Instituto de Matemática e Estatística
Universidade de São Paulo

Dissertação apresentada ao Programa de Pós-graduação
em Ciência da Computação para obtenção do
título de mestre em ciências

Orientador: Prof. Dr. Arnaldo Mandel

# AppRecommender: um recomendador de aplicativos GNU/Linux

Tássia Camões Araújo
<tassia@gmail.com>

São Paulo, 8 de julho de 2011

**Resumo**

A crescente oferta de programas de código aberto na rede mundial de computadores expõe potenciais usuários a inúmeras possibilidades de escolha. Em face da pluralidade de interesses destes indivíduos, mecanismos eficientes que os aproximem daquilo que buscam trazem benefícios para eles próprios, assim como para os desenvolvedores dos programas. O *AppRecommender* é um recomendador de aplicativos GNU/Linux que realiza uma filtragem no conjunto de programas disponíveis e oferece sugestões individualizadas para os usuários. Tal feito é alcançado por meio da análise de perfis e descoberta de padrões de comportamento na população estudada, de sorte que apenas os aplicativos considerados mais suscetíveis a aceitação sejam oferecidos aos usuários.

**Palavras-chave:** Sistemas de recomendação, distribuições GNU/Linux.

**Abstract**

The increasing availability of open source software on the World Wide Web exposes potential users to a wide range of choices. Given the individuals plurality of interests, mechanisms that get them close to what they are looking for would benefit themselves and the software developers as well. *AppRecommender* is a recommender system for GNU/Linux applications which performs a filtering on the set of available software and individually offers suggestions to users. This is achieved by analyzing profiles and discovering patterns of behavior of the studied population, in a way that only those applications considered most prone to acceptance are presented to users.

**Keywords:** Recommender systems, GNU/Linux distributions.

# Sumário

# Lista de Figuras

# Lista de Tabelas

*Capítulo 1*

# Introdução

O universo de programas livres e de código aberto oferece aos usuários uma grande amplitude e diversidade de opções no que diz respeito a aplicativos para complementar seus sistemas. No entanto, muitas dessas alternativas permanecem em relativa obscuridade, pois o caráter majoritariamente não comercial desses sistemas se reflete na ausência de propaganda e outras formas de divulgação ostensiva. Desta forma, a descoberta de programas úteis para um determinado usuário por vezes empaca no excesso de informações disponíveis e organização inadequada. É costume referir-se a esse fenômeno (p. ex., [Iyengar 2010]) como "mais é menos", no sentido de que o aumento da disponibilidade de escolhas pode confundir o usuário e diminuir sua satisfação.

Neste contexto de muitas possibilidades onde poucas são de fato atrativas, um sistema capaz de recomendar aplicativos que presumidamente são objeto de interesse de usuários exerceria um papel importante. Desenvolvedores se beneficiariam por meio de um consequente aumento na utilização de seus programas que, por serem experimentados por mais usuários, certamente receberiam mais relatórios de erro (*bug reports*), sugestões e contribuições diversas. Para os usuários o benefício seria alcançado de forma mais direta, dado que poupariam tempo e recursos outrora dedicados a buscas e filtragens manuais para encontrar os aplicativos mais adequados a seu ambiente de trabalho.

Tais benefícios motivaram a concepção do *AppRecommender*, um recomendador de aplicativos GNU/Linux desenvolvido no âmbito de um trabalho de mestrado, cujo objetivo principal é a experimentação de diferentes estratégias para recomendação no contexto de componentes de software.

O presente trabalho está organizado da seguinte forma: os capítulos 2 e 3 trazem uma breve introdução sobre distribuições GNU/Linux e sistemas de recomendação. No capítulo 4 trabalhos correlatos são apresentados e o 5 apresenta o AppRecommender como uma nova solução para o problema exposto. Em seguida são apresentados os experimentos realizados no capítulo 6 e, por fim, o capítulo 7 traz considerações finais sobre o presente trabalho e perspectivas de trabalhos futuros.

*Capítulo 2*

# Distribuições GNU/Linux

Neste capítulo o contexto histórico que propiciou o surgimento das distribuições GNU/Linux é apresentado, seguido por tópicos concernentes ao sistema de empacotamento de softwares que está diretamente relacionado com o a proposta deste trabalho.

