado: Aprender amadurece com o tempo. Como já dito, sistemas de recomendação são dinâmicos e dependem de dados entre usuários e itens. Para minimizar esta dependência e prover resultados significativos e úteis aos usuários é apresentada neste trabalho uma abordagem de Sistema de Recomendação orientada pelos Princípios do Aprendizado Sem-Fim. Os resultados obtidos sugerem que é possível minimizar ou retardar a dependência de dados através de técnicas de acoplamento de classificadores e do controle do desvio de conceito. Com isto, é possível atuar com poucos dados de um sistema de recomendação que será dinâmico e receberá novas informações. Estas novas informações auxiliarão ainda mais no controle do desvio de conceito e na promoção de recomendações mais úteis. Por tudo isto, este trabalho apresenta como proposta o desenvolvimento de uma Abordagem para Sistemas de Recomendação baseada no Aprendizado Sem Fim, como forma de contribuir para o estado da arte em sistemas de recomendação e de implementar um sistema com resultados práticos através do Aprendizado sem Fim.

Palavras-chave: Sistema de Aprendizado Sem Fim, Sistemas de Recomendação, Personalização, Mineração de Dados, Aprendizado de Máquina.

ABSTRACT

Recommender Systems have a very well defined function: recommend something to someone. Through Artificial Intelligence techniques, more particularly from areas such as Data Mining and Machine Learning, it is possible to build recommendation systems. These systems will analyze large amounts of data and will inform users about some items that will probably interest them. However, some limitations of the recommender systems, which are sometimes, caused by the Mining or Learning models themselves or by the lack of available data make them computationally expensive or inaccurate. Besides, recommender systems in real environments are dynamic: data change over time or with new ratings, new users, new items or when user updates previous ratings. The Never Ending-Learning Approach (NEL) aims at a self-supervised and self-reflexive learning to mainly maximize learning of a system based on data from several sources, algorithms that can cooperate to make a better knowledge base considering the dynamic of real learning problems: learning improves along the time. As mentioned before, recommender systems are dynamic and depend on data between user and items. In order to minimize this dependency and to provide meaningful and useful results to users, this work presents a Recommender System approach guided by NEL Principles. Results show that it is possible to minimize or delay the data dependency through classifiers coupling techniques and concept deviation control. Due to that, it is possible to start with little data from a recommender system that will be dynamic and will receive new information. These new information will help even more in controlling the concept deviation and promoting the most useful recommendations. Then, this thesis presents how the Recommender System guided by NEL principles can contribute to the state of the art in recommender systems and implement a system with practical results through the Never-Ending Learning Approach.

***Keywords**: Never Ending Learning, Recommender Systems, Personalization, User Model, Data Mining, Machine Learning.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

RTW – *Read the Web*

RTWP – *Read the Web in Portuguese*

NEL – *Never Ending Learning*

NELL – *Never Ending Language Learning*

MU – *Modelo de Usuário*

AL – *Active Learning*

FBC – *Filtragem Baseada em Conteúdo*

FC – *Filtragem Colaborativa*

SASF – *Sistema de Aprendizado Sem-Fim*

SR-SASF - Sistema de Recomendação inspirado no design de Sistemas de Aprendizado Sem-Fim

RSA-NEL - *Recommendation System Architecture inspired by Never Ending Learning design*

SUMÁRIO

Capítulo 1

INTRODUÇÃO

"O propósito de aprender é crescimento. E nossa mente, diferente do corpo, cresce enquanto vivemos."
Mortimer Adler

1.1 Contexto

Sistemas de recomendação são tradicionalmente construídos com base em técnicas de aprendizado de máquina (e/ou mineração de dados) com o objetivo de identificar padrões em itens e usuários para que recomendações possam ser feitas. Existem inúmeros domínios de aplicação para sistemas de recomendação, entretanto, o domínio do comércio eletrônico é um dos que mais tem gerado investimentos nos últimos anos. Neste sentido, notam-se grandes investimentos em sistemas de CRM (*Customer Relationship Management*), em *Data Warehouses* e em *Data Mining (mineração de Dados)* que, juntos, visam permitir às empresas mudarem o foco de seus negócios para a peça chave: o cliente. O CRM, também denominado Marketing de Relacionamento, é uma nova estratégia para negócios usada como ferramenta para expansão e manutenção da carteira de clientes de uma empresa. O Princípio de Pareto[1] apresenta um ponto importante sobre a conquista da fidelidade dos clientes de uma empresa. No entanto, mais recentemente, a Teoria sobre a Cauda[2] Longa tem mostrado a importância do Marketing de Nicho. Assim, nota-se a grande preocupação

[1] Este princípio afirma que apenas 20% dos clientes geram 80% da receita de uma empresa, ilustrando a importância da conquista da fidelidade dos clientes.
[2] Trata-se de um termo usado para identificar distriuições de dados como a curva de Pareto onde o volume de dados é classificado de forma decrescente. O termo ganhou popularidade com a estratégia de muitas empresas de vender vários itens a nichos diversos ao invés de focar num nicho específico. http://www.wired.com/wired/archive/12.10/tail.html

com o atendimento a diversos grupos sociais e suas necessidades e isto não é possível apenas com o Marketing de Massa.

A personalização de sistemas busca este atendimento tornando o relacionamento da empresa com o cliente muito mais fácil, mais rápido e valorizado. O cliente sente que a empresa foi criada para atendê-lo. Esta abordagem dá aos clientes serviços agregados aos produtos vendidos pela empresa, na medida em que visa atender necessidades especiais do mesmo e isto torna a personalização na Web, principalmente em Comércio Eletrônico, um fator de competitividade.

A personalização por meio de Sistemas de Recomendação tem grande importância para os empreendimentos na Web, pois se trata de uma forma de marketing direto que, baseando-se em perfis de usuários, direciona conteúdo e propaganda. Na pesquisa realizada por Kobsa [1], os *Websites* que ofereceram serviços personalizados conseguiram um aumento de 47% no número de novos clientes. O marketing direto vem substituindo as tradicionais estratégias de marketing de massa, como a utilização de banners e as propagandas na televisão e no rádio, que sobrecarregam os usuários (ou clientes) com informações, muitas vezes, desinteressantes. As tecnologias existentes na Web proporcionam a exploração potencial do marketing direto através do direcionamento de conteúdo, de propaganda, adaptação de interfaces, entre outras formas.

O uso das técnicas tradicionais de aprendizado de máquina fornece informações importantes para o estabelecimento de grupos e perfis de usuários. No entanto, estas técnicas não reaproveitam o conhecimento já existente para melhorar o aprendizado. Mesmo o aprendizado incremental, que adiciona informações novas àquelas já existentes, também não se preocupa em observar, novamente, o que já estava definido. A proposta de uma arquitetura SASF (Sistema de Aprendizado Sem-Fim) [2] [3] para um Sistema de Recomendação tornou-se o desafio inicial deste trabalho a fim de auxiliar o tratamento de problemas tradicionalmente existentes em Sistemas de Recomendação.

1.2 Motivação e Objetivos

Um sistema de recomendação é normalmente desenvolvido para ajudar usuários a encontrar itens relevantes. Basicamente, todo sistema de recomendação faz o mesmo: tenta identificar os itens mais importantes para os usuários e então os apresenta a eles [4]. Sua tarefa é responder a uma função de relevância/utilidade para o usuário alvo. Mais formalmente, de acordo com [5], o problema de recomendação pode ser formulado como sendo *A* o conjunto de todos os usuários do sistema e *I* o conjunto de itens que podem ser recomendados (por exemplo livros, filmes, produtos eletrônicos, objetos de aprendizagem, pessoas, etc). Tanto o conjunto *A* quanto o conjunto *I* podem ser muito grandes (com milhares ou milhões de elementos). Considere *u* uma função de *utilidade* que mede quão útil o item *i* é para o usuário *a*

$u: A \times I \rightarrow R,$ onde *R* é um conjunto limitado e conhecido (por exemplo, valores de 0 a 5). O que o sistema precisa é encontrar e escolher o item $i' \in I$ com maior utilidade para o usuário *a*:

$$\forall a \in A, i'_a = \underset{i \in I}{\arg\max}\, u(\mathrm{a}, \mathrm{i})$$

Ou seja, o sistema precisa maximizar a utilidade do item recomendado através dos mais diferentes tipos de funções e algoritmos.

Uma série de abordagens tem sido propostas para resolver problemas em sistemas de recomendação (como os problemas encontrados na seção 2.7) como: filtragem baseada em conteúdo, técnicas usando classificadores, filtragem colaborativa, abordagens baseadas em ontologias, dentre outras. Pela revisão do estado da arte em sistemas de recomendação, as técnicas que têm obtido melhor desempenho envolvem predições em modelos de fatoração de matrizes treinados numa base de dados estática. No entanto estes sistemas não são muito flexíveis e ideais para problemas do mundo real, pois o processo de recomendação é complexo, e medir a qualidade de uma recomendação não é uma tarefa simples, mas sim dependente de inúmeros fatores [6].

Para criar um Sistema de Recomendação orientado pelo Aprendizado Sem-Fim (SASF) foram consideradas algumas premissas baseadas no atual SASF implementado no projeto *Read the Web* (RTW): o NELL (*Never Ending Language*

Learning) [3]. O objetivo do NELL é extrair informações da web em inglês (a princípio, mas existem módulos para outras linguagens sendo pesquisados e desenvolvidos) e, a partir destas informações melhorar sua ontologia que relaciona os conceitos aprendidos.

Uma premissa importante num SASF é que o aprendizado pode ser melhorado com o tempo e com novas informações. Há um "modelo de maturidade". Assim como seres humanos, com o passar do tempo, novas experiências e informações, podem ter o seu desempenho de aprendizado num mesmo tema melhorado. Para isto são necessárias características como a autossupervisão e a autorreflexão discutidas nos capítulos 3 e 4 deste trabalho.

O objetivo desta tese é utilizar um novo paradigma de Aprendizagem de Máquina para criação automática e atualização contínua de perfis que serão usados em Sistemas de Recomendação, contribuindo no auxílio à resolução de alguns problemas encontrados, como:

1. Problemas de Cauda Longa: itens recentes ou que possuam poucas avaliações feitas por usuários podem ser problemas potenciais na recomendação, pois algoritmos de Filtragem Colaborativa podem, por vezes, ignorá-los em seus cálculos [7].
2. O problema de partida fria (*Cold-Start Problem*): Num cenário real existem vários usuários que fizeram poucas avaliações ou vários itens que receberam poucas avaliações. Quando um usuário novo entra no sistema não há informação inicial sobre suas avaliações (pois ele ainda não as fez). O mesmo vale quando um novo item é cadastrado no sistema [5].
3. Incorporar novos usuários e novas avaliações: o ambiente de sistemas de recomendação é dinâmico e será preciso considerar a chegada de novos usuários e novos itens nas avaliações.
4. Aumento da Cobertura: O sistema precisa gerar recomendações para o maior número possível de usuários.
5. Superespecialização (*Over-Specialization*): ocorre quando a recomendação retorna itens muito similares àqueles vistos e avaliados pelo usuário [8].

[3] Detalhes do Projeto http://rtw.ml.cmu.edu

6. Problemas de usuários Ovelha-Negra (*Gray-sheep users problem*): quando usa-se a filtragem colaborativa para sistemas de recomendações alguns usuários podem ter baixíssima correlação com outros e isto traz um problema particular, uma vez que será difícil encontrar recomendações baseadas em interesses da comunidade (um típico recurso da filtragem colaborativa) [9].

Buscando-se a integração dos princípios de Sistemas de Recomendação e SASF, foram consideradas as seguintes premissas:

- Existem usuários e Itens no Sistema
- Novos usuários e novos itens podem ser inseridos
- O problema de recomendação relaciona usuários a itens
- Num cenário ideal todos os usuários teriam emitido opiniões sobre todos os itens, no entanto, isto nunca ocorre num ambiente real. Uma matriz que relacione usuários a itens tende a ser esparsa em boa parte da existência de um sistema de recomendação (usuários avaliam poucos itens).
- Ao predizer uma recomendação, é feita uma tentativa de prever a opinião de um usuário sobre um item, porém, o usuário pode, posteriormente inserir sua própria opinião sobre o item, assim, corroborando com o sistema na avaliação deste e na continuidade de suas tarefas.

Por tudo isto, um ponto importante a destacar é que a abordagem proposta proporciona uma solução para recomendação até que o usuário insira suas avaliações. A isto foi chamado de minimização do desvio semântico. O desvio semântico ocorre quando a recomendação diverge da opinião do usuário pela falta de dados deste usuário. Assim, enquanto o usuário não insere suas avaliações o sistema oferece uma solução provisória e com grau de confiança muito próximo a uma situação onde existam os dados reais.

1.3 Contribuições da Pesquisa

O presente trabalho contribui para a área de Sistemas de Recomendação através da exploração da arquitetura SASF e gerou os seguintes resultados:

- Proposta de uma arquitetura para sistemas de recomendação usando princípios do aprendizado sem fim;
- Proposta de acoplamento de classificadores para auxiliar na geração de dados de perfis para posterior recomendação;
- Proposta do uso de algoritmos de agrupamento em conjunto com classificadores para promover mais informações em visões diferentes para a recomendação;
- Proposta de um formato incremental para o modelo de perfis de usuários (*bootstrapping*), usando as técnicas acima citadas, visando auxiliar o processo de filtragem colaborativa;
- Os problemas 1, 2 e 6 (cauda longa, partida fria e ovelha negra) citados anteriormente, são característicos da Filtragem Colaborativa. Como usamos técnicas de classificação nesta tese esses problemas são amenizados em ambientes reais;
- Contribuir para a diminuição do problema de “Aumento de Cobertura”, pois os resultados mostram que é possível aprender com poucos dados e gerar recomendações;

1.4 Organização do Trabalho

O trabalho está organizado da seguinte forma:

- No capítulo 2 é apresentado o conceito de Sistema de Recomendação, sua organização e forma como faz uso de técnicas de mineração de dados e aprendizado de máquina.
- No capítulo 3 são apresentadas as formas de aprendizado de máquina, sendo comparadas e, sobretudo, é discutido o Aprendizado Sem Fim.
- No capítulo 4 é descrita a arquitetura proposta neste trabalho e sua relação com as técnicas vistas, algoritmos utilizados e desenvolvidos e o funcionamento da mesma.
- No capítulo 5 são apresentados os experimentos e estudos de caso feitos para testes e validação da proposta.
- No capítulo 6 são apresentadas as considerações finais e o trabalhos futuros relacionados a esta tese.

Capítulo 2

SISTEMAS DE RECOMENDAÇÃO

"A grande finalidade da vida não é conhecimento, mas ação."
Thomas Huxley

2.1 Considerações Iniciais

Personalizar pode ser definida como a modificação da experiência do usuário de acordo com as suas preferências [10]. Portanto, um sistema que se propõe a esse objetivo tem que ser capaz de reconhecer padrões de comportamento e interesses por meio das operações que um usuário realiza. Atualmente, soluções de personalização têm sua aplicação em diversas áreas como em sistemas de informação, sistemas comerciais, help desks, sistemas de recuperação de informação e sistemas educativos [11].

As recomendações recebidas por outras pessoas são formas de abreviação de escolhas. Uma carta de recomendação é uma forma de apresentar um candidato a algum cargo. Para minimizar as dúvidas e necessidades que temos frente à escolha entre alternativas, geralmente confiamos nas recomendações que são passadas por outras pessoas, as quais podem chegar de forma direta (*word of mouth*) [4], cartas de recomendação, opiniões de revisores de filmes e livros, impressos de jornais, entre outros. Os sistemas de recomendação auxiliam no aumento da capacidade e eficácia deste processo de indicação já bastante conhecida na relação social entre seres humanos [10].

2.2 Os Sistemas de Recomendação

O sistema *Tapestry* é reconhecido como o primeiro sistema computacional de recomendação e trouxe com sua criação a expressão “filtragem colaborativa”, designando uma aplicação específica dos sistemas de recomendação. Nesse tipo de sistema o fornecimento de informações pelos usuários gera, mesmo que indiretamente, a criação de grupos de interesses [11]. São muitos sistemas, sobretudo Websites que utilizam a Filtragem Colaborativa. O princípio do algoritmo da filtragem colaborativa considera que o usuário ativo possui maior probabilidade de se interessar por itens que usuários semelhantes preferem ou preferiram. Para isto, calcula-se um grau de similaridade entre o usuário ativo (alvo) e os outros usuários. Os itens com maior grau de similaridade são recomendados ao usuário alvo. [12] [10].

Como visto na introdução deste trabalho, um sistema de recomendação pode ser formalmente definido como a maximização de uma função de utilidade:

- $u: A \times I \rightarrow R$ que relacione itens (do conjunto I) a usuários (do conjunto A).

Em geral, nos sistemas de recomendação, esta função de utilidade é um *rating* que indica como um usuário em particular gostou de um item. Por exemplo, João gostou do filme Superman, pois deu nota 5 (variando de 0 a 5). Este *rating* pode ser um valor explícito dado pelo usuário ou calculado pela aplicação com base em métricas (por exemplo, pode ser o resultado de uma avaliação, pode representar aprovado ou reprovado numa disciplina, etc). Os *ratings* referem-se às avaliações anteriormente feitas pelos usuários e, por isto, é necessário calcular a estimativa de um rating para um item que o usuário não viu para que as recomendações possam ser feitas. Veja o exemplo na tabela 2.1 onde “-“ representa itens que o usuário não avaliou.

As formas de se estimar (ou predizer) os valores de rating são, basicamente [5]:

- Recomendações Baseadas em Conteúdo,
- Recomendações Baseadas em Colaboração,
- Abordagem Híbrida.

Tabela 2.1 Exemplo de Avaliações Feitas por Usuários a Itens

Usuários	Itens							
	Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8
A1	5	-	-	4	5	1	-	-
A2	5	-	4	4	-	1	-	1
A3	-	4	-	1	1	4	-	4
A4	5	-	-	3	4	1	4	-
A5	-	2	1	-	4	1	-	-
A6	3	5	2	-	-	-	4	4

Com os avanços na área os métodos de recomendação se dividiram e incorporaram novos conceitos distribuindo-se da seguinte forma [9]:

1. Baseado em Conteúdo: visa recomendar itens similares (em relação aos seus conteúdos) a outros que o usuário gostou no passado.
2. Filtragem Colaborativa: visa recomendar itens de acordo com preferências de usuários similares ou de itens similares (em relação à interação entre usuários e itens).
3. Demográfico: baseado em informações demográficas do perfil do usuário (características que descrevam o usuário).
4. Baseado em Conhecimento: tipicamente baseados em casos anteriores (baseia-se em indicações de sucesso ou de especialistas, exige uma base de conhecimento que estabeleça relações entre itens).
5. Baseado em Comunidade: visa utilizar informações sobre relações entre os usuários
6. Híbridos: reúne mais de uma das técnicas apresentadas

O foco dos sistemas de recomendação tem sido comumente os *Websites* de *e-commerc*e (comércio eletrônico) [10]. A justificativa do emprego de tais sistemas é o aumento da lucratividade, como já dito, através da melhor adequação de consumo dos clientes. Esta adequação é realizada ofertando-se ao cliente os produtos mais recomendados ao seu perfil.

Através da taxonomia de sistemas de recomendação representada no trabalho de [13], serão apresentados os principais componentes observados nos sistemas de recomendação. A taxonomia está descrita abaixo, na figura 2.1. De acordo com essa taxonomia, os sistemas de recomendação ficam divididos em 4 componentes:

- A entrada de dados do usuário alvo e da comunidade;
- O método de recomendação (descoberta ou associação de informações) utilizado;
- A estratégia para apresentação das recomendações;
- Como será realizada a apresentação de recomendações e em que grau;

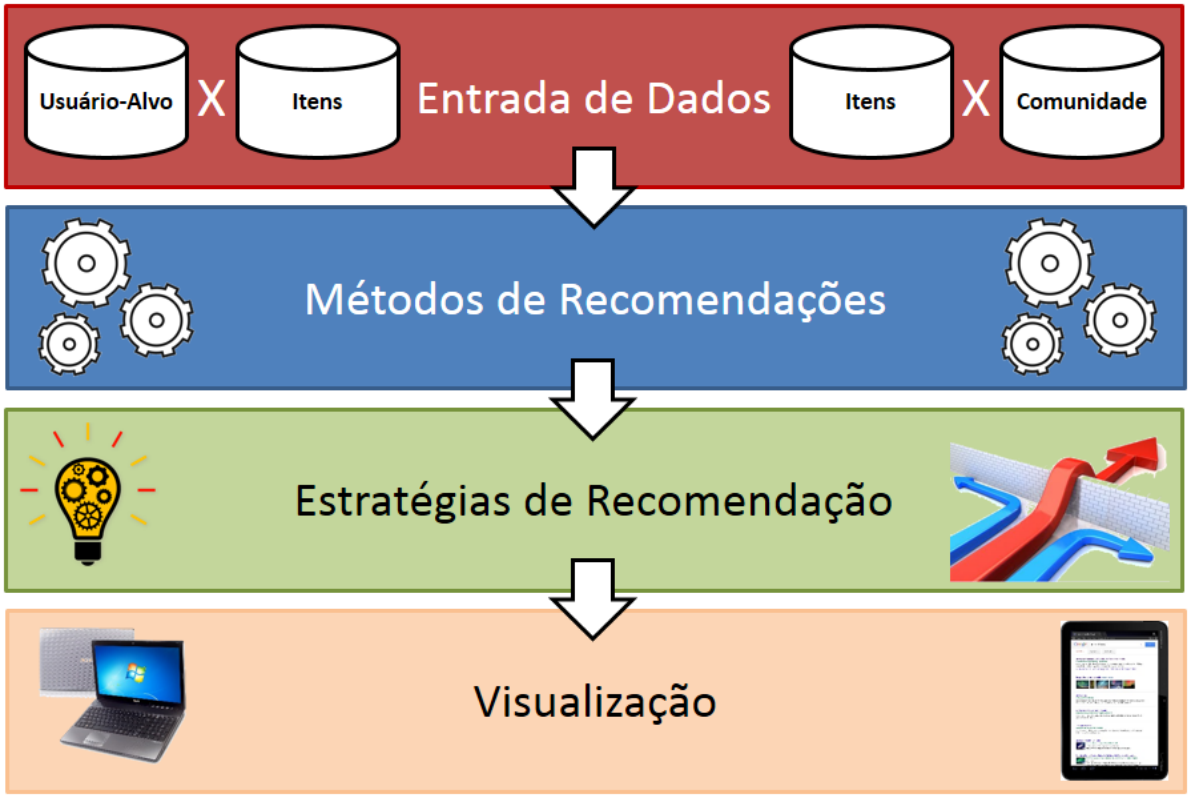


Figura 2.1 Taxonomia sobre Sistemas de Recomendação

Como pode ser observado na figura 2.1, as entradas do usuário alvo e da comunidade são armazenadas e tratadas segundo um método de recomendação. Após, são definidas as formas de recomendação. Necessita-se também a definição do grau de personalização adotado pelo sistema e de como o resultado será apresentado ao usuário. Nas seções seguintes serão descritas cada etapa desta taxonomia.

2.3 Entrada de Dados

Os dados de entrada do usuário alvo e os dados da comunidade (os demais usuários do sistema de recomendação) são a base para os sistemas de recomendação. As entradas do usuário alvo podem ser informações coletadas na sua navegação, de forma implícita (por logs ou mecanismos próprios de coleta), de sua opinião e interesses (coletados explicitamente por formulários ou campos para inserção de valores), através de seu histórico de compras e através de avaliações (que além da opinião do usuário incluem algum tipo de classificação).

As entradas da comunidade refletem o conjunto de dados dos demais usuários do sistema, agrupados por alguma critério (como contexto de navegação, gênero, dados demográficos, etc). Em sistemas de recomendação é usual dividir o grupo de usuários em usuário-alvo, já comentado acima, e vizinhança, que reflete os dados de outros usuários que serão usados para fins de cálculos de similaridade.

Adiciona-se a este tipo de entrada os comentários textuais, encorajados em alguns sites, para que os clientes deem explicitamente sua opinião.

2.4 Métodos de Recomendação

Dada a imensa quantidade de informações disponíveis na Web, usuários precisam de auxílio em mecanismos de busca e consumo de informações, pois nossos recursos, como seres humanos, são limitados para interpretação de todo esse conteúdo. Mesmo quando há restrição da análise de dados a um contexto ou a um Website específico, ainda haverá um grande volume de informações a ser trabalhada, entendida, consumida. Em Sistemas de Recomendação informações referem-se a itens que podem ser músicas, vídeos, artigos, pessoas, produtos, dentre outros. Cada usuário, com características e preferências de consumo distintas optará sempre por uma parte dessas informações. Mas, encontrar o que realmente se deseja pode ser uma tarefa dispendiosa. A necessidade por tecnologias para filtragem de informação é algo que remonta a década de 90. Já existia a preocupação com a quantidade de informação que estava sendo gerada pelos sistemas de informação e demasiada carga de informações que chegava até os usuários.