## 2.1 Surgimento

Distribuições GNU/Linux, popularmente conhecidas como *distros*, são variações do sistema operacional composto pelo kernel Linux e milhares de aplicativos em sua maioria desenvolvidos pelo projeto GNU. As primeiras iniciativas neste domínio surgiram em circunstâncias que favoreciam o desenvolvimento colaborativo, abertura de código e comunicação predominatemente por meio da Internet.

O projeto GNU[1] foi criado em 1983 por Richard Stallman com o objetivo principal de desenvolver um sistema operacional livre em alternativa ao UNIX[2] – solução comercial amplamente difundida na indústria – e que fosse compatível com os padrões POSIX[3,4]. Nos anos 90 o projeto GNU já havia atraído muitos colaboradores, que num curto espaço de tempo haviam desenvolvido inúmeros aplicativos para compor o sistema operacional. No entanto, o desenvolvimento do núcleo do sistema (*GNU Hurd*) não acompanhou o ritmo dos demais aplicativos.

Em outubro de 1991 o estudante finlandês Linus Torvalds publicou a versão 0.02 do Freax, o núcleo de um sistema operacional (*kernel*, em inglês) desenvolvido por ele na universidade, com o intuito de atrair colaboradores para o projeto. Mais tarde Torvalds admitiu que não imaginava que aquele projeto desenvolvido sem grandes pretensões teria a dimensão do que hoje se conhece como Linux [Torvalds and Diamond 2001].

Com o anúncio de Torvalds, Stallman vislumbrou a possibilidade de acelerar o lançamento do sistema operacional livre se os aplicativos GNU que já estavam prontos fossem combinados com o núcleo recém-lançado – de fato, a primeira versão estável do GNU Hurd foi lançada apenas em 2001. Em 1992 o Linux foi licenciado sob a GNU GPL[5] e as equipes dos dois projetos começaram a trabalhar na adaptação do kernel Linux para o ambiente GNU. Este esforço conjunto desencadeou o surgimento das primeiras distribuições GNU/Linux.

As distros oferecem diferentes "sabores" do sistema operacional, a exemplo do Debian, Fedora, Mandriva e Ubuntu, que são constituídos por aplicativos criteriosamente selecionados

[1] hhttp://www.gnu.org

[2] http://www.unix.org/

[3] Acrônimo para *Portable Operating System Interface*, é uma família de normas definidas pelo IEEE com foco na portabilidade entre sistemas operacionais.

[4] Disponível em http://standards.ieee.org/develop/wg/POSIX.html

[5] Acrônimo para *General Public License*, é um suporte legal para a distribuição livre de softwares.

por seus desenvolvedores. Tais iniciativas tendem a reduzir a complexidade de instalação e atualização do sistema para usuários finais [Cosmo et al. 2008]. Os desenvolvedores (ou *mantenedores*) de distribuições atuam como intermediários entre os usuários e os autores dos softwares (neste contexto denominados de *upstreams*), por meio do encapsulamento de componentes de software em abstrações denominadas *pacotes*.

## 2.2 Empacotamento de programas

### 2.2.1 Pacotes Debian

. Binários e fontes
. upstream, maintainer, uploader
. Prioridade: required, important, standard, optional, extra
. Base system = required or important (muitos são marcados como essenciais)
. Essenciais: o gerenciador de pacotes se recusa a remover, a menos que seja forçado

### 2.2.2 Pacotes rpm

### 2.2.3 Relação entre pacotes

. Dependência: Pre-depends, Depends, Recommends, Suggests, Enhances
. Anti-dependência: Breaks, Conflicts, Replaces
. Pacotes virtuais: Provides
. Entre fontes e binários: Build-Depends, Build-Depends-Indep, Build-Conflicts, Build-Conflicts-Indep

### 2.2.4 O Repositório de Pacotes

. The Debian Archive,
. Áreas: main, contrib, non-free
. Seções atualmente particionam o repositório em 53 grupos de pacotes[6], para fins de organização. a seção de um pacote é definida no momento do empacotamento (arquivo *control*)em seu conteúdo

## 2.3 Sistemas gerenciadores de pacotes

### 2.3.1 APT

O gerenciamento de pacotes em sistemas Debian GNU/Linux e derivados é realizado através do *APT (Advanced Packaging Tool)*[7]. Ações como a busca, obtenção, instalação, atualização e remoção de pacotes são disparadas pelo *APT*, que num nível mais baixo faz uso do *dpkg*, ferramenta que de fato realiza instalações e remoções de softwares. O *APT* também gerencia de maneira eficiente as relações de conflito e dependência entre pacotes. Ao receber um pedido de modificação da configuração do sistema – por exemplo, instalação de um novo componente – o *APT* tenta satisfazer a requisição a partir do conhecimento de como obter os componentes (endereço dos repositórios de pacotes) e das relações de dependência entre os mesmos. Desta forma, o gerenciador promove a instalação de todas as dependências de um pacote antes de instalá-lo, ao passo que não permite a instalação de pacotes que conflitem com outros já instalados no sistema.