Enquanto a expressão “filtragem de informação” refere-se ao nome dado a processos para entregar informação para as pessoas corretas, ou seja, aquelas pessoas que realmente necessitam destas informações [14], há outra expressão, “recuperação de informação”, que, frequentemente, é confundida com a filtragem de informação.

No entanto, mesmo sendo duas soluções com vistas a auxiliar na solução de problemas de sobrecarga de informações, a recuperação de informação requer armazenamento, geração e manutenção de índices e recuperação de documentos (textuais ou com conteúdo binários). Com essas etapas funcionando devidamente o usuário poderá consultar explicitamente as informações que deseja, realizando a interação com o sistema.

A filtragem de informação é focada no perfil de usuário, constituído de preferências de navegação e seu modelo de comportamento mapeado no sistema. O sistema usa esse perfil para fornecer informações aos usuários. O processo de oferecer informações ao usuário é comumente é automático, visando tornar mais dinâmica a interação do usuário com o sistema [15].

2.4.1 Medidas de Similaridade

Informalmente, a semelhança entre dois objetos é uma medida numérica do grau no qual estes se parecem. Assim, os valores são maiores para objetos que mais se parecem. Formalmente, é possível estabelecer uma série de métricas que descrevam essa semelhança. Serão vistas as principais métricas em Sistemas de Recomendação.

Uma medida de similaridade muito utilizada é a distância entre dois objetos ou itens, calculada de acordo com suas propriedades. Um exemplo de métrica de distância é a Distância Euclidiana:

$$d(x,y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2} \qquad 2.1$$

Onde *n* é o número de atributos (dimensões) e x_k e y_k são os k-ésimos atributos dos objetos comparados, respectivamente. A medida de Distância Euclidiana da

fórmula 2.1 é generalizada pela métrica de distância de Minkowski, mostrada na fórmula 2.2.

$$d(x,y) = (\sum_{k=1}^{n} |x_k - y_k|^r)^{\frac{1}{r}} \quad 2.2$$

Neste caso *r* é um parâmetro que identifica diferentes métricas:

- *r = 1;* especifica a distância de quadra de cidade (chamada também de Manhattan, táxi, norma L1)
- *r = 2;* especifica a Distância Euclidiana
- *r = ∞;* especifica a Distância Suprema (L_{max} ou norma ∞) que é a diferença máxima entre qualquer atributo dos objetos.

Outra abordagem comum em Sistemas de Recomendação é considerar itens como vetores de documentos num espaço n-dimensional e calcular a similaridade como o cosseno do ângulo de separação entre eles. Isto pode ser visto na fórmula 2.3.

$$cos(x,y) = \frac{(x \bullet y)}{||x||||y||} \quad 2.3$$

Na fórmula 2.3 o ● representa o produto escalar entre os vetores e ||*x*|| representa a norma do vetor x. Esta métrica é conhecida como Similaridade do Cosseno.

Uma das abordagens mais usadas na Filtragem Colaborativa em Sistemas de Recomendação é a similaridade calculada através da correlação entre itens que representa a relação linear entre seus atributos.

$$Pearson(x,y) = \frac{\Sigma(x,y)}{\sigma_x \times \sigma_y} \quad 2.4$$

A Correlação de Pearson, mostrada na fórmula 2.4, resulta no cálculo da soma da covariância entre dois itens *x* e *y* divididos pela multiplicação do desvio-padrão destes. A *Correlação de Pearson* é uma das métricas mais comuns, embora existam diversas outras métricas que possam ser usadas. Mais detalhes sobre métricas e utilização podem ser encontrados no trabalho de (ADOMAVICIUS e TUZHILIN, 2005).

2.4.2 Filtragem Colaborativa (FC)

A Filtragem Colaborativa visa a identificação de correlações existentes entre os itens comprados ou visitados por outros usuários e pelo usuário em questão (alvo). Para encontrar essas correlações são usadas técnicas de aprendizado de máquina ou estatísticas visando comparar histórico de compras, frequência de utilização, ou outras informações relevantes da base de dados.

A ideia geral da FC é detectar padrões de perfis com base em inteligência coletiva, ou seja, usando dados de comportamento de usuários e grupos de usuários similares será calculada a previsão de um usuário alvo gostar (ou não) de determinado item. A filtragem colaborativa pode ser usada para correlação item-a-item também. Segundo Herlocker [18] a FC pode ser descrita em três fases:

- Calcula-se a métrica de similaridade dos usuários com o usuário alvo;
- Os usuários com maior similaridade são selecionados;
- As características dos usuários selecionados predizem de forma ponderada o comportamento do usuário alvo.

Seja *p* a predição de interesse (ou avaliação, ou preferência) de um usuário-alvo *a,* em relação a um item *i* da base de dados (como um filme, por exemplo). O cálculo de *p(a,i)* é uma estimativa calculada com base nas preferências de usuários vizinhos.. Cada usuário vizinho *u,* tem seu interesse em itens *i* que não foram vistos pelo usuário-alvo *a.* Assim, *p(u, i)* é o interesse (uma pontuação numa escola de valores possíveis) do usuário vizinho *u* sobre o item *i.* Nesse caso, o usuário *u* pertence a uma vizinha $\hat{U}$, um conjunto de usuários (a comunidade do usuário-alvo *a*). Todos os valores *p(u, i)* são agregados de forma ponderada para obtenção da predição *p(a, i).*

$$p_{a,i} = \operatorname{aggr}_{u \in \hat{U}} p_{u,i} \qquad 2.5$$

O símbolo $\hat{U}$ representa o conjunto dos *N* usuários “u” mais similares ao usuário “a” que classificaram o item “i”. Este modelo de FC utiliza contribuições dos usuários para a classificação realizada pelo sistema. A entrada do sistema é uma matriz cujas linhas representam os usuários e as colunas são itens. Os dados na matriz são as

avaliações (*rating*) que cada usuário forneceu aos itens. Esta técnica também é chamada de técnica do vizinho mais próximo ou "*k-nearest-neighbor*" e a definição de similaridade pode utilizar coeficientes, como a correlação de Pearson [4] [18] [13].

Na equação (2.6) entende-se *Wa,u* como a correlação do usuário alvo (*a*) com um usuário vizinho (*u*). Para cálculo efetivo e viável há a necessidade de mais de uma avaliação em comum. Caso a correlação de Pearson seja usada como métrica (que é o caso da expressão abaixo) os resultados variam entre 1 para similaridade total, e -1 para total dissimilaridade (são inversamente similares) [10].

$$W_{a,u} = \frac{\sum_{i=1}^{m} [(r_{a,i} - \bar{r}_a) * (r_{u,i} - \bar{r}_u)]}{\sqrt{\sum_{i=1}^{m} (r_{a,i} - \bar{r}_a)^2 * \sum_{i=1}^{m} (r_{u,i} - \bar{r}_u)^2}} \qquad 2.6$$

a – usuário alvo

u – usuário "vizinho"

$W_{a,u}$ – correlação do usuário "a" com o usuário "u"

$r_{a,i}$ – avaliação do usuário "a" para o item "i"

$\bar{r}_a$ - média de todas as avaliações comuns entre o usuário "a" e o usuário "u"

$r_{u,i}$ – avaliação do usuário "u" para o item "i"

$\bar{r}_u$ - média de todas as avaliações comuns entre o usuário "u" e o usuário "a"

O cálculo da predição pode ser efetuado através da equação (2.7).

A predição $P_{a,i}$ da classificação do item "i" para o usuário alvo "a" é a média ponderada das avaliações que os N vizinhos (u) do usuário alvo (a) deram ao item "i". O valor N pode ser determinado pelo próprio sistema, de acordo com critérios particulares.

No exemplo a seguir extraído de [10], de acordo com a tabela 2.2 o usuário U2 avaliou 10 itens (item1 até item 10), segundo uma escala de classificação de 1, para menor interesse, até 5, para maior interesse. O item7 não foi avaliado pelo usuário U2.

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n}[(r_{u,i} - \bar{r}_u)^* w_{a,u}}{\sum_{u=1}^{n}(w_{a,u})} \qquad 2.7$$

De acordo com a matriz apresentada, o usuário U2 não gostou do item6, avaliando-o com o valor 1. No entanto, este usuário gosto do item1, avaliando com o valor máximo (5). Verifica-se também, que o usuário U2, nestes dois itens, concorda com a avaliação dada pelo usuário U1.

Tabela 2.2 Avaliação Usuários X Itens [10]

Usuários	Itens									
	Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8	Item9	Item10
U1	5	-	-	4	5	1	-	-	4	3
U2	5	1	4	4	4	1	-	1	5	4
U3	-	4	1	1	1	4	3	4	-	-
U4	5	1	3	3	4	1	4	-	-	-
U5	2	2	1	5	4	1	-	-	-	-
U6	3	5	2				4	4	4	4
U7	2	2	2	3	3	3	-	-	-	-
U8	4	1	4	3	-	-	5	-	4	2

Para medir a similaridade entre estes dois usuários utiliza-se o coeficiente de Pearson. Os itens em comum estão destacados na tabela 2.3.

Tabela 2.3 Avaliações em comum dos usuários U1 e U2 [10]

Usuários	Itens									
	Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8	Item9	Item10
U1	**5**	-	-	**4**	**5**	**1**	-	-	**4**	**3**
U2	**5**	1	4	**4**	**4**	**1**	-	1	**5**	**4**
U3	-	4	1	1	1	4	3	4	-	-
U4	5	1	3	3	4	1	4	-	-	-
U5	2	2	1	5	4	1	-	-	-	-
U6	3	5	2				4	4	4	4
U7	2	2	2	3	3	3	-	-	-	-
U8	4	1	4	3	-	-	5	-	4	2

As avaliações dos itens em comum entre os usuários ficaram da seguinte maneira:

U1: [5; 4; 5; 1; 4; 3]

U2: [5; 4; 4; 1; 5; 4]

Nota-se a concordância entre estas avaliações. A média das avaliações foi de 3,67 para U1 e 3,83 para U2. Aplicando a equação (2) obtém-se o valor 0,87. Desta forma, U1 e U2 são bastante similares, concordando em suas avaliações. Aplicando agora o coeficiente de Pearson (fórmula 2.4) para o usuário U2 em relação a todos os outros usuários têm-se os resultados na tabela 2.4.

Tabela 2.4 Coeficiente de Person dos usuários em relação ao usuário U2 [10]

Usuário	Pearson (U2)
U1	0,87
U3	-1
U4	0,95
U5	0,39
U6	-0,55
U7	-0,11
U8	0,86

Assim, após os cálculos e obtenção da tabela de relação de usuários (2.4) observa-se que o usuário U2 é bastante similar a U4, U1 e U8, relativamente similar a U5 e não é similar a U3, U6 e U7 em suas preferências. Logo, a partir da descoberta dos usuários com maior similaridade o próximo passo é efetuado, que éa predição para os novos itens deste usuário.

Caso seja aplicada a equação a equação 2.7 para predizer a nota que o usuário U2 atribuiria ao item7, considerando vizinhos com correlação maior que 0,9 (limite assumido) e os itens desses vizinhos em comum com o usuário alvo, tem-se o resultado de 4,33. Isso significa que caso o usuário U2 consumisse o item7, daria como uma nota igual a 4,33 (predição) para tal item, considerando a avaliação dos vizinhos mais próximos.

2.4.3 Filtragem Baseada em Conteúdo (FBC)

A filtragem baseada em conteúdo consiste em recomendar itens através das descrições (conteúdo) obtidas do próprio item. Este conteúdo pode ser extraído de forma automática ou pode ser anotado por um especialista de domínio. A FBC visa estimar a utilidade de um item *i* para um usuário *a* através de uma função *f(i, a) -> u*, baseando-se em utilidades de itens anteriormente aprovados pelo usuário, ou seja, calculando-se $f(i' \mid i' \in I, a)$ a utilidade de outros itens pelos quais o usuário se interessou.

Esta abordagem tem forte base nas pesquisas em Sistemas de Recuperação de Informação (RI, do inglês *Information Retrieval – IR*). O foco da RI é responder consultas do usuário, por exemplo, buscando todos os filmes relacionados ao termo *Superman*. Enquanto isto, a FBC procura padrões de compra ou consumo semelhantes para recomendar produtos ou informações aos usuários. Vale-se de mecanismos para Descoberta de Conhecimento em Bases de Dados, como regras de associação e Mineração de Dados. Pode ser realizada em conjunto de informações, baseando-se em palavras-chave para relacionar documentos textuais, ou qualquer outro tipo de dado. Os principais passos da FBC são [6]:

- Coleta de Informações dos Itens pelo Sistema, por exemplo, num sistema onde o item é um livro seriam coletadas informações descritivas como autor, ano, *tags,* editora, etc;
- Captura de Avaliações (*ratings*) do usuário dadas a itens, por exemplo, o usuário precisaria classificar alguns destes livros como *gostei/não-gostei* ou usando uma escala de 1 a 5;
- Criação de Perfis de Usuário gerando, por exemplo, listas de interesses com base em itens bem classificados no passo anterior. O sistema pode refazer o processo se o usuário classificar mais itens. Por exemplo, o sistema irá relacionar livros com conteúdos similares e criar listas a partir da avaliação daqueles que o usuário teve interesse (avaliados como "*gostei"* por exemplo);
- O Sistema deverá gerar um *ranking*, uma ordenação, daquilo que o usuário não viu, em função daquilo que ele já avaliou, comparando os conteúdos dos itens entre si (se o item possui conteúdo similar a algo que o usuário já gostou é provável que ele goste, caso seja similar a algo que ele não gostou, é provável que ele não goste!).

2.4.4 Demográfico

Sistemas de Recomendação baseados em Demografia valem-se de dados pessoais do usuário exemplos: idade, localização, gênero, profissão, dentre outros. As recomendações são feitas pelas similaridades de perfis usando classificadores para isto. Em muitos casos o usuário pode não fornecer tais dados e isto é um problema comum na abordagem demográfica.

2.4.5 Baseado em Conhecimento

Sistemas de Recomendação deste tipo utilizam alguma estrutura de conhecimento previamente definida para realizar inferências sobre preferências do usuário. O usuário é representado por um perfil e a inferência será de acordo com interesses deste perfil. A ideia geral é o uso de estruturas de conhecimento e, em geral, estas estruturas são ontologias. [16] [17]

2.4.6 Abordagem Híbrida

A abordagem híbrida implica na utilização de mais de uma das técnicas apresentadas. O intuito é reduzir os problemas gerados em cada uma das demais técnicas como será discutido na seção 2.7.

2.5 Estratégias de Recomendação

Tratam-se da forma de apresentação das recomendações aos usuários, estipuladas de acordo com o contexto do sistema.

2.5.1 Listas de Recomendação

Listas de recomendação usam algoritmos mais simples para agrupamentos de itens mais populares, não havendo necessidade de análise mais profunda para a criação destas. Nesse caso há aproveitamento de informações de contexto e uma abordagem Top-N frequentemente usada. Os Top-N itens são os melhores classificados de alguma forma (vendas, acessos, votos).

A figura 2.2 apresenta um exemplo de lista de recomendações da Livraria Cultura, que resulta nos livros mais vendidos na categoria geral no mês visitado (abril de 2014).

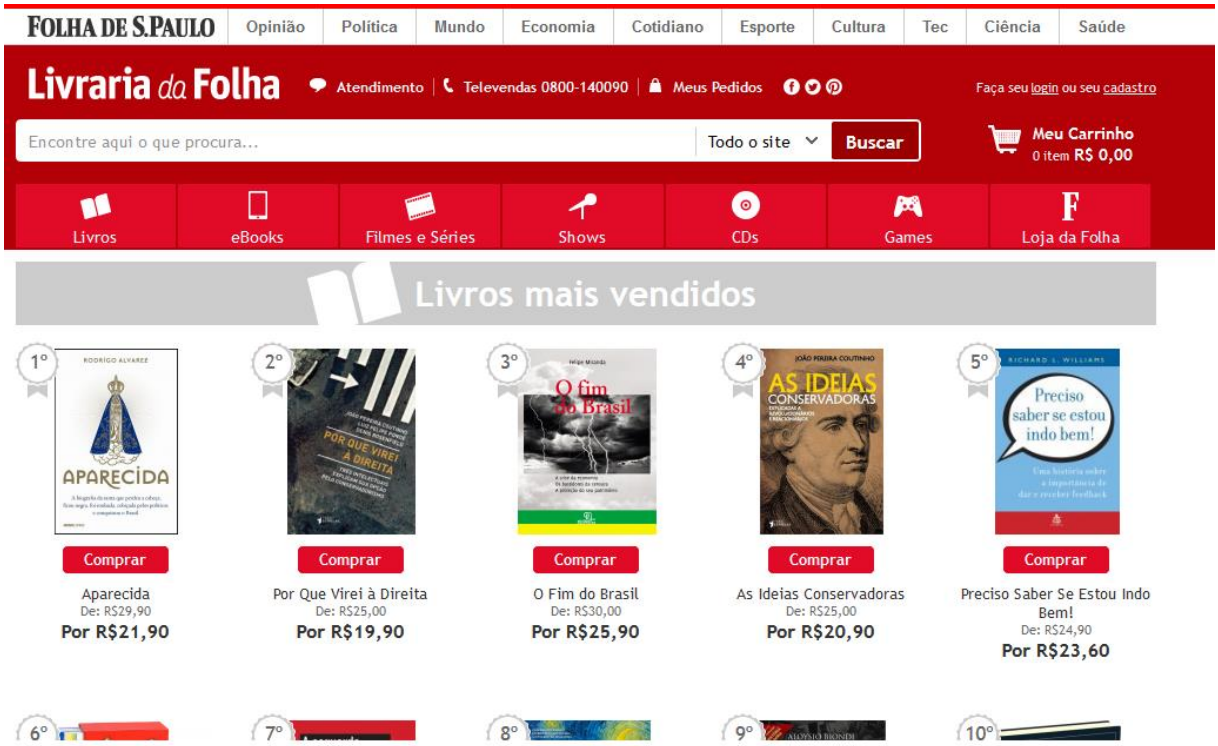


Figura 2.2 Exemplo de Listas de Recomendação

2.5.2 Avaliações de usuários

As avaliações dos usuários consistem na classificação do usuário dada a um determinado item. Num site de e-commerce, esta classificação pode ser após a compra de um produto. Num site educacional, pode ser a avaliação de um conteúdo apresentado. Este tipo de estratégia fornece aos futuros usuários a opinião dos anteriores que tiveram algum tipo de relacionamento com o item em questão. Isto assegura a novos consumidores (ou alunos) a melhoria no processo de escolha, visto que pode ser comparado com escolhas anteriores. A avaliação dada pelos usuários precisa ser verídica para garantir a “honestidade” da recomendação.

A figura 2.3 apresenta um exemplo de classificação de acordo com as avaliações dos usuários que compraram o livro Harry Potter no site da Amazon.com.

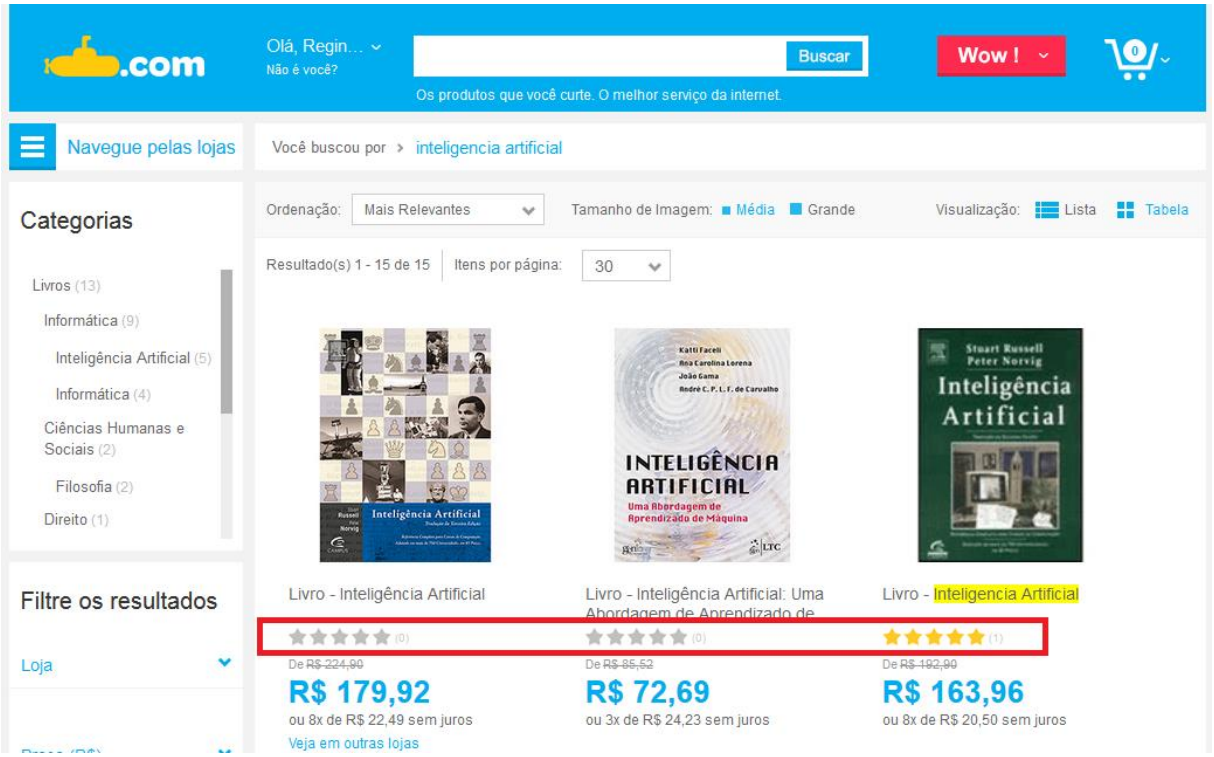


Figura 2.3 Exemplo de Avaliação dos Usuários

2.5.3 Associação por conteúdo

Neste tipo de recomendação considera-se o conteúdo dos itens, como por exemplo, autoria do conteúdo, contexto ou, no caso de um sistema de vendas de livros, quais os itens que possuem semelhança com o comprado, visitado ou apresentado. Diversas técnicas podem ser usadas para esta recomendação. Podem ser inseridas, manualmente, informações como tags que, posteriormente permitirão a identificação da associação [10]. A associação por conteúdo também pode ser realizada manualmente, mas trata-se de um serviço oneroso. A figura 2.4 apresenta um exemplo deste tipo de estratégia na forma: “Produtos do Mesmo Autor”, utilizada pelo site da Livraria Cultura.

Figura 2.4 Exemplo de Associação por Conteúdo

Já a figura 2.5 apresenta um exemplo de associação por conteúdo na forma de “Compre Junto”, realizada pelo site Livraria Folha.

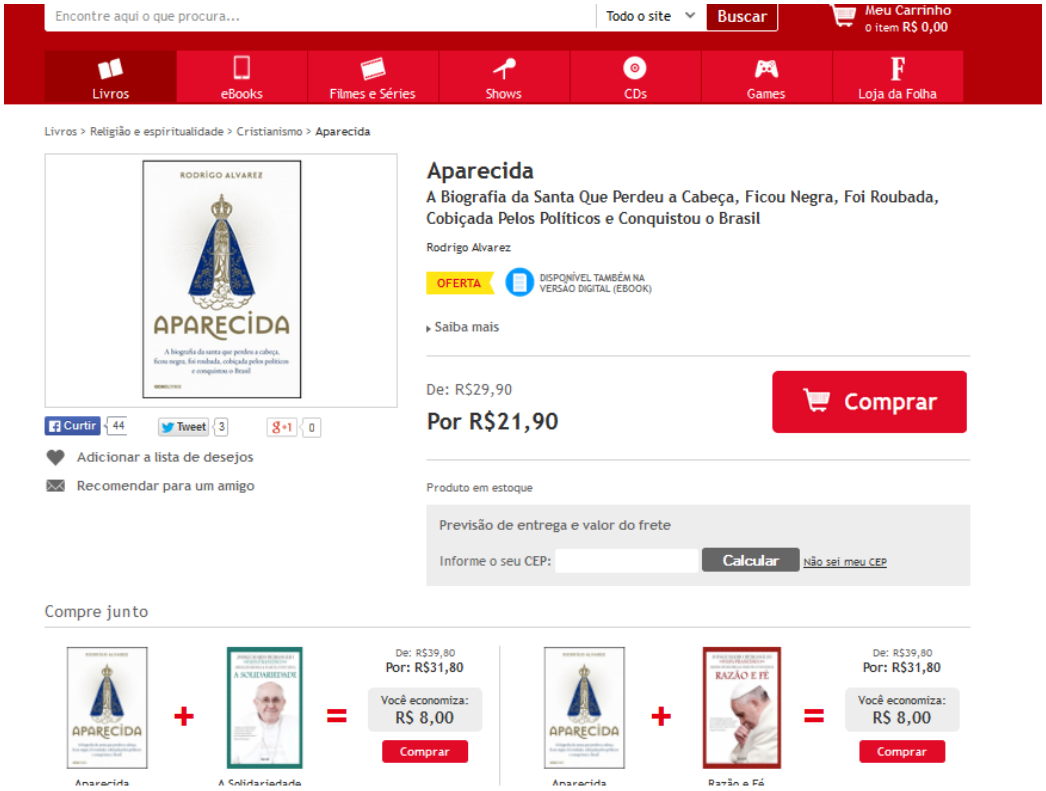


Figura 2.5 Exemplo de Associação por Conteúdo

2.5.4 Recomendações Pessoais

Através da análise detalhada das ações dos usuários pode-se traçar um perfil e representá-lo num modelo de usuário (MU). Este perfil servirá para futuras recomendações ao usuário. Além da análise de ações comportamentais, o usuário pode informar explicitamente suas preferências ou podem ser guardadas informações sobre as consultas realizadas pelo usuário, caso o sistema disponha de mecanismo de consultas [10].