[6] http://packages.debian.org/unstable/
[7] http://wiki.debian.org/Apt

### 2.3.2 Synaptic

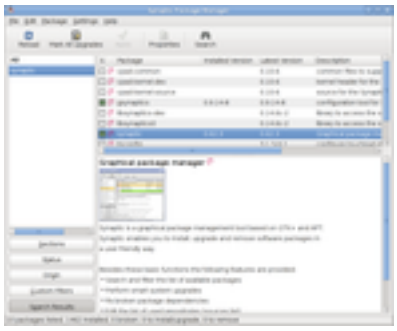

Figura 2.1: Screenshot do synaptic

### 2.3.3 Ubuntu Software Center

Ferramenta inicialmente denominada *AppCenter*, foi projetada com o intuito de centralizar as atividades relativas à instalação de aplicativos em sistemas Ubuntu (navegação pelo repositório, instalação e remoção de pacotes). Foi desenvolvido com base no *gnome-app-install*, sendo escrito em Python e usa GTK+ como biblioteca gráfica. Serve como interface para o APT, dado que o sistema de empacotamento desta distribuição é herdado do Debian.

Figura 2.2: Sreenshot do software-center

## 2.4 Seleção de programas

O conjunto de programas instalados num determinado sistema é resultado de dois processos de seleção: o primeiro é realizado no âmbito do desenvolvimento da distribuição e o segundo faz parte da manutenção cotidiana do sistema, realizado pelo usuário que administra a máquina.

A seleção e configuração dos aplicativos básicos de uma distribuição (instalados por padrão) são de responsabilidade da equipe que a desenvolve, com diferentes níveis de interferência da comunidade. Esta é uma questão crucial nos projetos, dado que é um dos fatores mais considerados por usuários finais na escolha por qual distribuição adotar. Por este motivo, mudanças na seleção de pacotes resulta em frequentes polêmicas em torno do tema, como a gerada pelo anúncio de que o projeto *Ubuntu* abandonaria o *Gnome* como interface padrão de usuário [Paul 2010].

Por outro lado, após a configuração de um sistema básico, este é geralmente personalizado para atender às demandas específicas dos usuários finais, o que é realizada por meio da instalação de programas adicionais. Ainda que a infraestrutura de instalação de softwares provida pelas distribuições (geralmente baseada em pacotes) simplifique a manutenção de sistemas [Cosmo et al. 2008], a seleção dos programas depende de uma ação humana (do

administrador). Com o desenvolvimento deste trabalho, pretende-se auxiliar o indivíduo nesta tarefa, especialmente quando este não possuir experiência pessoal para realizar escolhas neste contexto.

*Capítulo 3*

# Sistemas Recomendadores

O estado da arte em sistemas de recomendação é apresentado neste capítulo. As circunstâncias que justificaram o surgimento de tais sistemas, objetivos e desafios comuns no desenvolvimento, técnicas que apoiam a composição das recomendações e métricas de avaliação são alguns dos tópicos tratados.

## 3.1 Contexto histórico

A popularização de recursos computacionais e do acesso à Internet nas últimas décadas favoreceu um aumento expressivo na quantidade e diversidade de conteúdo e serviços à disposição dos usuários. Um dos fatores para este aumento é que indivíduos anteriormente considerados meros consumidores apresentam-se atualmente como produtores de conteúdo. Surgiram inúmeros casos de sucesso de serviços criados e/ou mantidos por usuários, a exemplo de *blogs*, enciclopédias colaborativas como a Wikipedia[1], repositórios para compartilhamento de fotografia e vídeo, como Flickr[2] e Youtube[3], entre outros. [Castells 2006] analisa este fenômeno e afirma que a maioria da população acredita que pode influenciar outras pessoas atuando no mundo através da sua força de vontade e utilizando seus próprios meios. Como ilustração deste tipo de influência, é comum que os clientes levem em conta opiniões de antigos hóspedes ao escolher o hotel em que se hospedará na próxima viagem.

Num cenário de sobrecarga de informações, sistemas recomendadores começaram a ser desenvolvidos com o intuito de potencializar o processo de indicação bastante popular nas relações sociais, aumentando sua capacidade e eficácia [Resnick and Varian 1997]. Um exemplo de recomendação tradicional são as avaliações de livros e filmes produzidas por críticos de arte e publicadas nos principais jornais e revistas do país. Nos últimos anos, no entanto, percebeu-se que a opinião e o comportamento de usuários não especializados agregariam valor às recomendações se considerados. A Netflix[4] é uma locadora de filmes norte-americana pioneira em utilizar o preferências de usuários e o histórico de transações para produção de recomendações automatizadas.

Este trabalho se insere num contexto onde o conteúdo tratado diz respeito ao universo de programas disponíveis, que também teve um crescimento significativo nas últimas décadas. A produção de software pode acontecer por meio de processos distintos. Quando os projetos são criados por empresas, costumam receber a colaboração dos usuários de forma limitada, visto que as decisões-chave são tomadas dentro da organização, mesmo que o código seja aberto. Este é o modelo de desenvolvimento descrito por [Raymond 1999] como catedral. Por outro

[1] http://wikipedia.org
[2] http://flickr.com
[3] http://youtube.com
[4] http://www.netflix.com

lado, os projetos criados independentemente, formam ao longo do tempo uma comunidade de desenvolvedores interessados em colaborar, sendo estes os únicos responsáveis pelo sucesso ou fracasso do projeto. Neste modelo, denominado bazar, o código-fonte está disponível durante todo o processo de desenvolvimento, não apenas nos lançamentos, permitindo que a contribuição seja mais efetiva. Segundo o autor, este modelo é mais favorável ao sucesso, pois um bom trabalho de desenvolvimento de software é motivado por uma necessidade pessoal do desenvolvedor. Nesses casos observa-se com mais clareza o fenômeno do consumidor produtor descrito anteriormente.

Segundo [Simon and Vieira 2008], o ambiente em torno de comunidades de software livre e/ou de código aberto (FOSS[5]) propicia a construção coletiva de uma ampla gama de softwares de qualidade em constante atualização e evolução, organizados na forma de um rossio[6]. O principal objetivo deste trabalho é contribuir de alguma forma para que os usuários encontrem com mais facilidade os programas que de fato necessitam, diante da enorme quantidade de opções disponíveis.

## 3.2 O problema computacional

O problema da recomendação é comumente formalizado através de uma estrutura de pontuação como representação computacional da utilidade dos itens para os usuários ou clientes. A partir de avaliações feitas pelos próprios usuários do sistema, tenta-se estimar pontuações para os itens que ainda não foram avaliados pelos mesmos. Uma vez que esta estimativa tenha sido feita, pode-se recomendar os itens com maior pontuação estimada.

O conceito de utilidade, porém, é subjetivo e arduamente mensurável devido às dificuldades em distinguir qualitativamente e definir quantitativamente os fatores que a determinam. Portanto, com a ressalva de que estas medidas não representam necessariamente a realidade, as pontuações são usadas como aproximações, pois têm como base as avaliações registradas pelos próprios usuários.

## 3.3 Identidade

Identidade, geração e manutenção de perfis de usuários, reputação.

## 3.4 Privacidade

Concurso do NetFlix suspenso após processo judicial. Ataques comuns a sistemas de recomendação[7].

## 3.5 Ações e desafios

Sistemas recomendadores são implementados nos mais diversos contextos e podem ser desenvolvidos para propósitos distintos, referenciados na literatura como ações de sistemas de recomendação [Herlocker et al. 2004]. Encontrar itens mais relevantes, encontrar todos os itens relevantes, recomendar sequência de itens, facilitar a navegação num repositório

[5]Termo consolidado no idioma inglês, acrônimo para *Free/Open-Source Software*, que diz respeito aos programas licenciados livremente para garantir aos usuários o direito de uso, estudo e modificação, por meio da disponibilidade do seu código-fonte.

[6]Termo pouco utilizado neste contexto, escolhido pelo autor para estimular o uso da palavra em português que significa um conjunto de recursos utilizados em comum e equitativamente por uma determinada comunidade (equivale ao termo *commons* no idioma inglês).