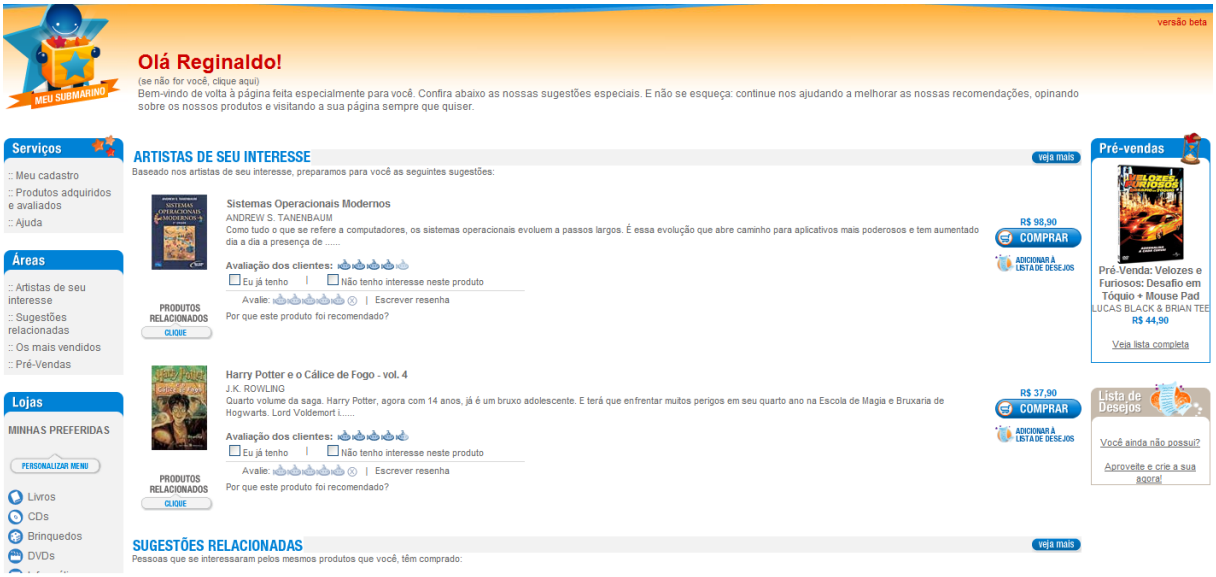


Figura 2.6 Exemplo de recomendações baseada no perfil do usuário

O usuário também pode marcar um determinado item como interessante e contribuir para a formação de seu perfil. Estas informações serão guardadas no MU. Na figura 2.6 é apresentado um caso de recomendações pessoais, direcionadas ao cliente de acordo com seu perfil de navegação.

Na figura 2.7 o usuário tem a opção de acompanhar um item específico e assim, explicitamente, declara-se interessado pelo Mp4 Player e, possivelmente, pela categoria de Eletrônicos, Áudio e Vídeo, a categoria que abrange o produto em questão.

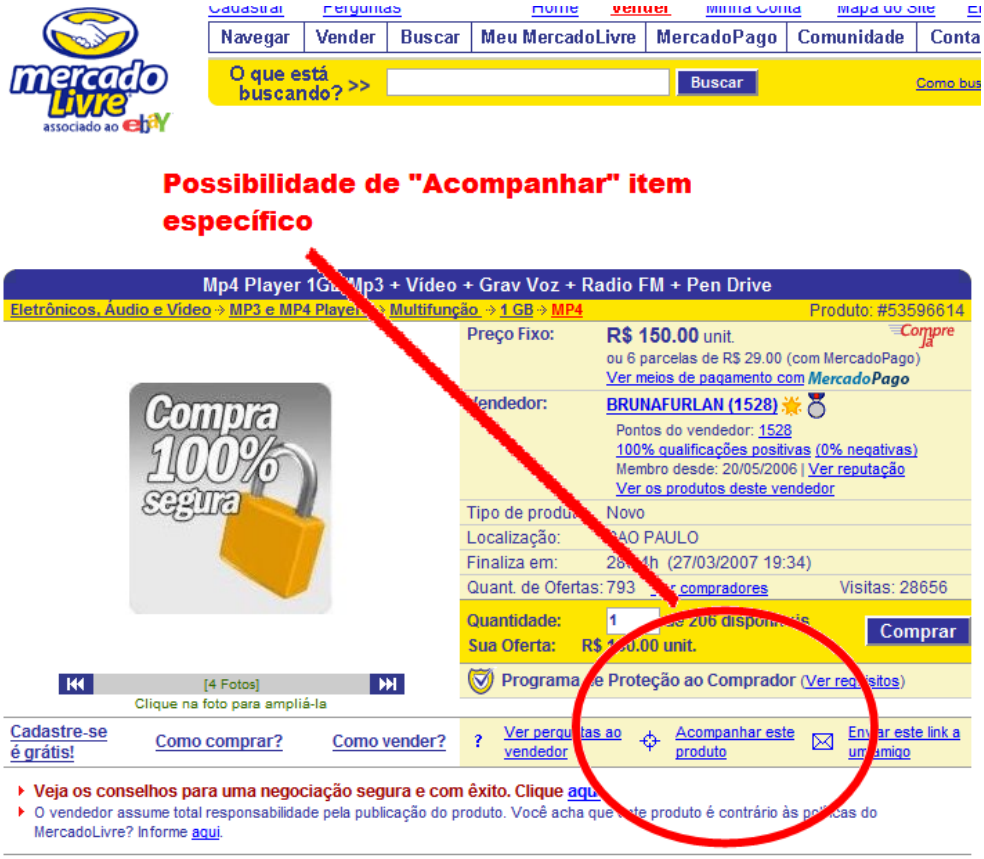


Figura 2.7 O usuário pode acompanhar um item específico

Neste tipo de estratégia devem ser utilizadas técnicas capazes de relacionar numa base de dados informações do usuário e armazená-las num perfil definido previamente, na criação do sistema.

2.5.5 Recomendações Off-line: E-mails

Para atrair a atenção dos usuários enquanto não está no sistema, a personalização vale-se também de técnicas off-line. As recomendações realizadas após a navegação do usuário e saída do sistema podem fazer uso das estratégias anteriores. Através dos e-mails de ofertas pode-se recomendar itens ou conteúdos aos usuários, estruturando-os conforme seu perfil de navegação ou de acordo com as estratégias do modelo de negócios do sistema [10].

Num sistema comercial podem ser oferecidos novos itens para compra. A figura 2.8 apresenta um exemplo de envio de ofertas pelo site MercadoLivre, baseadas na navegação do usuário.

Figura 2.8 Exemplo de Avaliação dos Usuários

2.6 Apresentação e Grau de Personalização

Os dados de entrada do usuário alvo da comunidade são a base para os sistemas de recomendação. No trabalho de [18] (VENSON, 2002) são apresentados tipos destes dados que podem ser coletados. As entradas do usuário alvo podem ser dados vindos de sua navegação implícita, de sua opinião e interesses (coleta explícita), através de seu histórico de compras e através de avaliações (que além da opinião do usuário incluem algum tipo de classificação). Além destas informações, a criação de categorias de produtos e itens pode incluir atributos-chave que, associados à coleta explícita ou implícita, fornecem dados relevantes. Por exemplo, saber que um usuário navega pela categoria de livros num *Website* de comércio eletrônico, através da inclusão de informações especiais (atributos-chaves) que identificarão esta preferência de usuário pela sua navegação.

As entradas da comunidade permitem a criação de um ambiente colaborativo. Incluem todos os itens de entrada do usuário alvo, porém aplicado a um grupo mais amplo de pessoas que possam ter interesses em comum. Adiciona-se a este tipo de entrada os comentário textuais, encorajados em alguns sites, para que os cliente deem explicitamente sua opinião [10].

A apresentação das recomendações de maneira adequada para os usuários (clientes, por exemplo) é decisão importante nos sistemas computacionais. Por isto, classificação define a apresentação da seguinte forma:

- Tecnologia Push: recomenda itens sem que o usuário esteja interagindo com o sistema;
- Tecnologia Pull: as recomendações só são apresentadas quando o usuário deseja e solicita explicitamente;
- Apresentação passiva: as recomendações estão inseridas no contexto da aplicação;

A eficiência do sistema de recomendação é determinada pela precisão deste e sua utilidade. A precisão diz respeito ao quão próxima a recomendação chega da avaliação real do usuário. Em sistemas onde há avaliação explícita (por exemplo uma avaliação de 1 a 5) o cálculo é estimado com relação ao valor previsto e ao valor avaliado. Já a utilidade diz respeito ao uso da recomendação pelo usuário (neste caso é preciso checar se o mesmo chega a utilizar a recomendação dada). Por isto, é preciso observar o comprometimento na oferta de recomendações que podem estar "erradas" ou fazerem parte de um processo de inferência não muito preciso.

Isto é um desafio para sistemas de recomendação e está associado a estratégias para que os sistemas não usem todos os recursos disponíveis na oferta de recomendações, atenuando o problema da imprecisão, mas com a contrapartida de que haverá perdas na oferta de serviços personalizados. Assim, os níveis de personalização para sistemas de recomendação podem ser agrupados da seguinte forma [20]:

- Não-personalizado: permitem a seleção manual através de categorias estabelecidas no projeto do sistema.
- Efêmero: Não usam dados persistentes para as recomendações, apenas dados da sessão de navegação atual do usuário.
- Persistente: oferecem as recomendações mais personalizadas e necessitam de grande quantidade de informações.

2.7 Fragilidades em Sistema de Recomendação

Nas subseções abaixo são descritas alguns pontos fracos mais comuns em relação aos sistemas de recomendação.

2.7.1 Novo Usuário (*Cold-Start*)

Quando um usuário é novo no sistema, as informações sobre ele geralmente são escassas. Desta maneira é possível analisar apenas informações adquiridas durante o processo de cadastro deste usuário. No caso da filtragem colaborativa, por exemplo, não podem ser recomendados itens já que o mesmo usuário não interagiu nem tampouco avaliou nenhum item do sistema, não havendo, assim, dados suficientes para calcular sua similaridade com a comunidade dos outros usuários.

Sistemas de Recomendação Demográficos não sofrem o problema do *Cold-Start* caso o usuário forneça dados sobre si. No entanto, o *Cold-Start* é um sério problema na Filtragem Colaborativa. Christakou e Stafylopatis [19] propuseram um sistema de recomendação híbrido que utilizasse resultados da Filtragem baseada em Conteúdo e Filtragem Colaborativa para recomendar filmes. O método deles consiste em usar informações dos filmes (sinopse, tipo, etc) que outras pessoas já avaliaram para buscar similaridade entre estas pessoas e o usuário alvo, gerando recomendações.

Já o método proposto por Kim e Li [22] usa informações dos itens para calcular a similaridade entre eles e gerar informações para a filtragem colaborativa (avaliações). Han e Karypis [20] apresentam dois algoritmos para Filtragem Colaborativa que recomendam itens baseando-se em características dos mesmos. No entanto, ambos apresentam a limitação de recomendação Top-N apresentada em [21] na qual não é possível correlacionar itens novos com itens existentes, caso os primeiros não possuam avaliações (a recomendação Top-N busca gerar as melhores recomendações com base em vizinhança).

Tiraweerakhajohn e Pinngern [22] propõem um algoritmo usando regras de associação para encontrar similaridade entre itens (como é feito na Filtragem baseada em Conteúdo) e depois recomendar usando técnicas da Filtragem Colaborativa. Liu [23] propõe um algoritmo para Filtragem Colaborativa baseado em modelo que calcula a similaridade entre itens usando características que os descrevem. Posteriormente, faz a predição de acordo com esta similaridade.

Como é possível observar, são diversos os trabalhos que lidam com o problema do *Cold-Start*. Nesta tese, a proposta apresentada orienta-se por princípios do Aprendizado Sem-Fim para auxiliar no tratamento deste problema.

2.7.2 Ovelha Negra (*Gray Sheep*)

Usuários com interesses raros levam desvantagem na filtragem colaborativa, pois a possibilidade de encontrar usuários semelhantes torna-se muito pequena. Novamente, aconselha-se o uso de filtragem baseada em conteúdo.

2.7.3 Primeira avaliação (*Early-rater*)

Quando um novo item é integrado ao sistema ele terá uma baixa probabilidade de ser recomendado já que ainda não há avaliações sobre o mesmo. Com o tempo e as avaliações dos usuários o item passará a ter maior probabilidade de ser recomendado.

2.7.4 Avaliações Esparsas

Isto ocorre quando o sistema contém muitos itens e/ou muitos usuários, mas poucos usuários realizando avaliações regularmente. Assim, é um problema mais comum à Filtragem Colaborativa. Deste modo, a matriz de relação usuário-item se torna esparsa e a busca de similaridade fica prejudicada. Para tratar este problema são comumente usadas técnicas de redução de dimensionalidade [24] [25].

2.7.5 Superespecialização

É um fato que acontece quando o sistema não é capaz de recomendar itens além daqueles determinados pelo perfil do usuário. O que o sistema ignora é a mudança de interesse do usuário com o passar do tempo. Este tipo de problema é mais comum na Filtragem Baseada em Conteúdo, pois a Filtragem Colaborativa possui uma característica Cross-Gênero, ou seja, por basear-se em opiniões da comunidade e em usuários similares (vizinhos) há um chance maior de recomendar itens com certo grau de surpresa, “pensando fora da caixa” [26].

2.7.6 Falta de surpresa (*Serendipity*)

Como consequência da superespecialização, um determinado item nunca poderá ser recomendado a um usuário que foi limitado aos interesses definidos no seu perfil. Por exemplo, José (usuário fictício) que gosta de filmes de japoneses dificilmente receberá como recomendação o filme “Clã das Adagas voadoras” pois o mesmo foi classificado como romance, apesar deste filme poder ter as duas classificações.

2.7.7 Impossibilidade de análise

Alguns tipos de conteúdo ainda não podem ser classificados e analisados devido a alguma limitação tecnológica ou ao alto custo computacional. Por exemplo, o conteúdo de um vídeo ou áudio só pode ser avaliado e descrito, mas sua análise automática ainda é um problema de pesquisa.

2.8 Considerações Finais

Este capítulo apresentou uma síntese sobre os Sistemas de Recomendação, seu funcionamento e características. Além disto, algumas das limitações mostradas e que comumente são tratadas com a abordagem híbrida ou técnicas de recomendação “genéricas” como a lista dos “Top-N” itens mais interessantes, e que foram tratadas neste trabalho usando a abordagem proposta.

O capítulo 3 a seguir apresentará conceitos e importância sobre o aprendizado de máquina e o aprendizado Sem-Fim.

Capítulo 3

APRENDIZADO DE MÁQUINA E O APRENDIZADO SEM-FIM

"My own view is that an expert system, a knowledge-based system is not, of itself, intelligent. If an expert system - brilliantly designed, engineered, and implemented - cannot learn not to repeat its mistakes, it is not as intelligent as a worm.[4]"

3.1 Considerações Iniciais

O Aprendizado de Máquinas pode ser considerado uma área da Inteligência Artificial. Seu objetivo é o desenvolvimento de algoritmos que possibilitem às máquinas computacionais aprenderem. No contexto de Inteligência Artificial, uma definição de aprendizado [27] é a seguinte:

> *"O processo, pelo qual um sistema inteligente é adaptado através do estímulo do ambiente no qual está inserido, chama-se aprendizado e seu tipo determina-se pela maneira como esta adaptação é realizada."*

Os métodos de aprendizagem são definidos pela forma como se relacionam com os estímulos do ambiente e os ajustes do sistema.

[4] Extraído de [86]

3.2 Aprendizado de Máquina

O aprendizado de máquina (AM ou ML termo em inglês *Machine Learning*) é uma área de pesquisa em Inteligência Artificial que visa automatizar um processo de aprendizagem. Um dos mecanismos fundamentais que rege os processos de aprendizagem de máquina automático pode ser considerado a busca por métodos e técnicas que permitam a sistemas computacionais melhorarem seu desempenho com base em informações novas ou atuais e de maneira autônoma.

Assim, um agente inteligente tentará compreender o comportamento de um sistema com apenas com algumas informações sobre o mesmo. A partir disto induzirá um modelo e tentará generalizar os conceitos aprendidos a partir de seu modelo. Uma das vantagens do Aprendizado de Máquina é a sua utilização em problemas complexos com soluções que nem sempre são exatas.

Existem três abordagens clássicas para o aprendizado de máquina: o aprendizado supervisionado, o aprendizado não-supervisionado [28] e o aprendizado semissupervisionado [29].

3.2.1 Aprendizado Supervisionado

O aprendizado supervisionado é o processo de aprendizado de máquina que fornece exemplos rotulados ao mecanismo de indução (o algoritmo ou técnica responsável pela geração da hipótese de saída dada uma entrada ou conjunto de entradas). Assim, inicialmente, o mecanismo de indução recebe exemplos rotulados, ou seja, exemplos descritos por um conceito de classe. O indutor gerará hipóteses para exemplos não rotulados, predizendo, assim, a classe deste exemplo.

Para ilustrar este tipo de aprendizado num sistema de recomendação comercial para livros e CDs considere que o sistema possua informações sobre clientes e que uma destas informações é uma categoria de cliente: assíduo (faz compras várias vezes num período), esporádico (faz poucas compras num período) ou novo cliente (nunca comprou). Estas classes de clientes são também chamadas de rótulos e são informações já existentes no sistema, previamente inseridas por um especialista. O sistema “sabe” identificar dentre os clientes atuais quais as classes destes, pois há uma relação implícita entre os dados de cada cliente e seu rótulo. A aplicação do aprendizado supervisionado possibilitaria induzir um modelo (explícito ou não) a partir

deste conhecimento pré-existente de maneira que um cliente ainda não classificado possa ser categorizado corretamente.

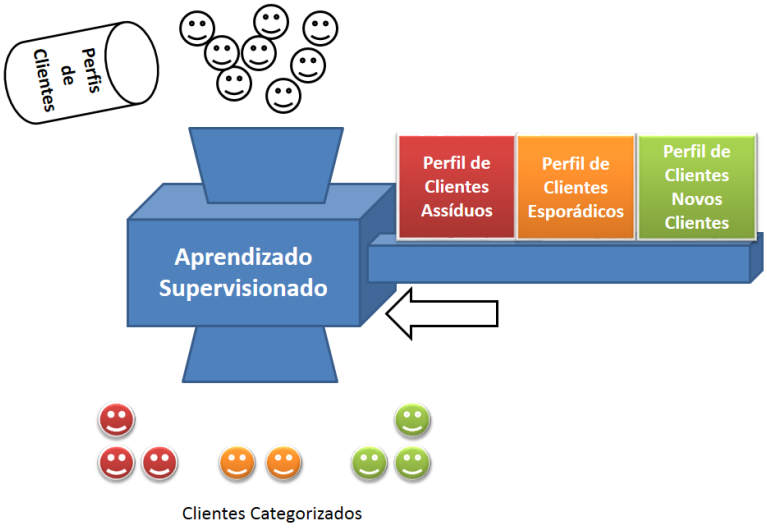


Figura 3.1 Exemplo de Aprendizado Supervisionado

Na figura 3.1 é apresentada a ilustração do exemplo de aprendizado supervisionado. São inseridos perfis de clientes que ainda não foram rotulados (categorizados em assíduos, esporádicos e novos). O sistema precisa indicar, dadas as informações de um cliente novo, qual o perfil adequado a ele, com base no conhecimento a priori (que é representando pelo conjunto de dados do qual se extrai o modelo e o qual foi rotulado por um especialista).

A classificação é a tarefa mais comum no aprendizado de máquina supervisionado. Classificar é a tarefa de organizar objetos, rotulando-os com uma dentre várias classes possíveis. É um problema que reúne várias aplicações diferentes como detecção de SPAM em mensagens de e-mail com base no cabeçalho da mensagem, categorização de animais com base em suas características.

Figura 3.2 Mapeando um conjunto de atributos x num rótulo de classe y

Os dados de entrada nas tarefas de classificação são um conjunto de instâncias ou registros (ou exemplos) de bases de dados. A aprendizagem do modelo de classificação determinará uma *função alvo f* que mapeará cada conjunto de atributos *x* para uma classe pré-determinada *y*. Esta função alvo é conhecida como o próprio modelo de classificação.

Um modelo de classificação pode servir para explicar as diferenças entre as classes. Neste caso é chamado de *Modelo Descritivo*. Um modelo de classificação também pode ser usado para prever o rótulo da classe de uma instância não rotulada e, por isto, é chamado de *Modelo Preditivo* [30].

Uma técnica de classificação é uma abordagem sistematizada para construção de modelos de classificação a partir de um conjunto de dados de entrada. Alguns exemplos são Árvores de Decisão, Classificadores Baseados em Regras, Redes Neurais, Máquinas de Vetor de Suporte e Classificadores *Naive Bayes*.

Cada técnica envolve um algoritmo de aprendizagem. A figura 3.3 mostra uma ideia geral da tarefa de classificação. Inicialmente é escolhido um conjunto de treinamento cujas instâncias já estão rotuladas. O conjunto de treinamento é usado para construir um modelo de classificação (função alvo y). Depois, o modelo é submetido a um conjunto de testes para analisar o seu desempenho.

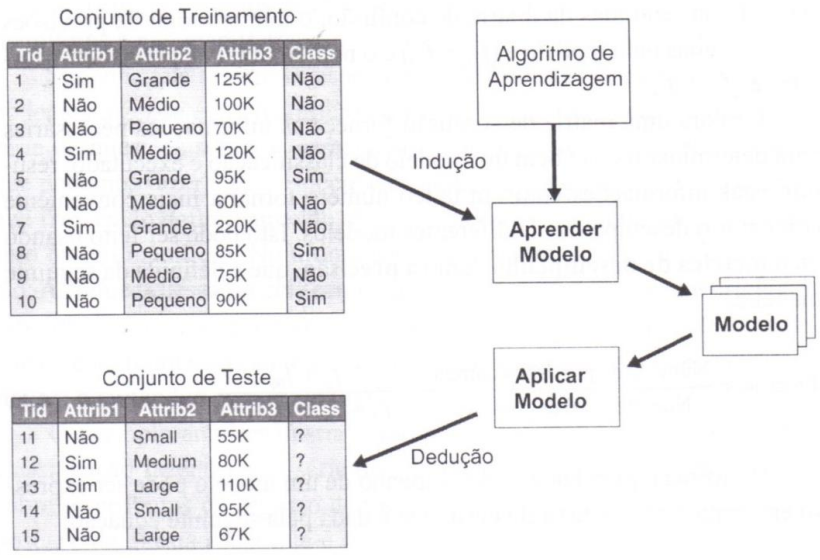


Figura 3.3 – Ideia geral para a tarefa de classificação e indução de um modelo [30]

A avaliação de desempenho reflete a contagem de instâncias de teste que foram correta e incorretamente rotuladas (classificadas) pelo modelo. O resultado

disto é chamado *matriz de confusão*. Cada entrada na matriz mostra os registros previstos de maneira correta ou incorreta.

Tabela 3.1 Exemplo de Matriz de Confusão para um Classificador Binário. Adaptado de [30]

		Classe Prevista	
		Classe = 1	*Classe = 0*
Classe Real	*Classe = 1*	f_{11}	f_{10}
	Classe = 0	f_{01}	f_{00}

A tabela 3.1 mostra a matriz de confusão para um caso de classificação binária, ou seja, onde existam apenas duas classes.