[7]"You might also like: privacy risks of collaborative filtering".

grande de itens e expressão de opinião para auxiliar outros usuários estão entre as ações mais frequentes.

Recomendadores desenvolvidos para finalidades diferentes por vezes compartilham as mesmas técnicas para o cálculo das recomendações, havendo um tratamento específico dos resultados antes de serem apresentados aos usuários. No entanto, a avaliação de eficácia de um sistema de recomendação está diretamente relacionada com sua ação principal.

Os desafios do desenvolvimento de tais sistemas estão relacionados a questões inerentes ao problema e sua representação computacional. As estratégias e técnicas propostas devem levar em conta tais questões, algumas das quais foram apontadas por [Vozalis and Margaritis 2003] e são citadas a seguir.

**Qualidade das recomendações.** Usuários esperam recomendações nas quais eles possam confiar. Esta confiabilidade é alcançada na medida em que se diminui a incidência de falsos positivos, em outras palavras, recomendações que não interessam ao usuário;

**Esparsidade.** A existência de poucas relações usuário-item resulta numa matriz de relacionamentos esparsa, o que dificulta a localização de usuários com preferências semelhantes (relações de vizinhança) e resulta em recomendações fracas.

**Escalabilidade.** A complexidade do cálculo de recomendações cresce tanto com o número de clientes quanto com a quantidade de itens, portanto a escalabilidade dos algoritmos é um ponto importante a ser considerado.

**Transitividade de vizinhança.** Usuários que têm comportamento semelhante a um determinado usuário não necessariamente têm comportamento semelhante entre si. A captura deste tipo de relação pode ser desejável mas em geral esta informação não é resguardada, exigindo a aplicação de métodos específicos para tal.

**Sinônimos.** Quando o universo de itens possibilita a existência de sinônimos, a solução deve levar esta informação em conta para prover melhores resultados.

**Primeira avaliação.** Um item só pode ser recomendado se ele tiver sido escolhido por um usuário anteriormente. Portanto, novos itens precisam ter um tratamento especial até que sua presença seja "notada".

**Usuário incomum.** Indivíduos com opiniões que fogem do usual, que não concordam nem discordam consistentemente com nenhum grupo, normalmente não se beneficiam de sistemas de recomendações.

## 3.6 Estratégias de recomendação

Em geral as estratégias de recomendação são classificadas como baseadas em conteúdo (recomenda-se itens semelhantes aos que o usuário já escolheu), colaborativas (recomenda-se itens que usuários semelhantes escolheram) ou híbridas (diferentes abordagens são combinadas). Este trabalho considera uma classificação um pouco mais abrangente, a partir de taxonomias propostas por diferentes autores. A peculiaridade de cada abordagem está relacionada com a fonte de dados utilizada para produzir o conhecimento do recomendador e o tipo de técnica aplicada para extrair as recomendações.

### 3.6.1 Reputação dos itens

Popular entre serviços de venda como livrarias, sites de leilão e lojas em geral, esta estratégia consiste no registro de avaliações dos produtos produzidas por usuários, bem como na apresentação das mesmas no momento e local apropriado [Cazella et al. 2010].

Esta é uma abordagem de simples implementação, visto que geralmente depende apenas da manutenção dos dados originais e nenhuma análise posterior é realizada. Os desafios surgem quando se tenta quantificar automaticamente as avaliações, seja pelo processamento do texto e classificação entre avaliação boa ou ruim [], ou ainda, quando a reputação é composta por meio de outros parâmetros, por exemplo, a quantidade de vendas, reclamações e devoluções de um produto. Outra dificuldade é lidar com a premissa a imparcialidade dos usuários em suas opiniões, que geralmente não pode ser verificada devido a seu caráter subjetivo e estritamente pessoal.

Atualmente existem serviços especializados em reputação de produtos que apenas disponibilizam as avaliações, sem haver venda associada. Alguns exemplos são o *Trip Advisor*[8], que oferece avaliações sobre hotéis, restaurantes e recomendações em geral para viagens, e o *Internet Movie Database*[9], que armazena uma vasta coleção de informações sobre cinema (figura 3.1).

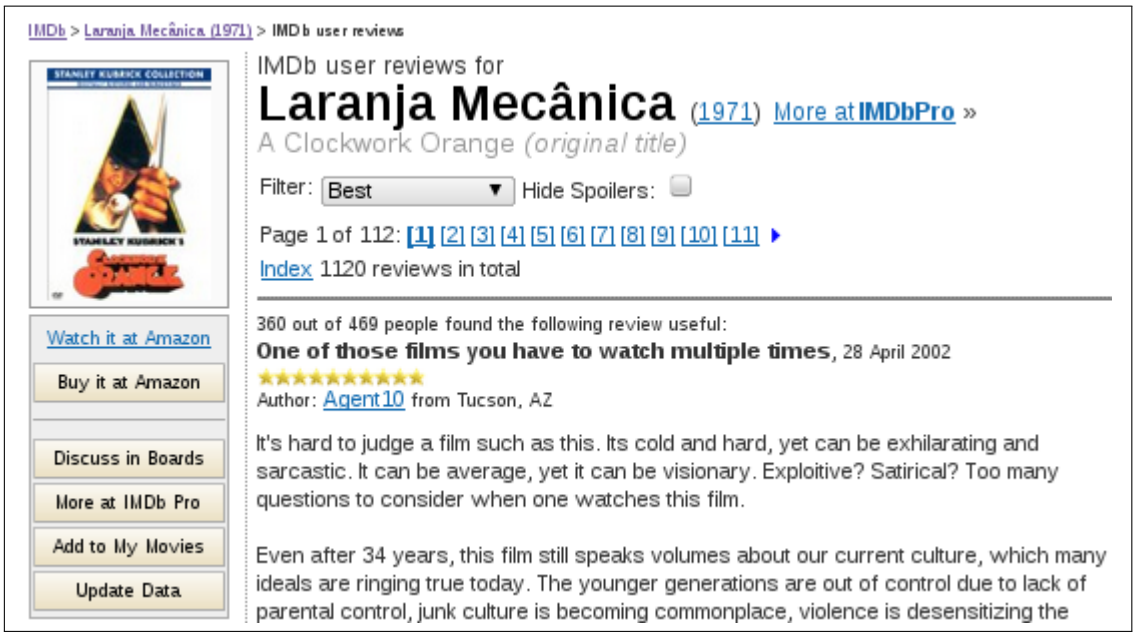


Figura 3.1: Avaliação de usuário no IMDb

### 3.6.2 Recomendação baseada em conteúdo

Esta abordagem parte do princípio de que os usuários tendem a se interessar por itens semelhantes aos que eles já se interessaram no passado [Herlocker 2000]. Em uma livraria, por exemplo, sugerir ao cliente outros livros do mesmo autor ou tema dos previamente selecionados é uma estratégia amplamente adotada.

O ponto chave desta estratégia é a representação dos itens por meio de suas características, por exemplo, descrever um livro pelo conjunto {*título, autor, editora, tema*}. A partir da identificação de atributos, aplica-se técnicas de recuperação da informação com o intuito de encontrar itens semelhantes (p. ex. *tf-idf* e *BM25*) ou de classificação a fim de encontrar itens relevantes (p. ex. *K-NN* e *Bayes ingênuo*). Estas e outras técnicas serão descritas em detalhes na seção 3.9.

Pelo fato de se apoiar na classificação dos itens, os resultados da recomendação são prejudicados nos casos em que os atributos não podem ser identificados de forma automati-

[8] http://www.tripadvisor.com/
[9] http://www.imdb.com/


Figura 3.2: Cenário de uma recomendação baseada em conteúdo

zada. Outro problema é a superespecialização, ou seja, a abrangência das recomendações fica limitada a itens similares aos já escolhidos pelos usuários [Adomavicius and Tuzhilin 2005].

### 3.6.3 Recomendação colaborativa

A recomendação colaborativa é fundamentada na troca de experiências entre indivíduos que possuem interesses em comum, portanto não exige o reconhecimento semântico do conteúdo dos itens. Esta estratégia é inspirada na técnica de classificação *K-NN* (seção 3.9.3).

A vizinhança de um determinado usuário é composta pelos usuários que estiverem mais próximos a ele. O ponto chave desta abordagem é a definição da função que quantifica a proximidade entre os usuários, que também pode ser herdada de soluções em classificação e recuperação da informação. A recomendação é então produzida a partir da análise dos itens que os seus vizinhos consideram relevantes. Geralmente os itens que ocorrem com maior frequência na vizinhança compõem a recomendação. A figura 3.3 ilustra o cenário de uma recomendação colaborativa.