O desempenho do modelo também pode ser calculado com base em métricas de precisão, como mostra a fórmula 3.1 ou com base em sua taxa de erro, mostrado na fórmula 3.2.

$$Precisão = \frac{Número\ de\ Previsões\ Corretas}{Número\ Total\ de\ Previsões} = \frac{f_{11} + f_{00}}{f_1 + f_{10} + f_{11} + f_{00}} \quad 3.1$$

$$Taxa\ de\ Erro = \frac{Número\ de\ Previsões\ Incorretas}{Número\ Total\ de\ Previsões} = \frac{f_{10} + f_{01}}{f_1 + f_{10} + f_{11} + f_{00}} \quad 3.2$$

O algoritmo kNN

O *kNN* (*k Nearest Neighbors* – k vizinhos mais próximos) é um classificador que utiliza a ideia de vizinhança entre as instâncias que servirão para rotular outra instância da base de dados. Dado uma instância a ser classificada o kNN encontra seus vizinhos, ou seja, as instâncias mais parecidas com a instância alvo e calcula uma função de predição que determinará o valor para a instância alvo.

A escolha do valor *k* pode ser um problema para esta abordagem. Se *k* for pequeno demais, então o classificador estará suscetível ao *overfiting*. Se *k* for grande demais o classificador pode classificar erroneamente a nova instância por conta de uma vizinhança grande demais que não caracterize um verdadeiro grupo de afinidades entre as instâncias. No exemplo da figura 3.4, cada tamanho de vizinhança para *k* implica na mudança de sua predição. Para o *k=1* a predição seria uma classe negativa (-) já para *k=3* a predição seria para uma classe positiva (+).

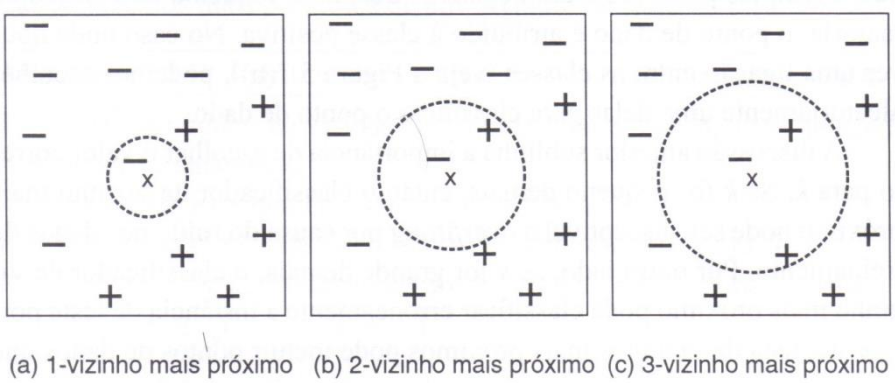


Figura 3.4 – Tamanho da vizinhança no kNN [30]

O algoritmo a seguir resume o método de classificação para o *kNN.* O algoritmo calcula a distância entre cada exemplo de teste (instância alvo a ser classificada) $z = (x', y')$ e todos os seus exemplos de treinamento $(x, y) \in D$ para determinar seus vizinhos mais próximos D_z. Se o número de exemplos de treinamento for grande o cálculo pode ser custoso. Após o cálculo dos vizinhos mais próximos, a instância alvo é classificada baseando-se na classe mais que mais ocorre nos seus vizinhos. Neste algoritmo v é um rótulo de classe e y_i é o rótulo de classe para os vizinhos mais próximos. I( ) é a função que determina o valor de v e pode ser substituída por funções que calculem distâncias, por exemplo.

Algoritmo para o *kNN* [30]

1: Seja k o número de vizinhos mais próximos e D o conjunto de exemplos de treinamento

2: Para cada exemplos de testes $z = (x', y')$ faça

3: Calcule $d(x', x)$, a distância entre z e cada exemplo $(x, y) \in D$

4: Selecione Dz D o conjunto dos k exemplos de treinamento para z

5: $$y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i, y_i) \in D_z} I(v = y_i)$$

6: Fim para

Os classificadores kNN estão entre os mais simples de todos os algoritmos de aprendizado de máquina. Apesar disto, a abordagem do kNN é a mais comumente usada na Filtragem Colaborativa, por assemelhar-se com o conceito deste tipo de filtragem [5].

A vantagem do *kNN* é não gerar um modelo, mas sim considerar todas as instâncias. A desvantagem é o custo desta abordagem, pois quanto mais avaliações recebe um sistema de recomendação, maior a quantidade de informações a considerar.

Árvores de Decisão

Árvores de Decisão são classificadores cujo modelo gerado é uma estrutura de árvore. Existem diversos algoritmos para indução de árvores de decisão como o C4.5 [34] e o ID3 [31].

Em todos os casos é preciso o uso de uma métrica para seleção da melhor divisão. Em geral, usa-se a Entropia [30].

Uma opção para o uso de Árvores de Decisão num Sistema de Recomendação é a formação de *ranking* de itens, como foi proposto em [31].

Redes Bayesianas

Um classificador Bayesiano é uma estrutura que representa probabilidades de eventos definida com base na probabilidade condicional e no teorema de Bayes. Um classificador Bayesiano muito comum é o *Naive Bayes*. Este classificador assume independência condicional dos atributos para estimar probabilidade condicional, ou seja, a presença ou ausência de um atributo não está relacionado a outro. A independência condicional é dada pela fórmula 3.3.

$$P(X \mid Y, Z) = P(X \mid Z) \qquad 3.3$$

X é condicionalmente independente de *Y*, dado *Z* se o resultado da fórmula for equivalente nos dois lados.

Uma das vantagens do *Naive Bayes* é que são classificadores robustos. Formalmente, um Classificador *Naive Bayes* pode ser definido pela fórmula 3.4.

$$P(Y \mid \boldsymbol{X}) = \frac{P(Y)\prod_{i=1}^{d} P(X_i \mid Y)}{P(\boldsymbol{X})}. \qquad 3.4$$

Cada conjunto de instâncias $X = \{X_1, X_2, ..., X_n\}$ consiste de *d* atributos. Para classificar uma instância o *Naive Bayes* calcula a probabilidade posterior para cada classe *Y.* P(X) é fixo para cada Y, basta escolher a classe que maximizar o numerador.

No entanto, a independência assumida pode não ser sempre uma boa alternativa. Neste caso a abordagem utilizada é a de Redes Bayesianas, chamadas de *Bayesian Belief Networks (BBN)*.

As Redes Bayesianas são grafos acíclicos direcionados cujos nós representam variáveis aleatórias. Cada nó associa-se a uma distribuição de probabilidade condicional que se combina com os demais nós para o cálculo da probabilidade de uma determinada hipótese.

A utilização das redes bayesianas assume que a quantidade de interesses é governada por distribuições de probabilidades e que decisões podem ser tomadas pela observação de informações em conjunto com tais distribuições. Promove um método quantitativo que suporta hipóteses alternativas baseado em evidencias. As Redes Bayesianas suportam tanto a abordagem colaborativa quando a baseada em conteúdo [28] [32].

Classificadores Bayesianos são populares em Sistemas de Recomendação baseados em Modelo. No trabalho de [33] é usado um classificador *Naive Bayes* para implementar um Sistema de Recomendação com Filtragem Baseada em Conteúdo. Também em [34] é implementado um Sistema de Recomendação com o uso de um Classificador *Naive Bayes*.

Alguns outros trabalhos têm evidenciado ganhos significativos de desempenho com o uso do *Naive Bayes* na implementação de Sistemas de Recomendação [35]. Em [36] foi testada uma abordagem de Sistema de Recomendação usando *Naive Bayes* com duas classes: assistiu ou não assistiu. Os perfis dos usuários contêm o número de vezes que assistiu determinado programa e são usados para predição de novas recomendações.

Redes Neurais

Também conhecidas como Redes Neurais Artificiais (ANNs – do inglês *Artificial Neural Networks*), provêm um método para aprendizagem de valores reais e discretos em bases de informações que possuam dados ruidosos (com grande discrepância ou possibilidade de). Através de redes interconectadas que definem pesos nas conexões entre os nós, calculam-se valores de saída baseando-se num conjunto de entrada de

dados. A rede procura aprender os pesos ideais para determinados valores e, assim, descobrir o padrão de comportamento necessário para predizer relações. O estudo de ANNs inspirou-se em tentativas de simular sistemas neurais biológicos.

As ANNs podem ser usadas para construção de Sistemas de Recomendação baseados em modelos. No entanto, não há estudos conclusivos sobre os ganhos através do uso de ANNs. Em [38], há um estudo experimental sobre o uso de vários algoritmos de aprendizado de máquina sites de recomendação. Compararam classificadores Naive-Bayes com técnicas para Árvores de Decisão e Redes Neurais. As ANNs não mostraram ganhos significativos. No entanto, ANNs podem ser usadas para combinar vários módulos de recomendações ou várias fontes de dados, como mostra [39] na construção de um Sistema de Recomendação para TV a partir de quatro fontes de dados diferentes: perfis de usuários, informações de comunidade, metadados dos programas e contexto de exibição dos programas. Outro trabalho mostra o uso de uma rede neural para construção de um Sistema de Recomendação baseado em conteúdo. A rede neural usa informações como “tipos”, “estrelas” e “sinopses” para gerar classificação [21].

Máquinas de Vetor de Suporte

O principal objetivo de uma Máquina de Vetor de Suporte (SVM – do inglês *Support Vector Machine*) é encontrar um hiperplano linear que separe os dados de tal forma que a margem de separação entre eles é maximizada. Nas SVMs, a função de decisão é especificada por um subconjunto de exemplos de treino: os vetores de suporte. O hiperplano é chamado de *hiperplano de margem máxima.*

Por exemplo, se o objetivo é separar um conjunto que possui duas classes possíveis, num cenário com duas dimensões. Cada classe será separada e terá uma margem de separação entre elas, como mostra a figura a seguir.

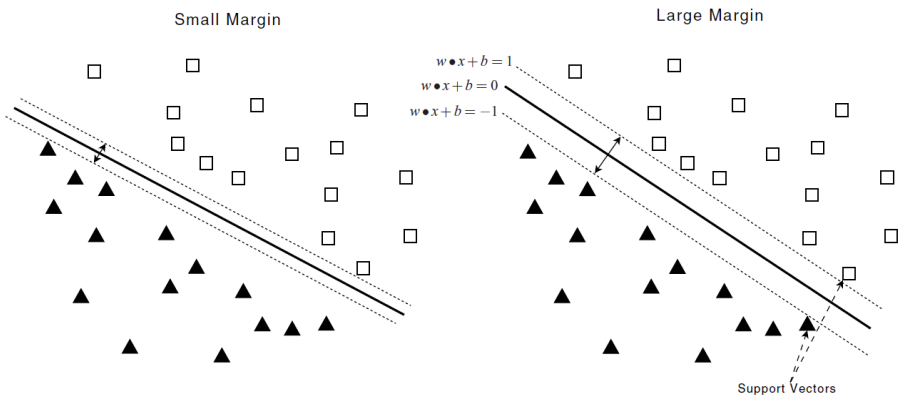


Figura 3.7 – Exemplo de SVM [30]

Se a SVM escolhe a função que maximiza a margem então a propensão a erros futuros de classificação é menor. A separação linear entre as duas classes é dada pela fórmula $w \bullet x+b = 0$ e a função que pode classificar os conjuntos de interesse em duas classes (-1 ou +1, quadrados ou triângulos) pode ser dada pela fórmula 3.6.

$$f(x) = \begin{cases} 1, & \text{if } w \bullet x + b \geq 1 \\ -1, & \text{if } w \bullet x + b \leq -1 \end{cases} \qquad 3.6$$

SVMs em Sistemas de Recomendação têm mostrado resultados interessantes e promissores, principalmente em áreas de aplicação como a recomendação em TV Digital. No trabalho de [40], por exemplo, busca-se através de um estudo experimental selecionar a melhor técnica de pré-processamento para predizer os valores de avaliações faltantes num Sistema de Recomendação baseado em SVM. Em [41] a SVM é utilizada para construção de um Sistema de Recomendação que utiliza informações da grade de programação (EPG – *Electronic Program Guide*) como fonte de dados. E, em [42], os autores apresentam o uso de SVM num cenário de Filtragem Colaborativa.

3.2.2 Aprendizado Não Supervisionado

O aprendizado não-supervisionado é aquele que se dá pela observação de informações e descoberta de padrões entre estas que permitam a formação de grupos distintos.

A tarefa do algoritmo é, então, agrupar exemplos não rotulados, ou seja, exemplos que não possuem classe especificada. Através deste aprendizado são descobertas "regularidades" entre as informações agrupadas em padrões. Os padrões são comumente chamados de clusters. Os exemplos de um cluster são muito similares entre si e diferenciam-se dos exemplos de outros clusters. Esta medida é feita a partir de alguma métrica de similaridade [43].

Aproveitando o sistema de recomendação para livros e CDs do item 3.2.1, de acordo com a figura 3.8, imagine que o sistema possua diversas informações sobre perfis de clientes, mas nenhuma classe. As entradas para um método de aprendizado não-supervisionado seriam perfis de clientes e suas informações. Com base nestas informações o sistema utilizará alguma medida de similaridade para "separar" ou "agrupar" os perfis em clusters. O número de clusters desejados é uma entrada para o sistema também.

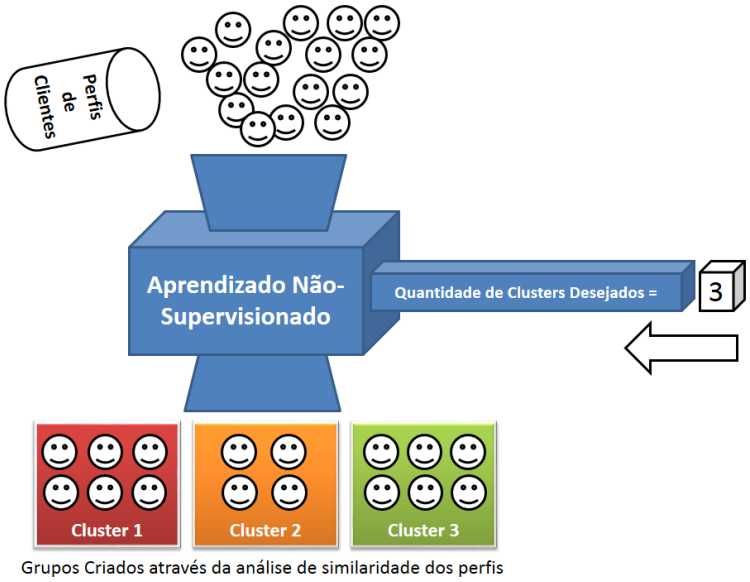


Figura 3.8 Exemplo de Aprendizado Não-Supervisionado

Esta técnica divide um conjunto de objetos de acordo com os valores de seus atributos, não necessitando de atributo objetivo. Assim, o algoritmo é responsável pela separação em grupos distintos (clusters), de acordo com os valores dos atributos. Os

elementos de um grupo são similares entre si e diferem na relação intergrupo. O algoritmo é não-supervisionado, pois não há informação *a priori* que distingue os elementos em classes. Resumidamente, a análise de grupos agrupa objetos baseada apenas em informações encontradas nos dados que descrevem os objetos e seus relacionamentos.

O Algoritmo K-Means

A técnica de agrupamento usada pelo algoritmo K-Means é simples. Primeiro são escolhidos *K* centróides iniciais. *K* é um valor estipulado pelo usuário que representa o número de grupos desejados. Cada ponto é atribuído ao centróide mais próximo e cada coleção de pontos atribuídos a um centróide é um grupo. O centróide de cada grupo é então atualizado baseado nos pontos atribuídos ao grupo. Estes passos são repetidos até que nenhum ponto mude de grupo ou quando os centróides permanecerem os mesmos.

Algoritmo para o K-Means [30]	
1:	Selecione K pontos como centróides iniciais
2:	**Repita**
3:	Forme K grupos atribuindo cada ponto ao seu centróide mais próximo
4:	Recalcule o centróide de cada grupo
5:	**Até que** os centróides não mudem

O cálculo para a formação de grupos é feito usando métricas de similaridade como as que foram apresentadas anteriormente, sendo a Distância Euclidiana a mais usada.

3.2.3 Aprendizado Semissupervisionado

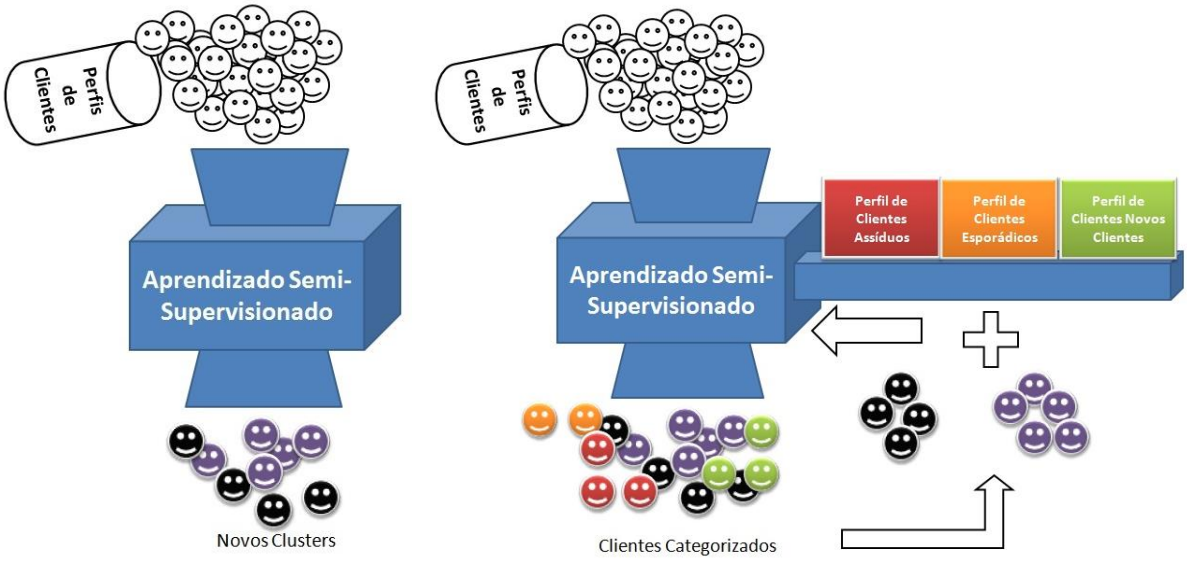
No Aprendizado Semissupervisionado, uma parte dos dados utilizados no treinamento é classificada, enquanto outra consiste de dados não-rotulados.

Tal paradigma de aprendizado é particularmente útil em casos onde a amostragem limitada do conjunto de treinamento não fornece informação suficiente

para a indução de uma regra geral. Assim, utiliza-se o conjunto de teste como fonte extra de informação para a resolução do problema.

Esta abordagem é útil para os problemas onde o espaço amostral é muito grande ou nos casos em que o classificador possui um custo computacional muito alto ou um alto grau de especialização. Um exemplo de problema para esta abordagem é a classificação de textos provindos da Internet, pois se trata de um grande volume de exemplos. Mesmo sem um conjunto de treinamento suficiente, uma máquina de busca e classificação pode conseguir bons resultados.

Aproveitando o exemplo explicado até agora, no aprendizado semissupervisionado seriam criados novos clusters iniciais a partir de dados não rotulados e estes auxiliariam o processo de classificação usando as classes anteriormente existentes.



a) b)

Figura 3.9 Exemplo de Aprendizado Semissupervisionado

Como podemos ver na figura 3.9, seriam criadas novas categorias de clientes a partir de clusters novos e estas seriam utilizadas na classificação de outros clientes posteriormente.

De acordo com [44] é possível dividir o Aprendizado Semissupervisionado em várias estratégias. A este trabalho, particularmente interessam:

- *Self-Training* (ou *Bootstrapping*): no qual o algoritmo reinsere novas instâncias classificadas por ele mesmo no conjunto de treinamento;

- *Co-Training:* no qual as características de uma instância podem ser divididas em dois conjuntos diferentes e cada conjunto será usado para treinar um classificador. Um classificador "ensina" o outro sobre a instância não rotulada e vice-versa. O processo se repete;
- *Multiview Learning*: podem ser usados diferentes classificadores para o mesmo conjunto de dados ou para o conjunto subdividido (similar ao *co-training*, porém com algoritmos diferentes).

3.3 Aprendizado Sem Fim

O paradigma de Aprendizado Sem Fim foi desenvolvido como uma resposta da comunidade de Aprendizado de Máquina às diferenças observadas nas habilidades de aprendizagem de homem e máquinas [45].

O avanço e conquista nas áreas de Aprendizado de Máquina e Mineração de Dados não permitiam o desenvolvimento de sistemas computacionais que aprendessem de maneira incremental e reutilizando os conceitos aprendidos para continuar este processo.

Enquanto pessoas possuem a habilidade de aprender com pequenos conjuntos de exemplos, os algoritmos de aprendizado de máquina demonstram dificuldades com poucas instâncias classificadas.

Pela definição de aprendizado sem fim um agente inteligente deve reduzir sua dificuldade de resolver problemas usando conhecimento adquirido em soluções anteriores. Assim, o processo de aprendizagem sem fim acontece de maneira incremental. Cada conceito aprendido pode ser usado para auxiliar a aprendizagem posteriormente.

Este tipo de comportamento foi e tem sido implementado e analisado em diferentes trabalhos e áreas como a robótica [46] e a extração de conhecimento a partir de grandes fontes como a Web. Para ilustrar o Aprendizado Sem Fim considere o Projeto RTW (*Read the Web*) que desenvolve o NELL (*Never Ending Language Learning*)[5]. O objetivo do NELL é extrair informações da web em inglês e, a partir destas informações melhorar sua ontologia que relaciona os conceitos aprendidos [47] [48] [49] [50].

[5] Detalhes do Projeto http://rtw.ml.cmu.edu

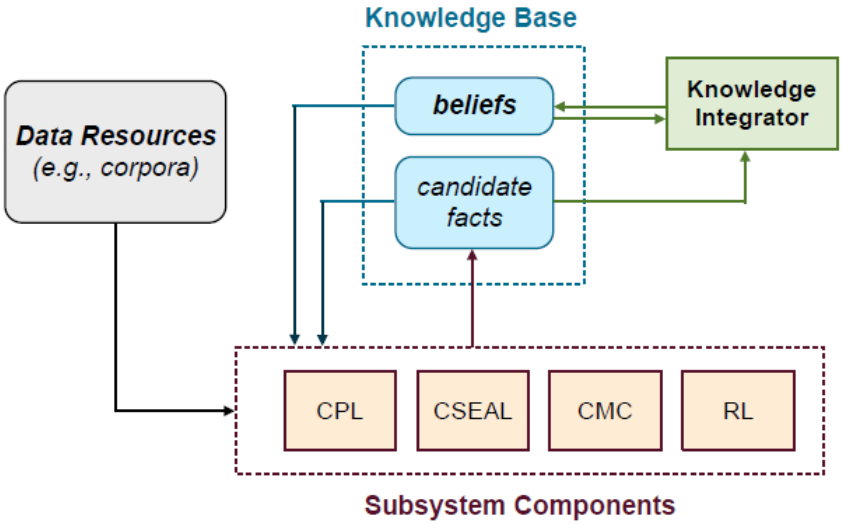


Figura 3.10 – Arquitetura do NELL [48]

A figura 3.10 mostra a arquitetura do projeto NELL, descrito anteriormente. Esta arquitetura conta com os componentes:

- *Knowledge Base* (KB): trata-se da base de conhecimento do NELL que contém fatos candidatos, ou seja, informações ainda não confirmadas segundo a proposta do NELL e crenças que são informações já validadas (por exemplo, ainda não foi confirmado que George W. Bush é um ser humano, mas foi confirmado que ele foi presidente dos Estados Unidos).
- *Data Resources*: São as fontes de dados que "alimentam" os componentes do NELL, incluindo a Internet.
- CPL (*Coupled Pattern Learner*): trata-se de um extrator de texto que aprende e usa padrões textuais como "maior que X" e "X joga com Y" usado para extrair instâncias de categorias e relações. O CPL usa co-ocorrência estatística entre padrões textuais e sintagmas nominais para aprender a extrair padrões em predicados de interesse e então usa estes padrões para encontrar instâncias adicionais para cada predicado.
- CSEAL (*Coupled Set Expander for Any Languge*): trata-se de um extrator semi-estruturado que consulta a Internet para checar informações sobre crenças de cada categoria ou relação. O CSEAL usa relações mutuamente exclusivas para prover exemplos negativos que serão usados para filtrar exemplos aprendidos.

- CMC (*Coupled Morphological Classifier*): trata-se de um conjunto de modelos de regressão logística para classificar sintagmas nominais com base em várias características morfológicas. Crenças (*beliefs*) da base de conhecimento (KB) são usadas como instâncias de treinamento.
- RL (*Rule Learner*): trata-se de um algoritmo de aprendizado relacional de primeira ordem que aprende cláusulas de Horn Probabilísticas. Estas regras aprendidas são usadas para inferir novas instâncias relacionadas de outras instâncias já presentes na base de conhecimento.
- KI (*Knowledge Integrator*): este componente promove fatos candidatos para crenças usando uma determinada estratégia. Fatos candidatos que tem alta confiança de um componente são promovidos e candidatos com baixa confiança são promovidos se forem propostos por múltiplos componentes.

Pode-se verificar, pelo exposto, que o NELL implementa vários algoritmos de aprendizado de máquina numa arquitetura que promova as informações validadas e possa se autossupervisionar.

A autossupervisão e a autorreflexão são, inclusive, duas das principais características do aprendizado sem fim: permitir que um agente aprenda de forma contínua, e, além disso, possua mecanismos para refletir sobre os resultados obtidos e realizar uma série de tarefas de solução de problemas simples para resolver problemas complexos de acordo com sua reflexão. Trata-se de um processo visto como um dos mais instigantes em Inteligência Artificial [51].

Um dos primeiros sistemas a tentar demonstrar este conceito foi o *"Automated Mathematician (AM)"*. Este era um agente que visava descobrir novos conceitos em matemática e os relacionamentos entre eles usando uma estratégia de busca heurística orientada [52], e o primeiro Sistema de Aprendizado Sem Fim a demonstrar na prática que é realmente possível que o sistema seja executado continuamente e não pare de aprender (expandir sua base de conhecimento) foi o NELL [52].

As principais tarefas do NELL consistem em extração de informações continuamente da web e aprendizado de relações entre estas informações visando melhorar a acurácia deste aprendizado. O NELL é um sistema de aprendizado Sem Fim com foco no processamento de linguagem natural, mas que visa também

estruturar o conhecimento difundido na web. O NELL pode aprender com base em relacionamentos entre categorias e relações. *Pessoas, objetos, esportes* são exemplos de categorias e *PessoaPraticaEsporte(nome, nomeesporte)* é um exemplo de relação. É possível navegar pela base de conhecimento no endereço do projeto[6].

Um processo de aprendizado sem fim não deve apenas ser ou representar um modelo estático a ser usado como inferência, mas sim um modelo incremental e com certa "maturidade" e autossupervisão. Por maturidade entenda-se a capacidade de observar conjuntos de treinamento anteriores sob nova perspectiva, visto que o sistema desenvolveu-se e adaptou-se a novas possibilidades. É importante ressaltar que este processo, com o tempo, também resulte em melhor desempenho do sistema, além de incremento do conhecimento adquirido.

O NELL tem em seu trabalho diversos algoritmos, teorias e princípios, mas são de importante destaque [3] [52]:

1. Aprendizado Semissupervisionado [44];
2. Uso de Acoplamento;
3. Uso de Amostras Negativas;
4. Autossupervisão;
5. Autorreflexão;

É possível notar certa similaridade entre um Sistema de Aprendizado Sem Fim e o Aprendizado Semissupervisionado. Em ambas as abordagens de aprendizado o sistema inicia com pouco conhecimento *a priori* e busca aprender mais usando informações que não foram rotuladas anteriormente. Tal similaridade levou à investigação e ao desenvolvimento de métodos de aprendizado semissupervisionado que incorporassem as características específicas de aprendizado sem fim. No entanto, o uso de técnicas de aprendizado semissupervisionado na geração de um sistema de aprendizado sem fim provoca um problema conhecido como "desvio do conceito (*concept drift*)". Isto ocorre quando o sistema classifica um exemplo de maneira incorreta e passa usar este exemplo para rotular novos exemplos. Com o passar do tempo um sistema de aprendizado sem fim apenas baseado no aprendizado semissupervisionado poderia gerar mais conceitos incorretos e isto o tornaria inconsistente ou demonstraria que o aprendizado não foi bem sucedido [53].

[6] http://rtw.ml.cmu.edu/rtw/kbbrowser/

Assim, um sistema de aprendizado sem fim pode valer-se de outras técnicas como o acoplamento de várias estratégias de aprendizado na geração de restrições, métodos supervisionados e não-supervisionados para auxiliar no seu processo de autossupervisão e correção do aprendizado.

Há também a necessidade de um modelo de dados (no caso de um sistema de recomendação, de um modelo de usuário) adequado a este tipo de sistema e que permita a evolução do conhecimento aprendido.

3.4 Considerações Finais

Uma das principais motivações para este trabalho de doutorado é iniciar o uso das técnicas e princípios que possam viabilizar a construção de um Sistema de Aprendizado Sem-Fim (SASF) capaz de realizar a tarefa de recomendação. Dada a complexidade do tema e quantidade de problemas a serem estudados e tratados na área de Sistemas de Recomendação, esta pesquisa deu passos iniciais nesta busca.

Assim, a arquitetura apresentada na seção 4 ilustrará como os conceitos 1 a 4 do NELL, apresentados anteriormente, foram explorados e os resultados obtidos em relação a sistemas de recomendação.

Este trabalho constitui-se de um desafio para área de Sistemas de Recomendação e de Aprendizagem de Máquina, demonstrando um sistema de aprendizado sem fim aplicado a um novo contexto. Assim, através do aprendizado sem fim, busca-se fornecer as características de autossupervisão e autorreflexão a sistemas de recomendação.

Tradicionalmente a interação de um RS (*Recommender System* – Sistema de Recomendação) com o usuário se dá quando o usuário avalia itens. Por isto, quando um usuário novo começa a usar um sistema de recomendação sabe-se muito pouco sobre suas preferências e interesses.

Uma abordagem comum para a aprendizagem de preferências do usuário é pedir-lhe para avaliar uma série de itens (conhecido como pontos de treinamento). Um modelo que aproxima as preferências do usuário é então construído a partir desses dados. Como o número de itens revistos pelo usuário é pequeno e não abrangerá todos os itens do sistema, a precisão do modelo aprendido depende muito de uma boa seleção de pontos de treinamento. Um sistema pode solicitar ao usuário que

avalie Star Wars I, II e III. Ao classificar os três volumes da trilogia, teremos uma boa ideia das preferências do usuário para Star Wars e, possivelmente, por indução, uma indicação para outros filmes dentro do gênero Ficção Científica, mas o conhecimento total sobre este usuário será limitado. É importante citar que escolher itens muito populares como Star Wars pode não gerar muita informação útil, já que, em tese, a grande maioria dará uma nota razoável.

O Aprendizado Sem Fim, o Aprendizado Supervisionado, o Aprendizado Não-supervisionado, e o Aprendizado Semissupervisionado serão usados na proposta deste trabalho para auxiliar um sistema de recomendação nos seguintes pontos:

- Um novo usuário num sistema de recomendação tem “expectativa” de ver sugestões interessantes logo no início. O sistema saberá pouco deste usuário, mas poderá realizar uma avaliação inicial com ele para estabelecer alguns pontos de interesse. Novos produtos introduzidos no sistema também não possuem ainda relação estabelecida com outros produtos.
- Através da implementação do Aprendizado Sem Fim busca-se evitar a superespecialização [5] [54] do sistema como num exemplo em que o usuário assiste filmes como Star Wars e o pontua positivamente e, então, recebe como sugestão a continuação da saga (Star Wars II, ou III). É interessante que o sistema possa evitar ou se adaptar a máximos locais, visando descobrir mais detalhes do perfil do usuário e não um único ponto específico. Através da Aprendizagem Sem Fim o sistema utiliza abordagens supervisionado e não supervisionado, além da semi e de um processo de autossupervisão. Tudo isto para permitir que o mesmo sempre continue a aprender.
- O tradicional objetivo dos Sistemas de Recomendação é diminuir o erro de previsão, mas busca-se neste trabalho mostrar que é importante também que o sistema “tente” alternativas e não apenas a previsibilidade de uma recomendação.

No capítulo 4, a seguir, serão apresentadas as principais técnicas e métodos desenvolvidos neste trabalho.

Capítulo 4

ARQUITETURA DO SISTEMA DE RECOMENDAÇÃO SR-SASF

"The essence of AI is learning and adapting…" [86]

4.1 Detalhamento da Arquitetura

Este trabalho apresenta uma nova abordagem na construção de Sistemas de Recomendação. Trata-se da abordagem que se orienta pelos conceitos de um Sistema de Aprendizado Sem Fim (SASF). Esta abordagem recebeu o acrônimo de SR-SASF (Sistema de Recomendação inspirado no design de Sistemas de Aprendizado Sem-Fim) e em inglês RSA-NEL (*Recommendation System Architecture inspired by Never Ending Learning design*).

Num Sistema de Recomendação tradicional a sua arquitetura divide o Aprendizado numa etapa off-line (como pode ser visto na figura 4.1) e a Recomendação é feita online (figura 4.2).

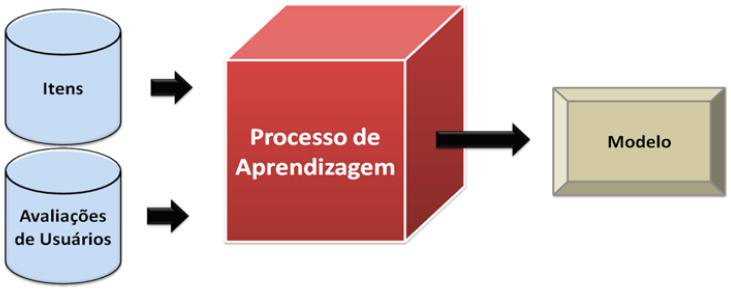


Figura 4.1 – Processo de Aprendizagem Off-line de um Sistema de Recomendação

Isto ocorre devido ao alto custo do Processo de Aprendizagem quando o número de Itens e Usuários aumenta e porque é comumente usada a abordagem de

cálculo de vizinhanças (*kNN)* na Filtragem Colaborativa e cálculos de similaridade, como o cosseno, na Filtragem Baseada em Conteúdo. Ambos necessitam avaliar todas relações para este cálculo.

As abordagens baseadas em modelo têm aparecido em alguns trabalhos de sistemas de recomendação, mas apresentam *overfiting*, visto que os sistemas de recomendação são usualmente aplicados a ambientes dinâmicos com alto crescimento da base de dados de informações tanto de usuários quanto dos itens com os quais se relacionam.

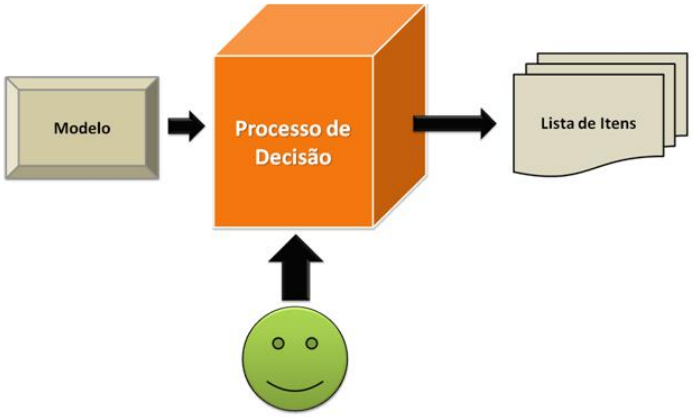


Figura 4.2 – Processo de Apresentação de Recomendações Online

Motivado pelo uso do Aprendizado Sem Fim, este trabalho apresenta uma abordagem para Sistemas de Recomendação que possa não só utilizar o conceito de Sistema de Recomendação Híbrido (valendo-se de mais de uma abordagem dos sistemas de recomendação), mas que possua um modelo de aprendizado contínuo (considerando novos dados chegando), multicritério (considera mais de uma visão – no nosso caso, mais de um classificador) e sensível ao contexto (adaptada ao contexto da informação).

O aprendizado contínuo se dará por um modelo que possa evoluir sem critério de parada. Além disto, o trabalho pode usar como fonte diversos algoritmos como o *kNN*, *K-Means*, *Naive Bayes e o C4.5* como forma de permitir uma análise com mais de uma perspectiva possível para os dados. Também faz a integração com o usuário de maneira a recolher informações de *feedback* e de controle e pode se adaptar, por exemplo, a novos contextos de um usuário.

Como apresentado na figura 4.3, a arquitetura proposta para um Sistema de Recomendação SASF (Sistema de Aprendizado Sem-Fim) foi inspirada na arquitetura

do sistema NELL (descrito na seção 3.3) e será guiada por três componentes importantes:

1. Estratégias
2. Heurísticas
3. Contexto

As estratégias envolvem os algoritmos desenvolvidos e os algoritmos utilizados. As heurísticas envolvem os problemas a serem resolvidos e as hipóteses assumidas para tratá-los. Já o contexto refere-se ao trabalho que será feito, inicialmente, a cada base de dados, a fim de transformar dados matriciais em dados vetoriais para o uso em classificadores.

Figura 4.3 – Resumo da Arquitetura Proposta

O detalhamento da Arquitetura de um Sistema de Recomendação SASF, na figura 4.4 (também inspirada no sistema NELL), possui os seguintes componentes:

- Fontes de Informações: trata-se da base ou bases de dados que servirão como fonte ao Sistema de Recomendação.
- Gerador de Modelos com Base em Supervisão: este componente será responsável por criar um modelo descritivo para o usuário. Como não será o único gerador de recomendações, pretende-se utilizar algoritmos que não necessitem, obrigatoriamente ter um ótimo desempenho inicial, já que será assessorado por outros algoritmos, mas que possam ser aplicados em tempo real.

- Gerador de Grupos: este componente usará um algoritmo não-supervisionado para particionar o conjunto de dados e auxiliar a criação de modelos de usuários mais específicos, quando necessário.
- Analisador de Contexto: este componente faz um tratamento inicial dos dados convertendo-os num formato passível de serem classificados. Tradicionalmente o cálculo em Sistemas de Recomendação envolve Matrizes de Dados e classificadores lidam com vetores rotulados.

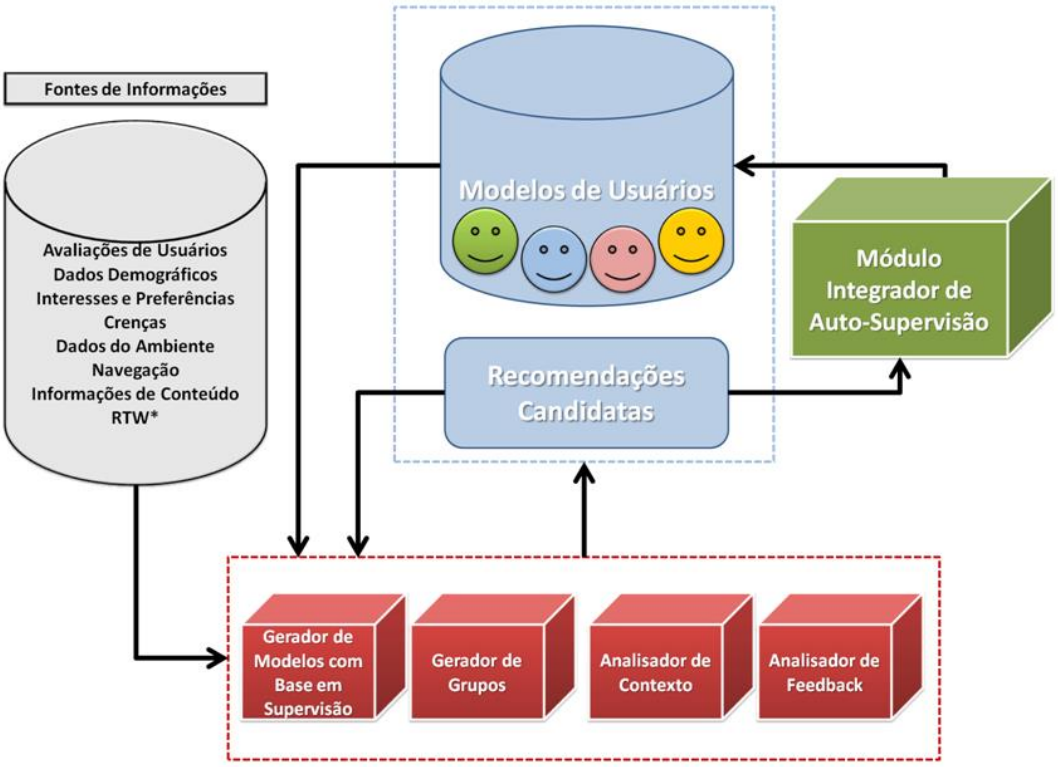


Figura 4.4 – Abordagem Proposta

Continuando o detalhamento da Arquitetura da figura 4.4 existem também:

- Analisador de Feedback: este componente visa comparar dados vindos do ambiente real com dados gerados pelo sistema. Por exemplo, uma recomendação de um item pode, posteriormente, gerar uma avaliação, e esta avaliação substituirá a predição feita.
- Recomendações Candidatas: as recomendações candidatas farão parte de um conjunto de recomendações geradas por todos os algoritmos usados. Deste conjunto, algumas integrarão o modelo dos usuários e serão analisadas, posteriormente, pelo *Feedback.*

- Módulo Integrador de Autossupervisão: este módulo tem como tarefa analisar a progressão de um modelo de usuário. Verificará, por exemplo, se este modelo encontra-se sem evolução e, através do uso dos outros módulos, pode, por exemplo, questionar o usuário ou mesmo privilegiar a geração de informações por um dos componentes. Isto é importante para que o modelo de usuário possa adaptar-se com o tempo e não seja um modelo tendencioso, buscando sempre minimizar o *overfiting.*
- Modelos de Usuários: são modelos com informações sobre os usuários, as recomendações geradas, recebidas, utilizadas e não utilizadas. Também servem como fonte de dados para os demais componentes.

4.2 Algoritmos Utilizados

Tradicionalmente a Filtragem Colaborativa envolve um cálculo matricial como pode ser visto na figura 4.5. Como já discutido na seção 2 e na seção 4 do capítulo 2, existem diversas formas de se predizer o *rating* de um usuário a um dado item. Na filtragem colaborativa são consideradas as “opiniões” da vizinhança do usuário-alvo.

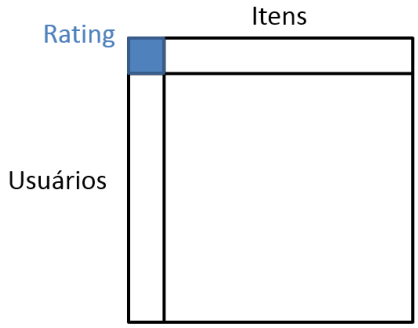


Figura 4.5 – Abordagem Tradicional na Filtragem Colaborativa

A Filtragem Colaborativa, como discutido na seção 2.7, fornece as melhores vantagens em sistemas de recomendação como “pensar fora da caixa” (recomendar novidades reais), focando em características da comunidade, mas também apresenta alguns problemas a serem solucionados em ambientes reais de aplicação. A Filtragem Colaborativa não considera os dados de perfis dos usuários e também os dados relacionados aos itens. Ou seja, ela apenas considera a relação usuário x item e o resultado desta relação que é o *rating* (representando a opinião do usuário sobre o item).

Como pode ser visto na figura 4.6 existem informações pertinentes a itens e usuários nas bases de dados, descrevendo-os.

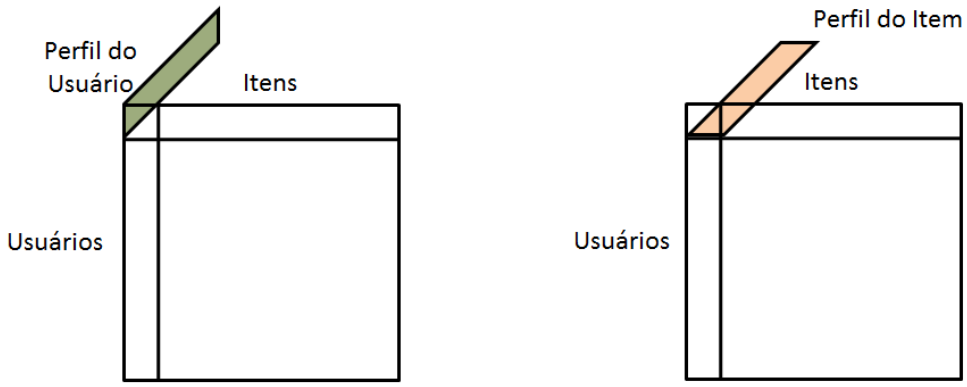


Figura 4.6 – Características dos Itens e dos Usuários

Através de uma tarefa de pré-processamento e *data warehousing* é possível combinar as informações de itens e usuários para a criação de um perfil descritor, cuja classe é a avaliação do usuário (o *rating* dado). Esta tarefa é feita pelo *Analisador de Contexto* e, no momento, não é automatizada, ou seja, as consultas à base de dados, a extração de resultados e conversão são feitas por um operador humano. No entanto, trabalhos futuros incluem a automatização da mesma em cenários que atendam critérios específicos (dados descritores de usuários e itens, nomes-padrão em tabelas de dados, etc).

Assim, como mostra a figura 4.7, é possível definir uma função de combinação, dependente de domínio, mas que pode ser generalizada para descritores do usuário, descritores do item e o rating dado pelo usuário a este item. Com isto, o cálculo matricial antes realizado na filtragem colaborativa, torna-se agora um cálculo de tuplas, vetores na forma $t1 = \{a_1, a_2, a_3, ..., a_n, Classe\}$ onde a_i representa um atributo daquela tupla.

Por exemplo, considere o usuário $u1 = \{a_1, a_2, a_3, ... a_n\}$ e o filme por ele avaliado $f1 = \{b_1, b_2, b_3, ... b_n\}$ Após a função de composição o resultado final será uma tupla $t1(u1 + f1) = \{a_1, a_2, a_3, ... a_n, b_1, b_2, b_3, ... b_n, aval(u1, f1)\ \}$ onde $aval(u1,f1)$ é o valor dado na avaliação do filme $f1$ pelo usuário $u1$.

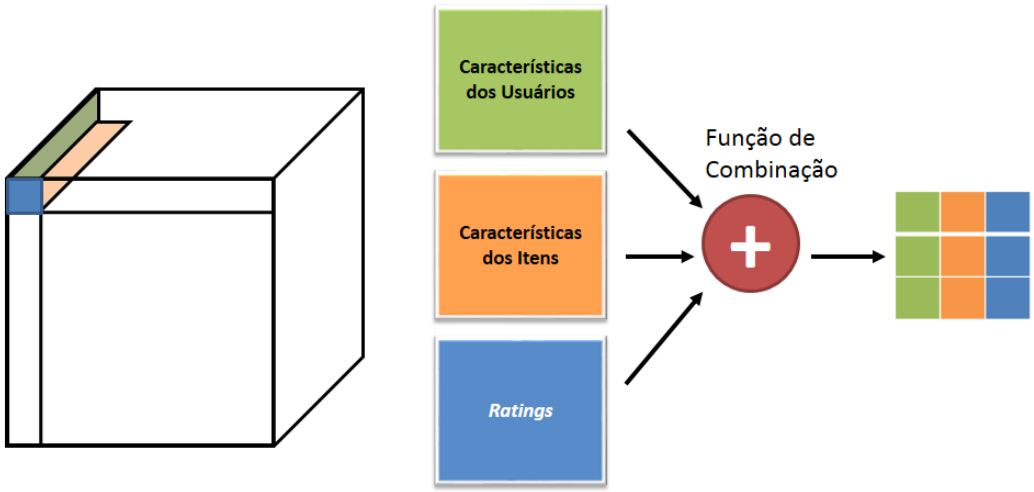


Figura 4.7 – Combinando dados demográficos e ratings

A partir desta tarefa de combinação e pré-processamento será possível utilizar algoritmos de classificação para a tarefa de predição.

Após o uso de algoritmos de classificação novas tuplas serão obtidas pela predição de suas classes. A partir daí, o processo de decomposição é parecido com o processo anterior, mas inverso. Por exemplo, após a classificação serão buscadas tuplas com valores iguais nos atributos usados e o valor obtido na avaliação será inserido nas mesmas.

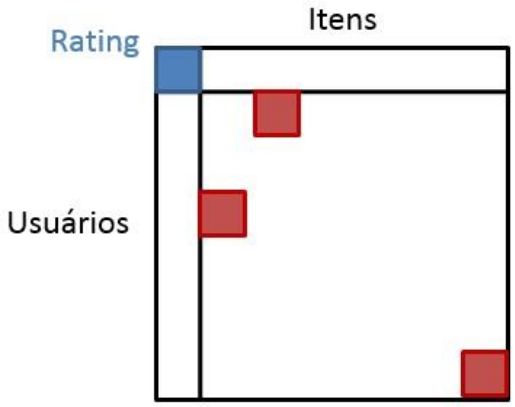


Figura 4.8 – Inserido valores na matriz através da decomposição do vetor de dados (tuplas obtidas no treinamento).