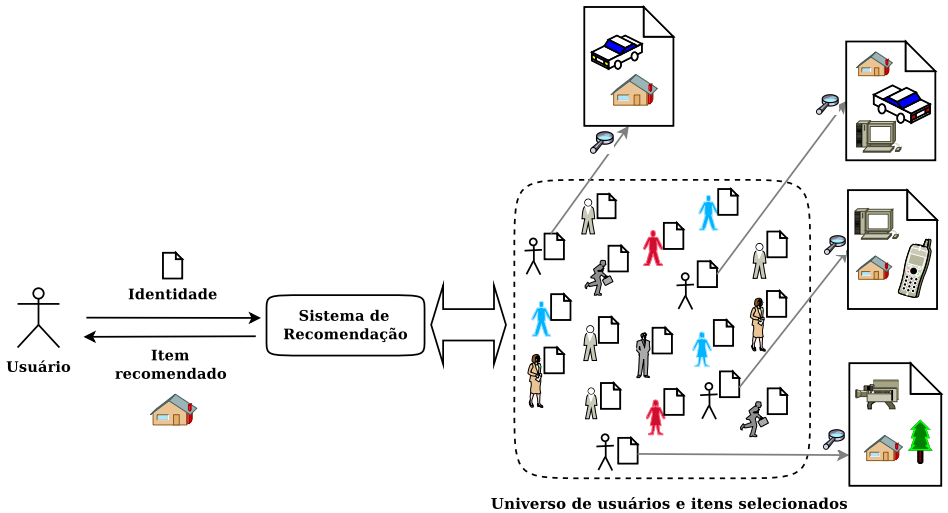


Figura 3.3: Cenário da recomendação colaborativa

O problema da superespecialização é superado, visto que a recomendação neste caso não se baseia no histórico do próprio usuário, consequentemente pode apresentar itens totalmente inesperados. Outro ponto positivo é a possibilidade de formação de comunidades de usuários pela identificação de interesses semelhantes entre os mesmos [Cazella et al. 2010].

### 3.6.4 Baseada em conhecimento

Esta estratégia tem como princípio a produção de recomendações a partir de um conhecimento previamente adquirido sobre o domínio da aplicação, em vez de avaliações prévias produzidas por usuários. A grande vantagem desta abordagem é que ela não depende de preferências individuais dos usuários, já que não usa a base de dados de avaliação usuário-item.

No entanto, a descoberta de conhecimento é o principal gargalo desta categoria de soluções, e por isso é mais utilizada nos casos em que já existe uma base de conhecimento disponível, por exemplo, na forma de uma ontologia [Adomavicius and Tuzhilin 2005]. Quando não é este o caso, técnicas de aprendizado de máquina e mineração de dados podem ser utilizadas para extrair correlações e padrões frequentes no comportamento dos usuários, por meio da análise das escolhas dos usuários ao longo do tempo.

Regras de associação são outro tipo de conhecimento formalizado, representado por regras de inferência que indicam a presença simultânea de conjuntos de itens numa determinada porcentagem dos casos conhecidos. As técnicas mais utilizadas para descoberta de tais regras são variações do algoritmo Apriori, apresentado na seção 3.9.6 [Kotsiantis and Kanellopoulos 2006]. Dado um conjunto de associações, a recomendação para determinado usuário é produzida de acordo com as regras satisfeitas pelo conjunto de itens que ele já tenha selecionado. Por exemplo, a regra $A, B, C \Rightarrow D$ seria satisfeita por usuários que possuem os itens $A$, $B$ e $C$, resultando na indicação do item $D$: *"Clientes que compraram os itens A, B e C também compraram o item D"*.

Segundo [Hegland 2003], recomendação baseada em conhecimento é frequentemente utilizada para sugestões implícitas, por exemplo, na definição do posicionamento de produtos numa prateleira ou a realização de propagandas dirigidas. Um caso popular deste estratégia é encontrado na loja virtual da empresa *Amazon*[10] (figura 3.4).


Figura 3.4: Recomendação por associação na Amazon

### 3.6.5 Baseada em dados demográficos

A estratégia demográfica fundamenta-se na composição de perfis de usuários e identificação de nichos demográficos para produção de recomendações. Os dados pessoais geralmente são coletados de forma explícita, através de um cadastro do usuário, e podem englobar informações como idade, sexo, profissão e áreas de interesse. Dados demográficos, no entanto, geralmente são utilizados em combinação com outras fontes de dados e técnicas diversas, como parte de uma estratégia de recomendação híbrida.