Trata-se de um processo de preenchimento da matriz de dados. Por exemplo, considere a nova tupla classificada no processo do Algoritmo 1 (descrito abaixo):

$T2 = \{a_1, a_2, a_3, \ldots a_n, b_1, b_2, b_3, \ldots b_n, aval\}$

O que o Analisador de Contexto faz na decomposição é procurar na matriz os usuários e filmes com atributos iguais aos da tupla *t2* e considerar como função de avaliação (o ponto de intersecção da matriz) o valor que veio no atributo *aval* de t2.

Na literatura, o processo de aprendizado de máquina apresenta algumas abordagens sobre como decidir sobre novos dados adquiridos e seus relacionamento. O aprendizado supervisionado usa dados já rotulados previamente para predizer os rótulos de novos dados. Esta abordagem também depende de um especialista que verifica os resultados e se eles estão de acordo com resultados esperados/desejados. A participação especialista ocorre no começo do processo, ao inserir dados já rotulados e na conferência dos resultados obtidos no processo. A maior vantagem do aprendizado supervisionado é a consistência estabelecida pelas relações descobertas. Em contrapartida, é necessária uma base de dados considerável para que estas relações tenham um desempenho satisfatório.

Para suprir a falta desta quantidade de dados, neste trabalho é apresentada uma abordagem de aprendizado sem-fim usando múltiplos algoritmos e visões para garantir que os dados obtidos aproximem-se ao máximo do conjunto real e suas relações existentes.

É importante citar que a autossupervisão é feita internamente através da própria integração de algoritmos diferentes, colaborando entre si, e externamente ao componente através do componente *Gerador de Grupos.*

Para aumentar o tamanho da base de dados disponível foi utilizada a abordagem de *Bootstrapping* na qual o conjunto de treinamento utilizado vai sendo incrementado escolhendo-se as melhores instâncias classificadas do conjunto de teste. Nos testes foi usada a probabilidade de acerto do classificador nos testes com o NaiveBayes. O problema é que o uso de apenas um classificador para a tarefa não apresentou bons resultados. A partir de um determinado tamanho de conjunto de treinamento o classificador passava a diminuir continuamente sua taxa de acerto (cerca de 10% da base no experimento 2 e 60% da base no experimento 1). O mecanismo interno de controle desenvolvido usa mais de uma visão proporcionada por mais de um classificador atuando em conjunto com outro e com dados de treinamento independentes. A solução apresentada na figura 4.8 usa a abordagem de aumentar o tamanho do conjunto de treinamento a cada execução do algoritmo. De maneira resumida o algoritmo é mapeado da seguinte forma:

1. O conjunto de treinamento (dados já rotulados) é dividido entre os classificadores;
2. Cada um deles gera um modelo e faz a classificação das instâncias não rotuladas (conjunto de testes);
3. Os dados já rotulados são ordenados de acordo com a probabilidade de acerto dada pelo seu classificador ou classificadores;
4. As instâncias melhores ranqueadas são escolhidas para serem inseridas no conjunto de treinamento do outro classificador (ou seja, um elege as melhores instâncias e as insere no outro e vice-versa). O intuito disto é fazer com que os classificadores cooperem entre si enquanto "decidem" as classificações dadas.
5. O processo é repetido e a base de treinamento vai aumentando seu tamanho.

Está técnica é conhecida como *bootstrapping* e o problema da mesma é que muitos erros podem ser propagados nos dados rotulados. Para diminuir este efeito foi usado mais de um algoritmo trabalhando em conjunto e rotulando os dados de outro algoritmo, de forma cruzada. Este acoplamento permite um modelo com melhor desempenho e é baseado no conceito de *Co-Training* [55].

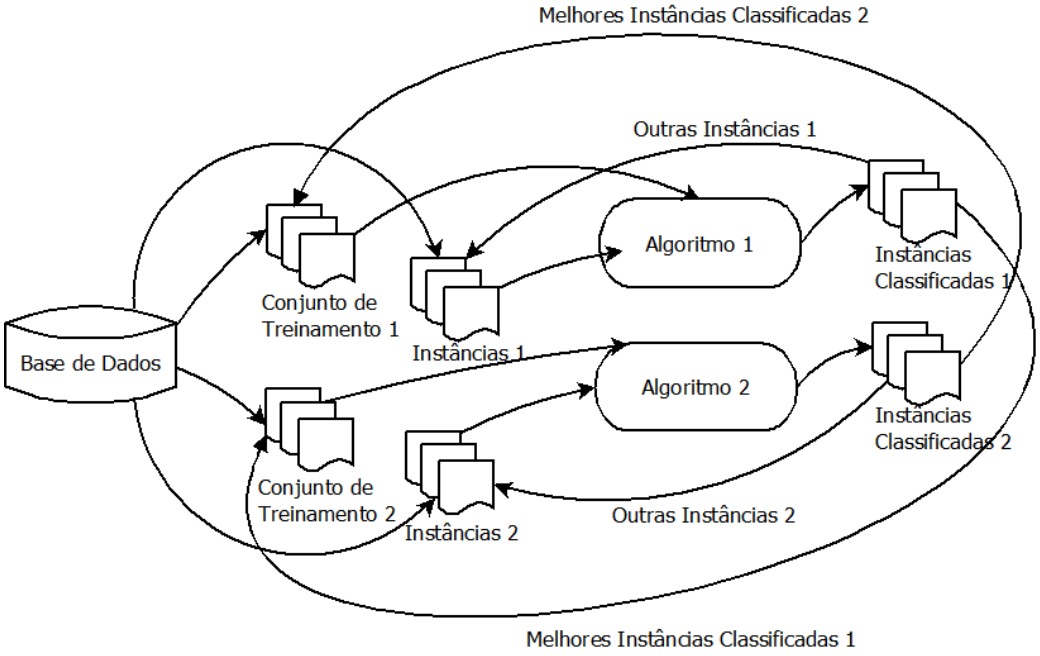


Figura 4.9 Aprendizado por Acoplamento e o Modelo usado para garantir novas instâncias.

Como é possível observar na figura 4.8 os dois algoritmos possuem seu próprio conjunto de instâncias que é dividido entre treinamento e testes. Cada algoritmo será

executado e gerará instâncias classificadas com base no seu modelo gerado. Estas instâncias serão ordenadas de acordo com o critério de probabilidade de acerto. As primeiras *N* instâncias serão inseridas no conjunto de treinamento do outro algoritmo. Esta abordagem permite que o algoritmo possa melhorar seu conjunto de treinamento a partir da visão e resultados de outro algoritmo. O *bootstrapping* sozinho não é suficiente para obter os mesmos resultados. O algoritmo desenvolvido está descrito abaixo:

Algoritmo 1: Arquitetura Proposta

```
Input: A – Conjunto de Dados é extraído da base
W <- parse(A) // o conjunto de dados é estruturado em W
R = getInstances(W, 0.05)  // conjunto de treinamento
V = getInstances(W, 0.95)  // conjunto de testes
// Veja que W = R ∪ V
C <- set of classifiers
for Ci in C do // faça iteração para todos os classificadores
   Ti <- R  // conjunto de treinamento
   Si <- V  // conjunto de testes
repeat
   for Ci in C do // para cada classificador
      trainClassifier(Ci, Ti) //treine-o com o conjto. de treinamento.
      Xi <- evaluateModel(Ci, Si) //avalie-o com o conjto. de testes
      Yi <- {}  //crie o conjunto de instâncias classificadas
      for x in Xi do
         Yi <- Yi + x // adicione instâncias ao conjunto
      end
    end
    for Ci in C do  //para cada classificador
       for Cj in C, Cj ≠ Ci do // adicione a melhor instância de um classificador
no outro
          Tj <- Tj + best(Yi)
          Sj <- Sj - best(Yi)
       end
    end
until true
```

De acordo com o algoritmo 1, no começo da execução o conjunto de dados é extraído de toda a base disponível e organizado da forma mais estruturada possível. Alguns elementos podem ser discretizados de acordo com as regras do modelo

semântico do domínio para melhor atender os objetivos da extração e classificação. A próxima fase consiste em dividir o conjunto em dois subconjuntos: um deles representando o conjunto de treinamento e o outro representando o conjunto de testes. Geralmente a divisão é feita com uma distribuição de 5/95 ou 10/90 (em porcentagem de instâncias). O próximo passo consiste em usar um *pool* de classificadores bem conhecidos.

Neste trabalho são apresentados os resultados da utilização de quatro deles: C4.5 [56] (implementado pelo algoritmo J48) e NaiveBayes, BayesNet e ZeroR. Como já comentado, cada classificador trabalha com seu conjunto de treinamento e testes, inserindo no outro seus melhores resultados. O algoritmo repete o processo indefinidamente.

O funcionamento do algoritmo:

1. O que precisamos é apoiar a Filtragem Colaborativa. No entanto, seu funcionamento depende de avaliações comuns entre usuários e entre itens. No começo de um sistema de recomendação, como já mencionado na seção 2.7.4, há o problema das avaliações serem esparsas devido à pouca quantidade de informação. Nosso cenário seria próximo ao da figura 4.9:

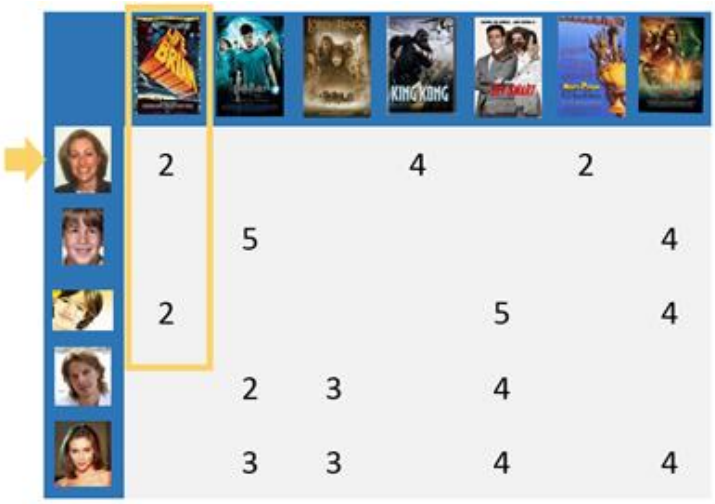


Figura 4.10 Matriz de Avaliação Usuário X Itens no começo de um SR.

Veja que há apenas uma avaliação em comum entre os dois usuários apresentados.

2. Com a abordagem proposta na figura 4.8 e com o algoritmo descrito neste capítulo o que fazemos é gerar mais avaliações para a Filtragem

Colaborativa possa ocorrer. Para isto, usamos o analisador de contexto e combinamos os dados demográficos, criando tuplas, onde a classe é o rating que os usuários deram aos itens dos usuários. Como comentado, o analisador de contexto ainda foi executado manualmente. No momento seu algoritmo compreende a função de combinação descrita na figura 4.7, ou seja, ele cria um vetor de informações demográficas de usuários e itens e considera o *rating* como classe. Futuramente, novas abordagens para o analisador de contexto podem ser testadas (como utilização de informações externas – discutido na conclusão do trabalho).

3. A partir daí, usamos o processos de *bootstrapping* para gerar novas tuplas.
4. Como discutido, o *bootstrapping* sozinho gera *overfiting* e por isto adotamos o *co-training* [55] cujo detalhamento do algoritmo está compreendido no Algoritmo 1 (anteriormente apresentado).

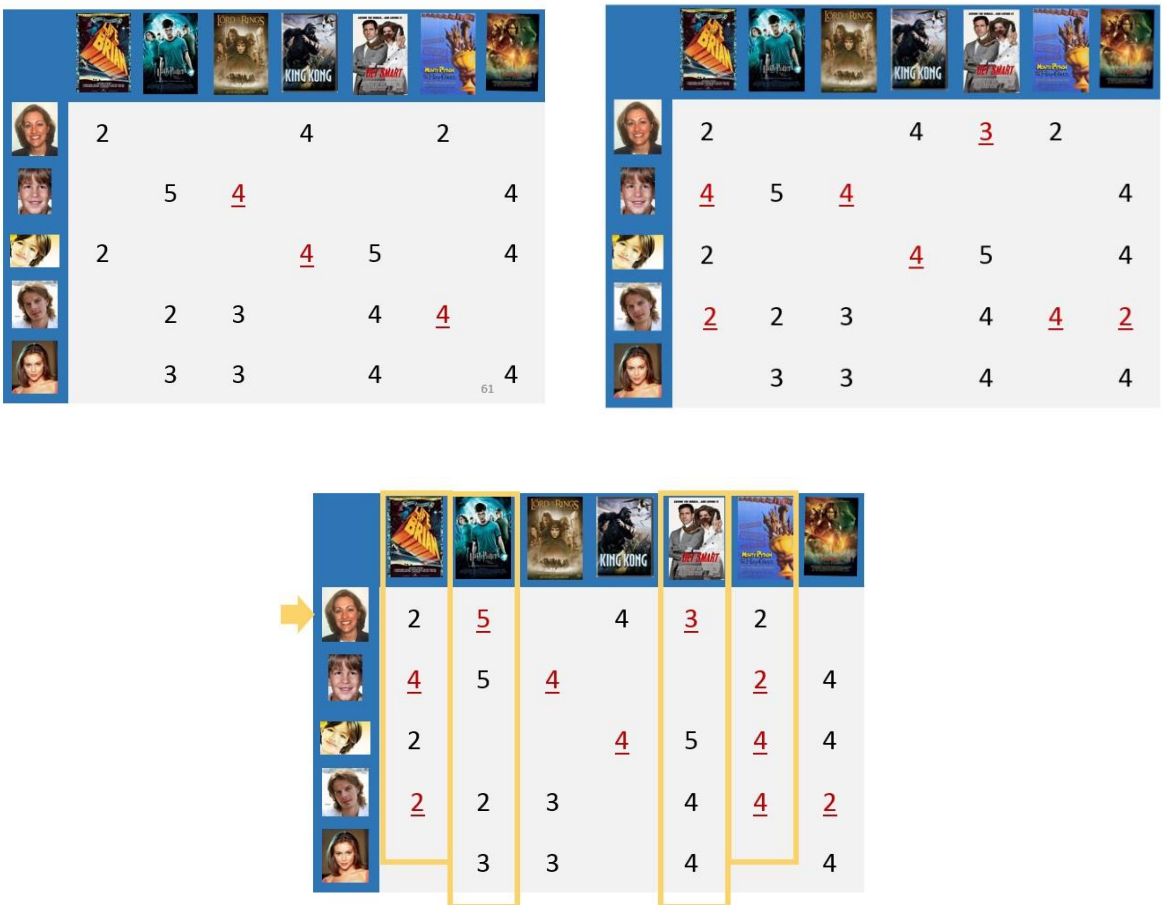


Figura 4.11 Evolução da Matriz durante o Processo.

5. A cada iteração do algoritmo a matriz vai sendo preenchida e permitindo que a Filtragem Colaborativa possa usar mais dados de usuários e itens para seus cálculos.

Para testar esta abordagem nós realizamos alguns experimentos com bases de dados reais. Nos experimentos nós fizemos a validação cruzada de forma a avaliar se o algoritmo conseguia atingir valores próximos dos reais, quando estes valores eram colocados como conjunto de teste. Além disto, usamos a abordagem incremental, reinserindo os dados resultantes (como comentado no Algoritmo 1).

Capítulo 5

EXPERIMENTOS E RESULTADOS

"Perguntaram a dois pedreiros de cantaria o que estavam fazendo. O primeiro disse: "Estou cortando essa pedra em bloco". O segundo respondeu: "Faço parte de uma equipe que está construindo uma catedral".[7]

5.1 Sobre os Experimentos

Para testar a abordagem proposta orientada pelos conceitos do SASF foram realizados experimentos com duas bases de dados diferentes, apresentadas na próxima seção. Os passos para realização dos experimentos foram:

- Passo 1: Construção do Vetor de Informações sobre usuários e itens aplicando a função de combinação de acordo com cada domínio das bases de dados (Tarefa do Analisador de Contexto);
- Passo 2: Aplicação de Algoritmos Tradicionais usando situações com mais e menos dados de treinamento e avaliando o desempenho dos classificadores.
- Passo 3: Aplicação da Abordagem Proposta, iniciando com poucos dados e monitorando seu desempenho a cada iteração (reinserção de dados) – Abordagem do Algoritmo 1, capítulo 4.
- Passo 4: Os dados voltam para a base de dados através da função de decomposição e podem ser usados para recomendação.

[7] Extraído de [85]

5.2 Experimento 1 – Base de Dados Educacional

Para testar a abordagem proposta foram realizados experimentos com uma base de dados que representa a interação de um conjunto de alunos, seus cursos e polos, com dados cedidos pela Secretaria de Educação à Distância SEaD-UFScar. A base de teste contempla um curso com 5 polos e 252 alunos (não foram excluídos alunos que desistiram ou tiveram matrícula cancelada, pois investigava-se o comportamento dos alunos até então). A partir dos dados de acesso dos usuários obtidos pelo *log* do sistema foram analisados estes acessos e referenciados por usuário (através do id do usuário) e por curso acessado (através do id do curso).

Os dados representam os descritores de usuário, curso e acesso, através de vários atributos e o resultado final do mesmo: se foi aprovado ou reprovado no curso em questão. Na tabela 5.1 são descritas as ferramentas monitoradas no log e usadas para compor a informação de perfil do usuário. Os dados consistem em um sumário do acesso do usuário considerando as ferramentas (como fóruns, chats, etc) e sua frequência de acesso. Como resultado final no perfil do usuário tem-se a informação que indica se foi ou não aprovado no curso. Cada elemento na tabela 5.1 representa a ferramenta disponível no sistema e utilizada pelo usuário

Tabela 5.1: Descrição da Base de dados do Experimento

Tipo de Dado	Descrição
Userid	O id do usuário
Courseid	O id do curso que o usuário participou
Assignment	Submissão de Tarefas
Choice	Perguntas e Respostas
Course	Lista de Cursos
Forums	Acesso ao Fórum
Journal	Jornal
Label	Postagens e Atividades para Casa
Resource	Arquivos
Upload	Upload de Arquivos
User	Informações do Usuário
Days	Qtd. de dias de Acesso
Avg_days	Média de Acessos por Dia
Result	Informação sobre Aprovação/Reprovação

A tabela 5.2 mostra alguns exemplos de perfis (*profiles/user model*) dos usuários/alunos. No experimento foi comparada a porcentagem de acerto de um classificador usando os dados reais nas porcentagens indicadas (10%, 20%, 30% e 40% do conjunto de todos os dados) como conjunto de treinamento e a abordagem aqui proposta partindo-se de apenas 10% da base de dados e fazendo a reinserção de acordo com os treinamentos dos algoritmos. Nos gráficos apresentados nas figuras 5.1 e 5.2 pode-se verificar que as situações são bem similares e isto indica que é possível incrementar a base de dados com algum grau de confiança usando esta abordagem.

Tabela 5.2 Exemplo do Conjunto de Dados.

Userid	Courseid	Assignment	Forum	…	Days	Avg_days	Result
115	5	Yes	No	…	30	5	Yes
113	5	No	Yes	…	40	4	No
98	1	No	Yes	…	10	2	Yes
…	…	…	…	…			…

Na figura 5.1 é apresentada a comparação entre o algoritmo J48 e a abordagem deste trabalho usando o J48 acoplado. Neste experimento os resultados se mantiveram próximos até o equivalente a 60% da base dados.

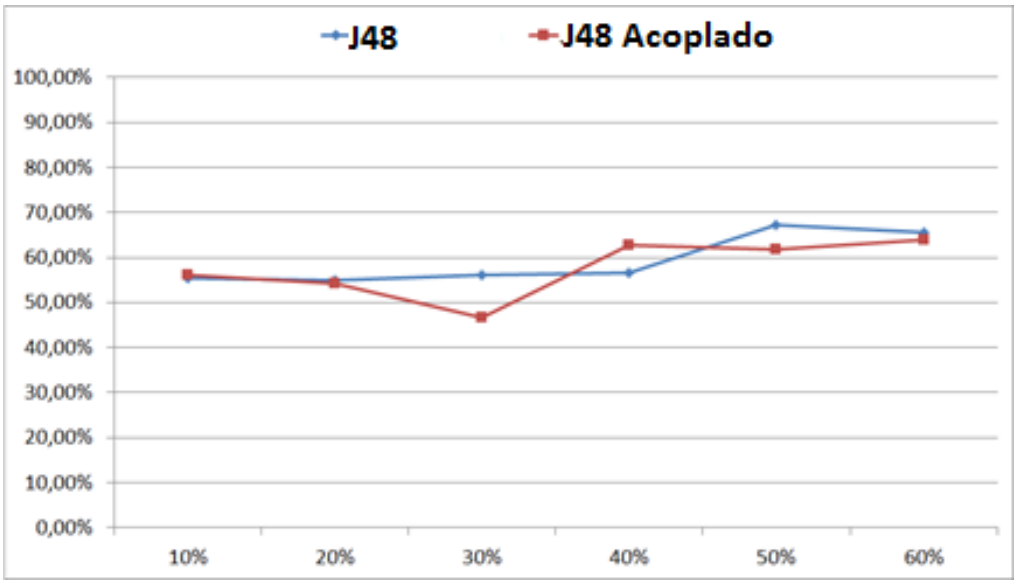


Figura 5.1 Comparação do Desempenho (eixo Y) entre um algoritmo que usa a base de dados original (J48) e o algoritmo acoplado (J48 + NaiveBayes). O eixo X representa o tamanho do conjunto de treinamento em relação ao tamanho total da base original.

Inicialmente foram executados experimentos com a base de dados dividindo-a em conjunto de treinamento e conjunto de testes e aplicando algoritmos tradicionais de aprendizado. Neste caso, o C4.5 (cuja implementação no Weka é denominada J48). A divisão começou na proporção de 10% da base para treinamento e 90% para testes (validação do treinamento). E caminhou seguindo novas proporções de treinamento e testes (20/80, 30/70, 40/60 até 60/40). Isto foi feito para verificar que mais dados usados para treinamento permitiriam melhor desempenho do classificador (o que de fato ocorreu até 60% da base, a partir daí não houve ganho significativo). O gráfico aponta então, por exemplo, que com 10% da base (total) sendo usada como conjunto de treinamento a porcentagem de acerto no conjunto de testes (os outros 90%) foi pouco mais de 55%.

Já o segundo experimento (rodado com a mesma base e nas mesmas condições, exceto o tamanho do conjunto de treinamento – explicado a seguir) considerou a abordagem proposta (acoplamento de classificadores) e foi iniciado com apenas 10% da base de dados sendo considerada como conjunto de treinamento. Os outros 90% foram considerados dados não rotulados. A partir do treinamento inicial, a cada iteração os dados não rotulados eram classificados e organizados de acordo com um ranking. Este ranking é obtido através da probabilidade que o classificador estimou determinado rótulo para a instância em questão (por exemplo, o classificador pode classificar a instância como *Result = Yes* com 0,8 de certeza). É feita uma ordenação em relação às melhores probabilidades, ou seja, aquelas instâncias onde o classificador (cada um deles) "têm mais certeza sobre seu resultado". Cabe ressaltar que esta probabilidade é fornecida pela implementação do Weka[8], cujos algoritmos tradicionais de aprendizado de máquina foram usados neste trabalho na sua forma de API´s, e o método que fornece tais distribuições é chamado de *distributionForInstance*[9].

As instâncias no topo do ranking eram reinseridos nos classificadores e uma nova iteração começava. Foram testadas situações com reinserção de grupos maiores e menores de novas instâncias (por exemplo as 100 melhores ou as 2 melhores). Inserir mais instâncias diminui o tempo de processamento do algoritmo

[8] Disponível em http://www.cs.waikato.ac.nz/ml/weka/

[9] http://weka.sourceforge.net/doc.dev/weka/classifiers/Classifier.html#distributionForInstance(weka.core.Instance)

(para alcançar tamanhos maiores de treinamento), mas faz com que o desvio semântico surja mais rapidamente, ou seja, instâncias rotuladas incorretamente entram no conjunto de treinamento mais rapidamente.

Estes testes foram repetidos dez vezes e a amostra escolhida era aleatória. Os resultados apresentam as médias dos resultados obtidos.

O que é possível observar pelos resultados obtidos é que o processo de *Bootstrapping* com o acoplamento aproxima os resultados da base real até, aproximadamente, 60% do conjunto total de dados. A partir daí o desvio semântico faz com que o desempenho caia bastante (o desvio semântico esperado começa a acontecer). Isto significa dizer que o sistema de recomendação com dados treinados e reinseridos, partindo de 10% do tamanho da base, conseguiria um desempenho semelhante a um conjunto de dados reais com 60% da base testada.

Este experimento foi repetido, comparando-o também com o algoritmo *NaiveBayes* tradicional e proporções de treinamento e teste em 10/90, 20/80, 30/70 e 40/60 da base de dados.

De maneira similar à anterior, primeiro foi testada a base original dividindo-a em conjuntos de treinamento e testes usando as proporções citadas. Após isto, foi testada a abordagem proposta com um conjunto inicial de 10% no conjunto de treinamento e o restante da base (90%) foi considerada como não rotulada e usada nas iterações com classificação, ordenação das classificações (*ranking*) e reinserção. A partir da reinserção das melhores instâncias classificadas o conjunto de treinamento foi aumentando, ou seja, eram gerados novos rótulos artificialmente. O desempenho do algoritmo foi comparado até 40% da base.

Como é possível observar na figura 5.1, o eixo X mostra o tamanho do conjunto de treinamento e o eixo Y mostra o percentual de acerto, considerando o restante dos dados. Os resultados obtidos mostram que o comportamento do algoritmo na abordagem proposta é similar ao uso dos dados originais da amostra. Isto quer dizer que, partindo de apenas 10% da base de dados original o algoritmo foi capaz de incrementá-la a até o equivalente a 40%, usando o aprendizado acoplado e a reinserção das instâncias.

Este resultado permite que o sistema possa gerar recomendações, aumentando sua matriz de dados, e aguardar que os valores reais dos usuários possam ser inseridos, ou seja, o desvio semântico do erro na predição é atrasado. Este resultado colabora com a recomendação e com o tratamento do problema do

cold-start na medida em que fornece alternativa inicial para o problema de poucos dados.

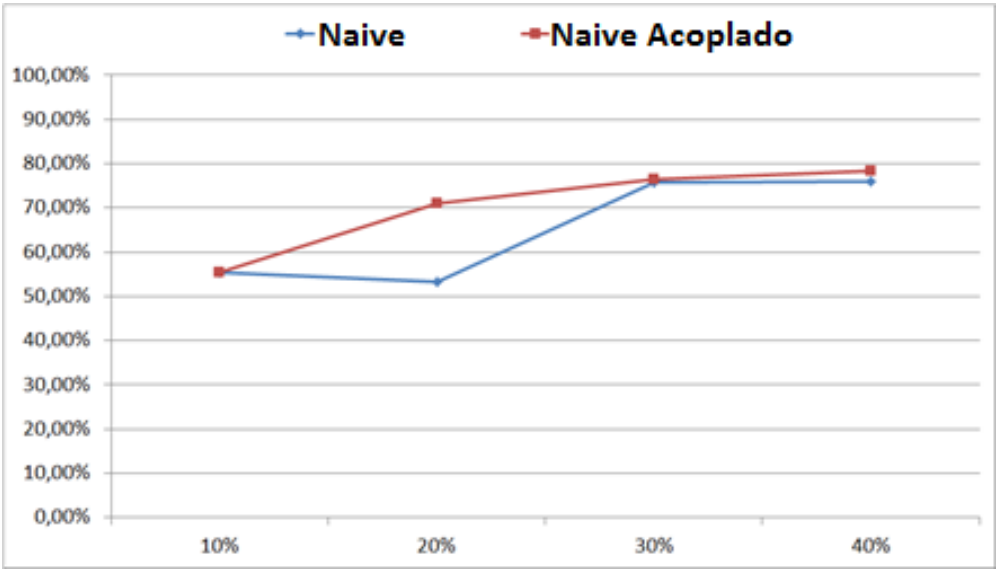


Figura 5.2. Comparação do Desempenho (eixo Y, % de acerto) entre um algoritmo que usa a base de dados original (NaiveBayes) e o algoritmo acoplado (NaiveBayes + NaiveBayes).

Através destes resultados podem ser criadas recomendações que indiquem preferências do aluno que obtém melhores resultados, uma vez que é possível melhorar o acerto sobre o desempenho do aluno com base em seu comportamento. Os trabalhos futuros incluem a integração dos resultados obtidos com os demais módulos (como automatizar a análise de contexto e relacionar os dados ao gerador de grupos bem como ao analisador de feedaback) e a oferta de outros tipos de recomendação, além do desempenho do aluno em si.

Pela análise dos resultados observados nos experimentos é possível concluir que o uso do acoplamento para gerar recomendações sobre o desempenho do aluno incrementa a base inicial de dados (conjunto de treinamento) e permite a oferta de recomendações com dados mais próximos de uma base de dados real. O *bootstrapping* sozinho não conseguia obter os mesmos resultados e por isto a abordagem de algoritmos acoplados foi utilizada. Futuramente será possível comparar a abordagem com outras técnicas de recomendação e integrá-la ao restante desse trabalho, sendo uma das geradoras de recomendações candidatas no sistema que ainda contará com um módulo para decidir a oferta das recomendações.

É importante citar que um princípio do Aprendizado Sem Fim é aprender sem parada. Assim, os resultados experimentais mostram que há um limiar onde o desvio semântico começa a acontecer. Mas também mostra que é possível melhorar o

desempenho da recomendação quando existem poucos dados. Como critério de autocorreção foi pensada a própria inserção dos dados reais do usuário na base, ou seja, a chegada de instâncias já rotuladas. Isto foi simulado no experimento 2.

5.3 Experimento 2 – Base de Dados de Filmes

Para poder analisar os resultados obtidos tendo em vista uma base com mais instâncias e um contexto diferente do experimento 1, foi usada a base de dados de experimentos do *Movielens* com 100 mil instâncias.

O conjunto de dados do MovieLens é disponibilizado publicamente[10] e possui dados sobre filmes, usuários e suas avaliações sobre os filmes. Trata-se de um conjunto de dados utilizado por vários pesquisadores.

Tabela 5.3: Descrição da Base de dados do Experimento

Tipo de Dado	Tabela (BD)	Descrição
Userid	**Tbl_User**	ID do usuário
Age		Idade (Número)
Gender		Sexo (M/F)
Occupation		Ocupação (String com um conjunto padrão de ocupações)
ZipCode		CEP
Movieid	**Tbl_Movies**	ID do Filme
Movietitle		Título do Filme (String)
Releasedate		Data de lançamento no cinema
Videoreleasedate		Data de lançamento em video
IMDbUrl		Link IMD
Action		É ou não é deste gênero. Valor 0 ou 1 (binário)
Adventure		É ou não é deste gênero. Valor 0 ou 1 (binário)
…		São vários gêneros.
Western		É ou não é deste gênero. Valor 0 ou 1 (binário)
Userid	**Tbl_Ratings**	ID do usuário que classificou o filme
Itemid		ID do filme
Rating		Valor dado (1 a 5 – classe)
Timestap		Data

[10] http://grouplens.org/datasets/movielens/

Os dados sobre os usuários são formados por *userid, age, gender, occupation,* e zipcode. Os dados sobre filmes são *movieid, movietile, releasedate, videoreleasedate, IMDbUrl,* e um conjunto de descritores sobre a classificação do filmes (são binários, ou o filme é ou não é classificado deste tipo) que incluem *unknown, Action, Adventure, Aninmation, Children´s, Comedy, Crime, Documentary, Fantasy, Film-Noir, Horror, Muscial, Mystery, Romance, Sci-Fi, Thriller, War, Western.* Os dados das avaliações (*ratings*) dos usuários incluem: *userid, itemid, rating* e *timestamp*.

Dividimos a tarefa em dois experimentos diferentes apresentados a seguir em conjunto com seus resultados para posterior discussão e análise.

O primeiro experimento analisa o comportamento de quatro classificadores:

- C4.5 (Implementação J48) [57];
- Naive Bayes;
- ZeroR;
- BayesNet;

Todas as implementações foram feitas usando-se das API´s do *Weka*.

Os quatro classificadores foram analisados separadamente e com validação cruzada. O experimento rodou com 10% das instâncias como conjunto de treinamento e os 90% restantes foram usados para os testes. O mesmo conjunto foi usado com cada classificador.

Na figura 5.3 vemos a abordagem deste experimento que trata de um caso tradicional num Sistema de recomendação: uso de classificadores para prever a preferência de perfis em relação aos filmes disponíveis. Assim, este experimento serve para que tenhamos evidências sobre qual seria o comportamento de sistemas de recomendação tradicionais quando aplicados a esta base de dados. Desta forma, poderemos ter base de comparação para analisar se a arquitetura proposta traz ganhos, se atinge os objetivos e se é viável em aplicações reais com grande volume de dados.

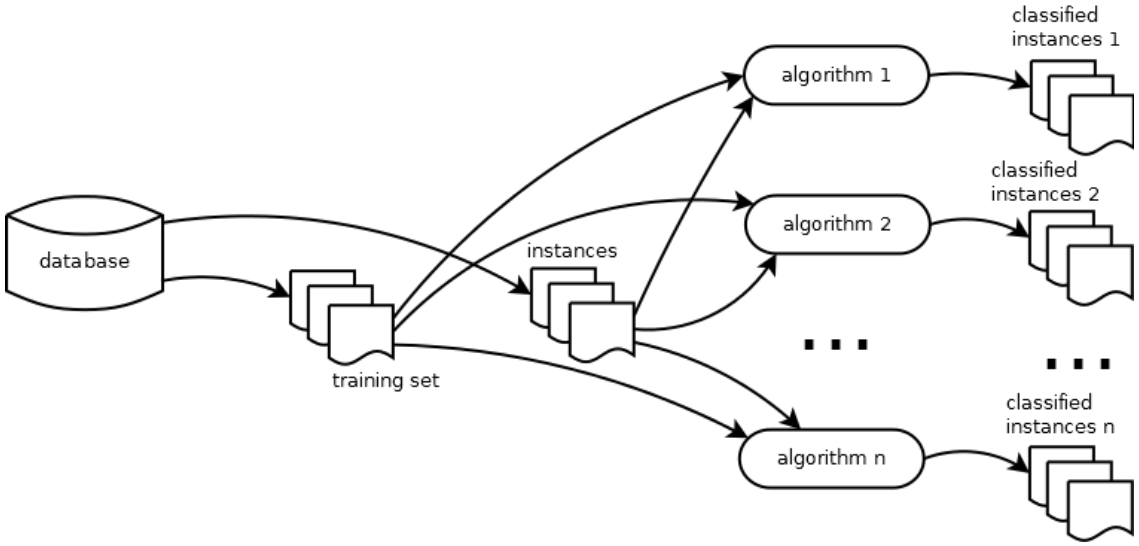


Figura 5.3 Experimento com Algoritmos Tradicionais

Foi observado na aplicação deste experimento que a taxa de acerto dos 4 algoritmos com aproximadamente 10.000 instâncias no conjunto de treinamento foi pouco mais de 80%. A partir desta quantidade de instâncias o aumento do tamanho do conjunto de treinamento não trazia ganhos significativos ao desempenho do classificador.

Tabela 5.3 Acertos no Experimento com a Base do Movielens e Algoritmos Tradicionais

Considerando um conjunto de treinamento de 10% da Base de Dados			
J48	Naive Bayes	ZeroR	BayesNet
0,81%	0,82	0,83	0,82

No segundo experimento foram usados os mesmos quatro classificadores anteriores, combinados dois a dois e um conjunto inicial de treinamento com apenas 100 instâncias. Mas, é importante lembrar que é possível a utilização de mais de dois classificadores. As abordagens usando três classificadores não mostraram melhora significativa e a abordagem usando os quatro mostrou o mesmo desempenho da melhor abordagem (C4.5 + BayesNet).

Como mostrado na figura 4.8 a arquitetura proposta usa a combinação de vários classificadores. Um classificador atua como supervisor dos resultados do outro, pois cada um reaproveita as melhores instâncias do outro para continuar seu conjunto de treinamento. E, como visto na figura 5.4, a classificação final de uma instância é obtida pela votação entre os classificadores (os seus resultados independentes sobre as instâncias).

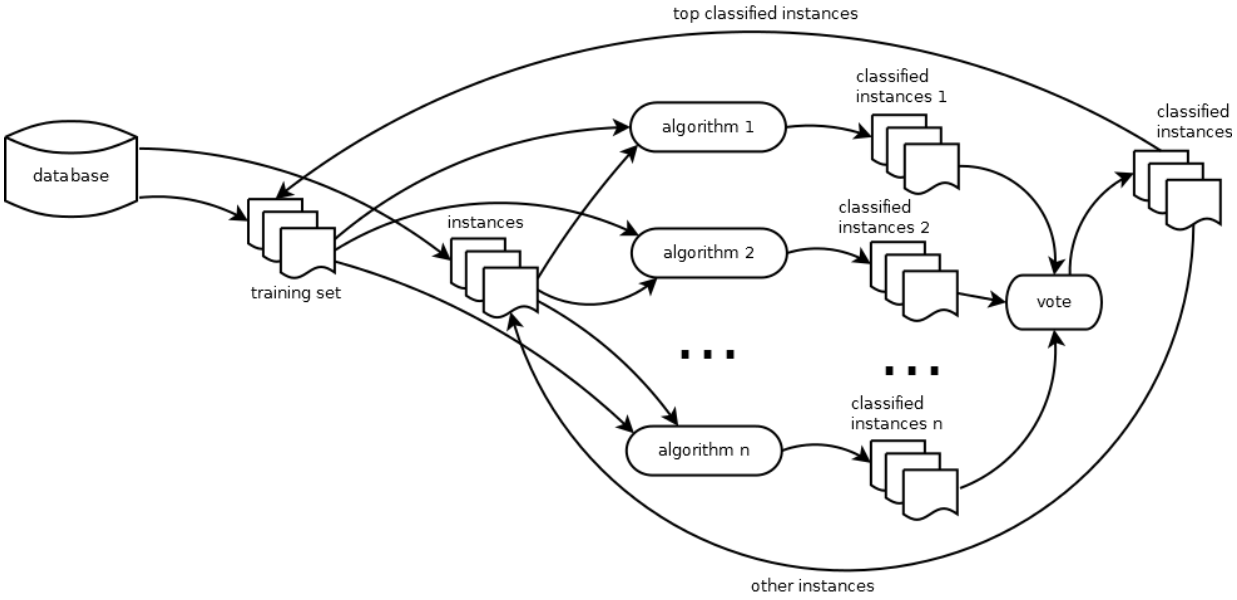


Figura 5.4 Experimento com a Arquitetura Proposta

Como é possível observar pela figura 5.5 o algoritmo que combina outros algoritmos de classificação começa com uma taxa de acerto baixa em relação à abordagem tradicional, mas consegue igualar esta taxa e mantê-la por várias execuções. Cabe lembrar que este algoritmo inicia com um pequeno conjunto de dados rotulados que são usados como base de treinamento e vai incrementando sua própria base usando seus próprios resultados e sua autossupervisão para gerar mais dados rotulados.

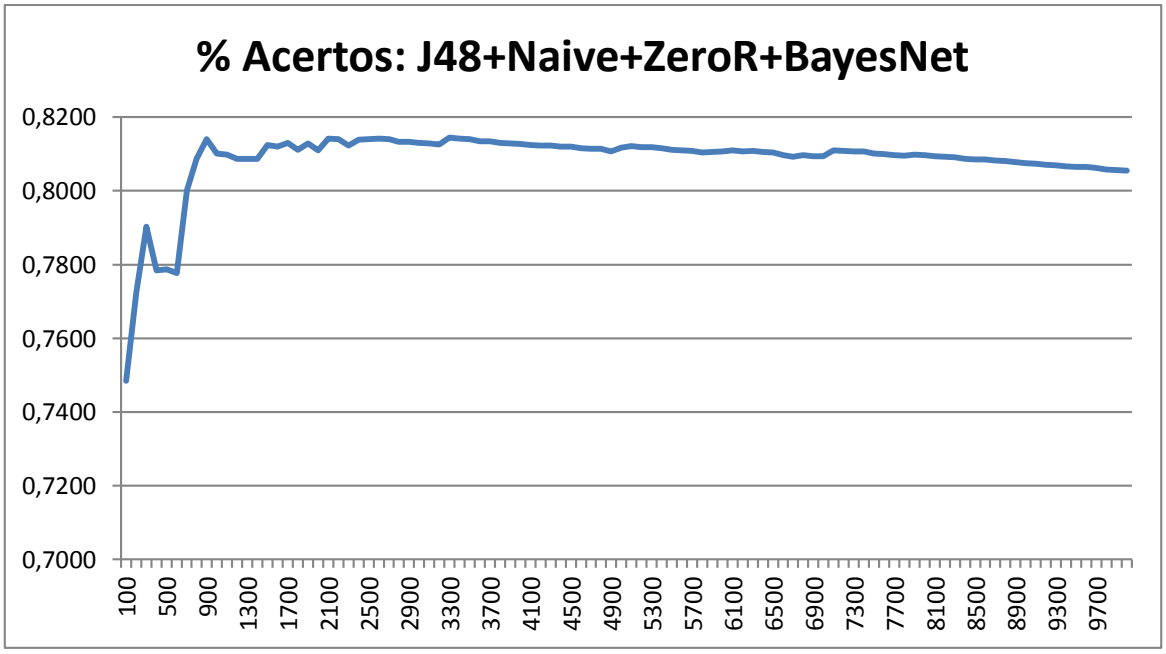


Figura 5.5 Acertos no Experimento do MovieLens com a Arquitetura Proposta

Para efeito de comparação foram testados também os 4 classificadores separadamente com a abordagem de *BootStrapping*. O resultado mostrado na figura 5.6 aponta que sozinhos os algoritmos acabam gerando muitos resultados indesejados a partir de certo ponto.

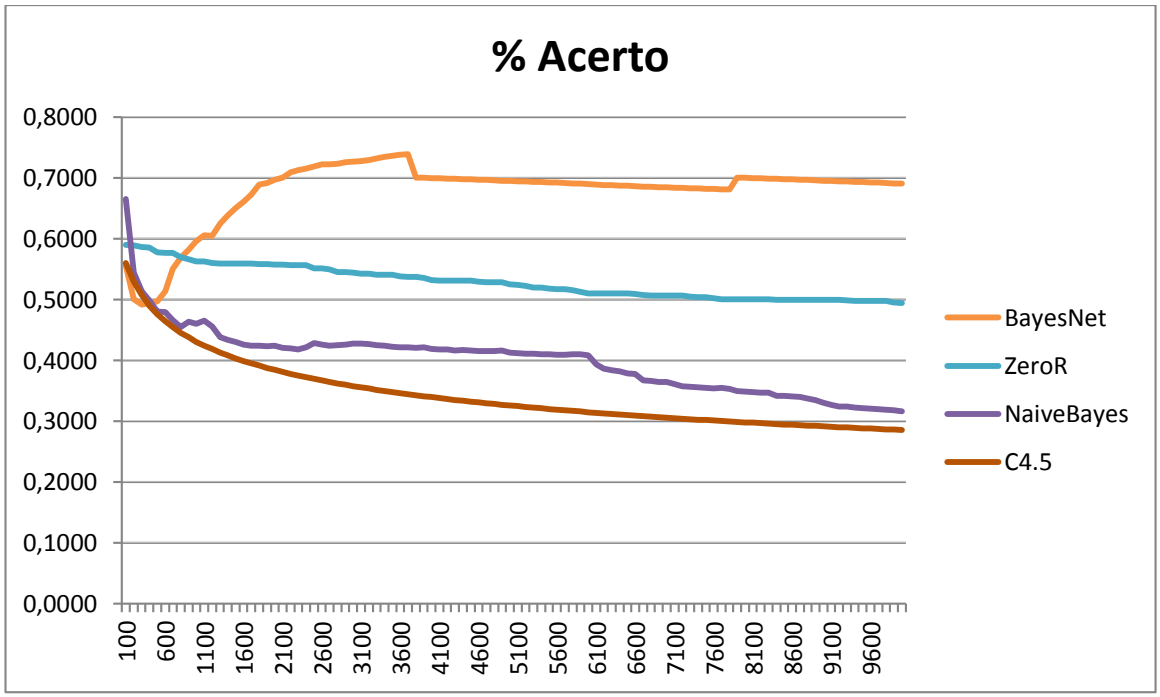


Figura 5.6 Acertos no Experimento do MovieLens com Algoritmos Sozinhos

A melhor combinação dos resultados dos classificadores é obtida através do cálculo de uma matriz de independência entre eles. Durante o experimento foram testadas abordagens com apenas 2 classificadores (combinando os 4 utilizados).

A matriz de independência é calculada considerando os erros dos classificadores. O critério adotado é que eles errem em instâncias diferentes, ou seja, o par de classificadores com maior quantidade de erros em instâncias diferentes foi considerado mais independente. Esta independência é usada para composição do par. Este cálculo é feito apenas uma vez, com uma pequena amostra (20% do conjunto inicial) já rotulada da base de dados, dividida em conjunto de treinamento (cada classificador recebia o mesmo conjunto) e teste (cada classificador aplicava a classificação no mesmo conjunto de testes) comparando para cada instância (é possível obter um id de instância no próprio weka) se os classificadores A e B acertam ou erram. Onde A acerta e B erra e onde B erra e A acerta é considerada independência e pontuada na matriz.

O algoritmo abaixo expressa o cálculo da matriz de independência:

Algoritmo 2: Matriz de Independência

```
Input: I – Conjunto de Dados Rotulados
parse(I, Tr, Tes) // o conjunto de dados é dividido em treinamento e testes
A = setClassifier("nome_classificador") // conjunto de treinamento carregado em A
B = setClassifier("nome_classificador") // conjunto de treinamento carregado em B
trainClassifier(A, Tr) //treine A com o conjto. de treinamento
trainClassifier(B, Tr) //treine B com o conjto. de treinamento
Xi <- evaluateModel(A, Tes) //avalie A com o conjto. de testes
Yi <- evaluateModel(B, Tes) //avalie B com o conjto. de testes

for each id in Xi && Yi do // faça iteração para todos as instâncias
   if getResult(getInstance(Xi, id)) != getResult(getInstance(Yi, id))
   matInd++     // se os classificadores discordarem incrementa independência
end
matInd = matInd / size(I) // Normaliza entre 0 e 1
```

O intuito disto é gerar e checar a matriz de independência entre eles, a fim de iniciar testes com a autorreflexão. O objetivo é trabalhar com classificadores com erros independentes, ou seja, erros em diferentes instâncias e agregar valor ao algoritmo de votação. Posteriormente, em trabalhos futuros, será possível usar esta matriz dinamicamente para trocar os algoritmos de classificação em tempo de execução.

Tabela 5.4 Cálculo da Independência dos Classificadores através de suas divergências em erros de classificação

	ZeroR	**BayesNet**	**C4.5**	**NaiveBayes**
ZeroR	-	0.22	0.21	0.20
BayesNet	-	-	0.32	0.25
C4.5	-	-	-	0.28
NaiveBayes	-	-	-	-

Para simular a entrada de dados pelo usuário no segundo experimento foi realizada uma abordagem de fluxos onde a cada ciclo maior um conjunto de dados rotulados foi adicionado à base de treinamento. Se os dados novos trouxessem rótulos divergentes daqueles obtidos então foi feita a substituição, ou seja, dados rotulados prevalecem sobre rótulos obtidos nos treinamentos. Os ciclos maiores ocorreram a cada 1000 instâncias novas no conjunto de treinamento (quando o conjunto chegava a 1000 instâncias um fluxo de 1000 era adicionado à base; este processo foi repetido até o 10% do tamanho total do conjunto de treinamento, ou seja, próximo de 10.000

instâncias no conjunto). O resultado mostrado na figura 5.7 é próximo do que foi obtido sem os fluxos (figura 5.5), mostrando que poucas foram as substituições efetuadas.

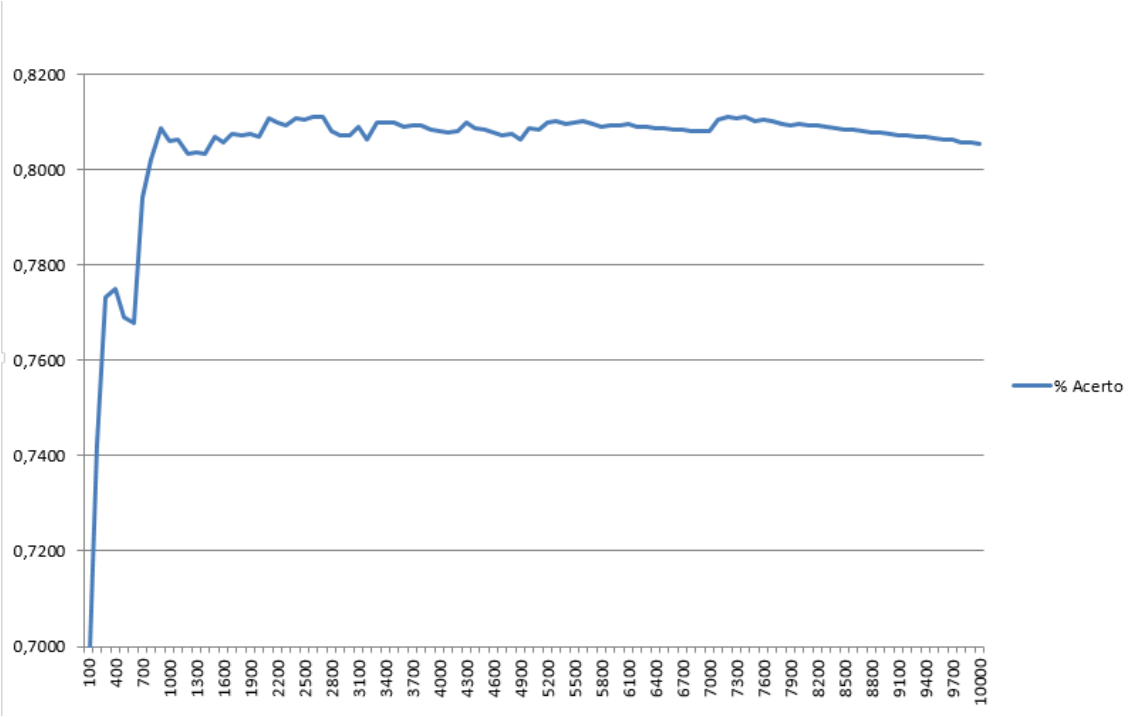


Figura 5.7 Acertos no Experimento do MovieLens com a Arquitetura Proposta e com a simulação de entrada de novos dados através de fluxos

Capítulo 6

CONCLUSÕES E TRABALHOS FUTUROS

"Nem todo ponto final indica fim de história, pode ser só o começo de um novo parágrafo."
(Autores Diversos!)

O volume de informações gerado pelas pessoas é cada vez maior. As pessoas possuem interesse em gerar informações sobre artefatos, em recomendar e receber recomendações. No entanto, um sistema de recomendação relaciona-se a um domínio específico que pode, inicialmente, não ter muitos dados disponíveis. Para tanto é comum valer-se de conhecimentos de especialistas de domínio para geração de recomendações no início do ciclo do vida do sistema.

Este trabalho, como já citado, tem o intuito de dar os primeiros passos na direção de uma arquitetura SASF para um Sistema de Recomendação e por isto guia-se pelos conceitos de um SASF. Com os experimentos foi possível observar consistência nesta direção, mas deficiências a serem supridas e trabalhos futuros a serem realizados. É importante ressaltar que além dos resultados empíricos mostrados também foi feita a proposta de conversão do cálculo matricial da Filtragem Colaborativa para um formato que aproveite os dados que caracterizam usuários e itens e permite que o aprendizado possa ser feito por classificadores. O principal problema encontrado foi estabelecer o critério de parada dos algoritmos. O desvio semântico ocorre, mas foi perceptível nos experimentos e tratado com a simulação de dados de entrada dos usuários. No entanto, é importante buscar formas alternativas de validar esta informação e detectar o desvio semântico além do especialista humano ou dados externos do usuário (avaliações).

O sistema também mostra características relevantes como o aprendizado ao longo do tempo, considerando que novas opiniões possam surgir e que o usuário pode mudar de opinião. Itens que podem não ser observados se a recomendação basear-se num modelo previamente treinado apenas.

A abordagem proposta mostra resultados importantes com os componentes implementados e testados e com poucos dados iniciais. No entanto, trabalhos futuros deverão tratar situações onde existem muitos dados e o sistema deva contribuir com abordagens tradicionais já existentes.

A filtragem colaborativa é a técnica que gera os melhores resultados em relação à todas as necessidades de um sistema de recomendação. No entanto, para que ela ocorra, são necessários dados de avaliações de usuários. Tentar gerar recomendações sem tais dados gerará um desvio semântico, ou seja, a recomendação não será bem sucedida.

Para auxiliar problemas como este foi proposto nesta pesquisa uma arquitetura cujo objetivo principal é adiar este desvio semântico. Assim, busca-se auxiliar o sistema a gerar recomendações enquanto o usuário realmente insere suas avaliações.

Os testes realizados mostram que, até certo tamanho da base de dados (cerca de 10%), é possível aproximar os resultados obtidos de uma base real (aquele onde os usuários realmente avaliaram itens) e de uma base promovida pelo processo proposto.

Infelizmente este processo não consegue estimar todas as avaliações. Ele possui, ainda, limitações do próprio processo de *self-learning*, o *bootstrapping,* que insere inferências que causarão desvio de conceito. Apesar disto, como o sistema é dinâmico, novas avaliações chegam constantemente e substituem aquelas que o sistema inferiu (caso sejam divergentes), pois a opinião do usuário é o que realmente deve ser considerado na recomendação.

O desempenho da predição de uma avaliação num sistema de recomendação é de suma importância. Mas, tão importante quanto prever corretamente os interesses do usuário é não prever de maneira totalmente incoerente, ou seja, não ter um desvio semântico tão alto que faça o usuário pensar que não há mecanismo inteligente recomendando algo a ele.

Sendo assim, a proposta mostrou bons desempenhos nos experimentos, pois seus resultados foram próximos ao de uma base de dados real, ou seja, o aprendizado

gerado pelo processo iterativo se aproximou daquele que seria obtido por um processo tradicional com mais dados. Isto, considerando a porcentagem de acertos nas tuplas geradas pelo processo apresentado no trabalho.

Pela análise dos resultados observados no experimento 1 é possível concluir que o uso do acoplamento para gerar recomendações sobre o desempenho do aluno incrementa a base inicial de dados (conjunto de treinamento) e permite a oferta de recomendações com dados mais próximos de uma base de dados real. O *bootstrapping* sozinho não conseguia obter os mesmos resultados e por isto a abordagem de algoritmos acoplados foi utilizada.

Em relação ao experimento 2, com uma base de dados de filmes, os resultados também foram muito próximos. Iniciar o treinamento com poucos dados permitiu ao sistema alcançar o desempenho da base com quase 10% dos dados reais.

O principal trabalho futuro a ser comentado é a busca para tornar a arquitetura proposta completa pela utilização de todos os princípios do Aprendizado Sem-Fim. Para este trabalho, serão necessários alguns passos, dentre eles:

- Tornar o processo de análise de contexto automático: permitir que a função de composição que agrega informações de usuários e itens possua um processo automático para o restante do processo e também possa fluir sem intervenção do usuário;
- Criação de novos acoplamentos e mudança de algoritmos de classificação em tempo de execução: como visto, são usados múltiplos algoritmos que colaboram entre si. É importante considerar a possibilidade de mudar estes algoritmos em função de seu desempenho nesta colaboração;
- Exploração das amostras negativas: em sistemas de recomendação a função de utilidade responde a uma pergunta de qual item seria mais interessante ao usuário. Um trabalho futuro nesta linha seria buscar informações de itens desinteressantes e inseri-las nos treinamentos dos classificadores.
- Trabalhar a validação dos dados obtidos usando técnicas como o *Conversing Learning* [58] que usa técnicas de *Active Learning* e Interação Social (com perguntas e respostas) para auxiliar nos critérios de parada dos classificadores a afim de adiar ainda mais o desvio semântico e não necessitar tanto de intervenção humana. Isto seria

possível se dados externos puderem avaliar quando os classificadores começam a causar desvios semânticos e aguardar a inserção dos dados rotulados pelo usuário.

- Também seria interessante trabalhar numa forma de inserir dados rotulados através de outro sistema (como uma sistema de regras, uma base de conhecimento ou um *knowledge graph*) e não apenas depender da rotulação de dados do usuário.

O componente de geração de grupos não trouxe ganhos efetivos nos experimentos, mas continuará sendo explorado como trabalho futuro. O que pode ter ocorrido é que os grupos obtidos aproximam-se muito das classes e por isto os resultados não adicionaram ganhos.

REFERÊNCIAS

1. KOBSA, A.; KOENEMANN, J.; POHL, W. Personalised hypermedia presentation techniques for improving online customer relationships. **The Knowledge Engineering Review**, 2001. 111-155.

2. DUARTE, M. C.; HRUSCHKA-JR., E. R.; NICOLETTI, M. D. C. **Minimização do Impacto do Problema de Desvio de Conceito por Meio de Acoplamento em Ambiente de Aprendizado Sem Fim**. The 8th Brazilian Sympsium in Information and Human Language Technology - STIL 2011. Mato Grosso, Brazil.: [s.n.]. 2011.

3. AMARAL, L. R. **Aplicando Princípios do Aprendizado de Máquina Sem-Fim na Construção de um Biocurador Automático para o Gene Ontology (GO)**. São Carlos: [s.n.], 2013.

4. SHARDANAND, U.; MAES, P. **Social Information Filtering:** Algorithms for Automating "Word of Mouth". CHI '95 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. New York: ACM Press/Addison-Wesley Publishing Co. 1995. p. 210-217.

5. ADOMAVICIUS, G.; TUZHILIN, A. Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. **Knowledge and Data Engineering, IEEE Transactions on**, 17, Junho 2005. 734-749.

6. GHAZANFAR, M. A. **Robust, Scalable, and Practical Algorithms for Recommender Systems**. [S.l.]: Thesis for the degree of Doctor of Philosophy, 2012.

7. PARK, Y.-J.; TUZHILIN, A. **The long tail of recommender systems and how to leverage it.** Proceedings of the 2008 ACM conference on Recommender systems, RecSys. Nova Iorque: ACM. 2008. p. 11-18.

8. ABBASSI, Z. et al. **Getting recommender systems to think outside the box**. Proceedings of the third ACM conference on Recommender systems. Nova Iorque: ACM. 2009. p. 285-288.

9. RICCI, F.; ROKACH, L.; SHAPIRA, B. Introduction to Recommender Systems. In: RICCI, F., et al. **Recommender Systems Handbook**. [S.l.]: Springer, 2010. p. 1-38.

10. GOTARDO, R. A. Modelo I2P: Recomendação de Recursos baseando-se em Preferências, Interesses e Popularidade. **Dissertação de Mestrado**, São Carlos, 2008.

11. DE-BRA, P. **Making General-Purpose Adaptive Hypermedia Work**. WebNet. [S.l.]: [s.n.]. 2000. p. 117-123.

12. RESNICK, P.; VARIAN, H. R. Recommender Systems. **Communications of the ACM**, 40, March 1997. 55-58.

13. REATEGUI, E. B. et al. Personalização de Páginas Web através dos Sistemas de Recomendação. In: _____ **Tópico em Sistemas Interativos e Colaborativos**. São Carlos: [s.n.], 2006.

14. GOLDBERG, D. et al. Using collaborative filtering to weave an information Tapestry. **Communications of the ACM**, Dezembro 1992. 61-70.

15. SCHAFER, J. B.; KONSTAN, J. A.; RIEDL, J. E-Commerce Recommendation Applications. **Data Min. Knowl. Discov**, 5, 2001. 115-153.

16. BELVIN , N. J.; CROFT, W. B. Information Filtering and Information Retrieval: two sides of the same coin? **Communications of the ACM**, 35, December 1992. 29.

17. FOLTZ, P. W.; DUMAIS, S. T. Personalized information delivery: an analysis of information filtering methods. **Communications of ACM**, 35, 1992. 51-60.

18. HERLOCKER, J. L. **Understanding and improving automated collaborative filtering systems**. University of Minnesota: [s.n.], 2000.

19. MAIDEL, V. et al. **Evaluation of an ontology-content based based filtering method for a personalized newspaper**. Proceedings of the 2008 ACM conference on Recommender systems, RecSys '08. Nova Iorque, EUA: ACM. 2008. p. 91-98.

20. WENG, S.-S.; CHANG, H.-L. Using ontology network analysis for research document recommendation. **Expert Syst. Appl.**, 2008.

21. VENSON, E. **Um Modelo de Sistema de Recomendação Baseado em Filtragem Colaborativa e Correlação de Itens para Personalização no Comércio Eletrônico (Mestrado)**. [S.l.]: Sistemas de Computação, Universidade Federal de Santa Catarina - UFSC, 2002.

22. TORRES, R. **Personalização na Internet**. [S.l.]: Novatec, 2004.

23. CHRISTAKOU, C.; STAFYLOPATIS, A. **A hybrid movie recommender system based on neural networks**. proceedings of the 5th international conference on intelligent systems design and applications. Wroclaw, Poland: [s.n.]. 2005. p. 500-505.

24. KIM, B. M.; LI, Q. **Probabilistic Model Estimation for Collaborative Filtering Based on Itens Attributes**. International Conference on Web Intelligence. [S.l.]: [s.n.]. 2004.

25. HAN, S. E.-H.; KARYPIS, G. **Feature-based Recommendation System**. CIKM '05: Proceedings of the 14th ACM International Conference on Information and Knowledge Management. New York: [s.n.]. 2005. p. 446-452.

26. KARYPIS, G. **Evaluation of item-based top-n recommendation algorithms**. Proceedings Of The Acm Conference On Information And Knowledge Management. Atlanta, Georgia, USA: [s.n.]. 2001.

27. TIRAWEERAKHAJOHN, C.; PINNGERN, O. **Finding item neighbors in itembased collaborative filtering by adding item content**. proceedings of the 8th international conference on control, automation, robotics and vision. Kunming, China: [s.n.]. 2004. p. 1674-1678.

28. LIU, J. et al. **An optimized collaborative filtering approach combining with item-based prediction**. proceedings of the 11th international conference on computer supported cooperative work in design. Melbourne, Australia: [s.n.]. 2007. p. 157-161.

29. SARWAR, B. et al. **Incremental singular value decomposition algorithms for highly scalable recommender systems**. Proceedings of the 5th International Conference in Computers and Information Technology. [S.l.]: Citeseer. 2002. p. 27-28.

30. XUE, G.-R. et al. **Scalable collaborative filtering using cluster-based smoothing**. Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR. Nova Iorque: ACM. 2005. p. 114-121.

31. BURKE, R. Hybrid recommender systems: Survey and experiments. **User Modeling and User-Adapted Interaction**, Novembro 2002. 331-370.

32. MENDEL, J. M.; MCLARE, R. W. Reinforcement-learning control and pattern recognition systems. **A prelude to neural networks**, 1994. 287-318.

33. MITCHELL, T. M. **Machine Learning**. [S.l.]: The McGraw-Hill Companies, 1997.

34. ZHU, X.; GOLDBERG, A. B. **Introduction to Semi-Supervised Learning**. [S.l.]: Morgan and Claypool Publishers, 2009.

35. TAN, P. N.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining**. [S.l.]: Pearson Education, 2006.

36. QUINLAN, J. R. **C4.5:** programs for machine learning. [S.l.]: Morgan Kaufmann Publishers Inc., 1993.

37. CHENG, W.; HUHN, J.; HULLERMEIER, E. **Decision tree and instance-based learning for label ranking**. Proceedings of the 26th Annual International Conference on Machine Learning. Quebec - Canadá: [s.n.]. 2009. p. 161-168.

38. ROSATELLI, M. C.; TEDESCO, P. A. **Diagnosticando o Usuário para Criação de Sistemas Personalizáveis**. Anais do XXIII Congresso da SBC - III Jornada de MCIA Porto Alegre. [S.l.]: SBC. 2003.

39. GHANI, R.; FANO, A. **Building Recommender Systems using a Knowledge Base of Product Semantics**. In 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems. Málaga, Espanha: [s.n.]. 2002.

40. MIYAHARA, K.; PAZZANI, M. J. **Collaborative filtering with the simple Bayesian classifier**. Proceedings of the 6th Pacific Rim international conference on Artificial intelligence. Melbourne, Australia: Springer-Verlag. 2000.

41. PRONK, V. et al. **Incorporating user control into recommender systems based on naive bayesian classification**. Proceedings of the 2007 ACM conference on Recommender systems. Minneapolis, USA: ACM. 2007. p. 73-80.

42. GUTTA, S. et al. **TV Content Recommender System**. Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence. [S.l.]: AAAI PRess. 2000.

43. HAYKIN, S. **Neural Networks:** A Comprehensive Foundation. [S.l.]: Prentice Hall PTR, 1998.

44. PAZZANI , M.; BILLSUS, D. Learning and Revising User Profiles: The Identification ofInteresting Web Sites. **Machine Learning**, 1997. 313-331.

45. HSU, S. H. et al. **AIMED:** a personalized TV recommendation system. Proceedings of the 5th European conference on Interactive TV: a shared experience. Amsterdam, Holanda: Springer-Verlag. 2007. p. 166-174.

46. KANG, H.; YOO, S. J. SVM and Collaborative Filtering-Based Prediction of User Preference for Digital Fashion Recommendation Systems. **IEICE - Trans. Inf. Syst.**, v.E90-D, 2007. 2100-2103.

47. XU, J. A.; K, A. **A SVM-based personal recommendation system for TV programs**. Multi-Media Modelling Conference Proceedings. Beijing: [s.n.]. 2006.

48. XIA, Z.; DONG, Y.; XING, G. **Support vector machines for collaborative filtering**. Proceedings of the 44th annual Southeast regional conference. Melbourne, Florida: ACM. 2006.

49. MATSUBARA, E. T. **Algoritmo de Aprendizado Semi-Supervisionado CO-TRAINING e sua Aplicação na Rotulação de Documentos**. São Carlos: ICMC-USP, 2004.

50. ZHU, X. **Semi-Supervised Learning Literature Survey**. [S.l.]. 2008.

51. THRUN, S. Lifelong learning algorithms. In: _____ **Learning to learn**. [S.l.]: Kluwer Academic Publishers, 1998. p. 181-209.

52. THRUN, S. B.; MITCHELL, T. M. **Lifelong Robot Learning**. [S.l.]: University of Bonn, 1993.

53. COUPLING Semi-Supervised Learning of Categories and Relations. Proceedings of the NAACL HLT Workshop on Semi-supervised Learning for Natural Language Processing. [S.l.]: [s.n.]. 2009.

54. CARLSON, A. et al. **Toward an Architecture for Never-Ending Language Learning**. Proceedings of the Conference on Artificial Intelligence (AAAI). [S.l.]: [s.n.]. 2010.

55. CARLSON, A. et al. **Coupled Semi-Supervised Learning for Information Extraction**. Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM). [S.l.]: [s.n.]. 2010.

56. MITCHELL, T. M. et al. **Populating the Semantic Web by Macro-Reading Internet Text**. Invited Paper, In Proceedings of the International Semantic Web Conference (ISWC). [S.l.]: [s.n.]. 2009.

57. BANKO, M.; ETZIONI, O. **Strategies for lifelong knowledge extraction from the web**. Proceedings of the 4th international conference on Knowledge capture. Whistler, Canadá: ACM. 2007. p. 95-102.

58. LENAT, D. B. **Automated theory formation in mathematics**. Proceedings of the 5th international joint conference on Artificial intelligence. Cambridge, USA: Morgan Kaufmann Publishers. 1977.

59. CURRAN, J. R.; MURPHY, T.; SHOLZ, B. **Minimising semantic drift with mutual exclusion bootstrapping**. PACLING. [S.l.]: [s.n.]. 2007.

60. SHANI, G.; GUNAWARDANA, A. Evaluating Recommendation Systems. In: _____ **Recommender Systems Handbook**. [S.l.]: Springer US, 2011. p. 257-297.

61. SELFRIDGE, O. G. The Gardens of Learning - A Vision for AI. **AI Magazine AAAI 14**, 1993.

62. BLUM, A.; MITCHELL, T. **Combining labeled and unlabeled data with co-training**. COLT'98 Proceedings of the eleventh annual conference on Computational learning. New York: ACM. 1998.

63. QUINLAN, J. R. **C4.5:** programs for machine learning. [S.l.]: Morgan Kaufmann Publishers Inc, 1993.

64. PEDRO, S. D. S.; HRUSCHKA-JR., E. R. **Conversing Learning:** active learning and active social interaction for human supervision in never-ending learning

systems. Proceedings of the 13th Ibero-American Conference on AI (IBERAMIA). [S.l.]: [s.n.]. 2012.

65. ZUNINO, D. L. **A matemática na escola:** aqui e agora. Porto Alegre: Artes Médicas, 1995.

66. ZHOU, F.; DUH, H. B.-L.; BILLINGHURST, M. **Trends in augmented reality tracking, interaction and display:** A review of ten years of ISMAR. Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality. [S.l.]: [s.n.]. 2008. p. 15-18.

67. YUEN, S. C.-Y.; YAOYUNEYONG, G.; JOHNSON, E. Augmented Reality and Education: Applications and Potentials. **Reshaping Learning**, 2013. 385-414.

68. ROMERO, C.; VENTURA, S.; DE BRA, P. Knowledge discovery with genetic programming for providing feedback to courseware author. **User Model. User-Adapted Interaction: J. Personalization Res.**, 14, 2004. 425-464.

69. ROMERO, C. et al. **Handbook of Educational Data Mining**. New York: Taylor & Francis, 2010.

70. ROMERO, C.; VENTURA, S. Educational Data Mining: a Review of The State of The Art. **IEEE Transactions on Systems, Man and Cybernetics, part C: applications and reviews**, 40, November 2010. 601-618.

71. ROMERO, C.; VENTURA, S. **Data Mining in E-Learning**. Ashurst: Wit Press, 2006.

72. ROMERO, C.; VENTURA, S.; DE BRA, P. Knowledge discovery with genetic programming for providing feedback to courseware author. **User Model. User-Adapted Interaction: J. Personalization Res.**, 2004. 425-464.

73. ROMERO, C. et al. **Handbook of Educational Data Mining**. New York: Taylor & Francis, 2010.

74. ROMERO, C.; VENTURA, S. **Data Mining in E-Learning**. Ashurst: Wit Press, 2006.

75. RAO, H.; WAI-TAT, F. **A General Framework for a Collaborative Mobile Indoor Navigation Assistance System**. roceedings of the 3rd International Workshop on Location Awareness for Mixed and Dual Reality. [S.l.]: [s.n.]. 2013.

76. PARK, S. T.; CHU, W. **Pairwise preference regression for cold-start recommendation**. RECSYS '09: Proceedings of the third ACM Conference on Recommender Systems. New York: [s.n.]. 2009. p. 21-28.

77. MINAEI-BIDGOLI, B. **Data Mining for a Web-Based Educational System**. Doctor of Philosophy. [S.l.]: Departament of Computer Science and Engineering Michigan State University. 2004. p. 267.

78. MILGRAM, P.; KISHINO, F. Taxonomy of Mixed Reality Visual Displays. **EICE Transactions on Information and Systems**, 1994. 1321-1329.

79. MERCERON , A.; YACEF, K. **Educational data mining:** A case study. Proc. Int. Conf. Artif. Intell. Educ. Amsterdam, The Netherlands: [s.n.]. 2005. p. 1-8.

80. MERCERON, A.; YACEF, K. **Educational data mining:** A case study. In: Proc. Int. Conf. Artif. Intell. Educ. Amsterdam, The Netherlands: [s.n.]. p. 2005.

81. KOEDINGER, K. et al. **An open repository and analysis tools for fine-grained, longitudinal learner data**. Proceedings 1st Int. Conf. Educ. Data Mining. Montreal, Canada: [s.n.]. 2008. p. 157-166.

82. KOEDINGER, K. et al. **An open repository and analysis tools for fine-grained, longitudinal learner data**. In: Proceedings 1st Int. Conf. Educ. Data Mining. Montreal, Canadá: [s.n.]. 2008.

83. CASCALES, A. et al. An Experience on Natural Sciences Augmented Reality Contents for Preschoolers. **Lecture Notes in Computer Science : Virtual, Augmented and Mixed Reality. Systems and Applications**, 2013. 103-112.

84. BRUSILOVSKY, P.; MILLAN, E. User Models for Adaptive Hypermedia and Adaptive Educational Systems. In: _____ **The Adaptive Web:** Methods and Strategies. Berlim, Heidelber, New York: Springer-Verlag, 2007.

85. BRUSILOVSKY, P. Adaptative Hypermedia. **User Modeling and User Adapted Interaction**, 11, 2001. 87-110.

86. BLUM, A.; MITCHELL, T. **Combining labeled and unlabeled data with co-training**. COLT' 98 Proceedings of the eleventh annual conference on Computational learning theory. ACM, New York, NY, USA: [s.n.]. 1998.

87. BANDEIRA, F.; CARDOSO, A.; CAIRRÃO, Á. The Wearable World in the Palm of our Hand: The Perceived Importance of Augmented Reality in Marketing Strategies. **International Journal of Business and Social Research**, 2013.

88. AZUMA, R. A Survey of Augmented Reality. **Presence: Teleoperators and Virtual Environments**, August 1997. 355-385.

89. ANDRADE, A. F. D. et al. **Uma Aplicação da Teoria Sociointeracionista de Vygotsky para construção de um Modelo de Aluno**. XIV Simpósio Brasileiro de Infolmática na Educação – SBIE. Rio de Janeiro: UFRJ. 2003.

90. BROWN-JR, H. J. **De Pai para Filho**. [S.l.]: Ediouro, 2005. 152 p.