[10] http://www.amazon.com/

### 3.6.6 Estratégia híbrida

Sistemas de recomendação híbridos combinam duas ou mais estratégias, com o intuito de obter melhor performance do que a que as estratégias oferecem individualmente. A tabela 3.1 apresenta as principais técnicas de hibridização segundo [Burke 2002].

Tabela 3.1: Métodos de hibridização

| ***Método*** | ***Descrição*** |
|---|---|
| Ponderação | Pontuações de relevância oriundas de diversas técnicas de recomendação são combinadas para compor uma única recomendação. |
| Revezamento | O sistema reveza entre técnicas de recomendação diversas, de acordo com a situação do momento. |
| Combinação | Recomendações oriundas de diversos recomendadores diferentes são apresentados de uma só vez. |
| Combinação de atributo | Um algoritmo de recomendação único coleta atributos de diferentes bases de dados para recomendação. |
| Cascata | Um recomendador refina a recomendação produzida por outro. |
| Acréscimo de atributo | O resultado de uma técnica é usado como atributo de entrada para outra. |
| Meta-nível | O modelo que um recomendador "aprendeu" é usado como entrada para o outro. |

## 3.7 Similaridades e distâncias

tipos de dados intervalos numericos - possibilidade de padronizar a escala (pros e contras) requisitos fundamentais para uma operacao de distancia (p. 26) importancia da selecao de atributos - trach variables (p.27) missing values - posicao preenchida com valor negativo p. ex., e como as distancia devem ser calculadas? valor medio? - comparar com postings lists diferenca entre dissimilarida e distancia, que exige a propriedade da desigualdade triangular pearson e spearman como medidas de similaridade - coeficientes de correlacao, medem o quanto duas variaveis estao relacionadas - variam entre -1 e +1 . podem ser implementadas se as listas forem completadas com zeros . pearson busca uma correlacao linear entre os dois conjuntos - spearman procura uma relacao monotona (qd um cresce a outra cresce tb) dissimilaridades a partir de correlacoes d(x,y)=(1-c(x,y))/2 propriedades de medidas de similaridade (p.33) * proximityMatrix resemblaceMatrix transformar similaridade em dissimilaridade: d(x,y)=1-s(x,y) uso de coeficientes de correlacao para calcular proximidade entre objetos nao e adequado (p. 34) - envolvem calcula de medias em variaveis com unidades diferentes, o que nao faz sentido dados binarios coeficientes invariantes - para variaveis simetricas (p.37) nao faz diferenca qual coeficiente usar se os algoritmos dependem do ranking de similaridades - ex: single linkage method - relacao monotona entre os coeficientes coeficientes variantes - p.ex. considerar atributos raros: jaccard coeficiente

## 3.8 Seleção de atributos

Uma grande quantidade de atributos a ser considerada resulta em alta complexidade computacional, além de geralmente mascarar a presença de ruídos (dados que prejudicam a acurácia da classificação quando considerados). A fim de amenizar este problema, é comum a realização de um processo denominado seleção de atributos, que consiste em escolher alguns atributos dos dados e utilizar apenas estes como conjunto de treinamento para a classificação. Esta seleção equivale à substituição de um classificador complexo por um mais simples. [Manning et al. 2009] defende que, especialmente quando a quantidade de dados de treinamento é limitada, modelos mais fracos são preferíveis.

A seleção de atributos geralmente é realizada para cada classe em separado, seguida pela combinação dos diversos conjuntos. Abaixo são apresentados alguns métodos de escolha.

**Informação mútua.** Análise de quanto a presença ou ausência de um atributo contribui para a tomada de decisão correta por uma determinada classe. Informação mútua máxima significa que o atributo é um indicador perfeito para pertencimento a uma classe. Isto acontece quando um objeto apresenta o atributo se e somente se o objeto pertence à classe.

**Independência de eventos.** Aplicação do teste estatístico $\chi^2$ para avaliar a independência de dois eventos – neste caso, um atributo e uma classe. Se os dois eventos são dependentes, então a ocorrência do atributo torna a ocorrência da classe mais provável.

**Baseado em frequência.** Seleção dos atributos mais comuns para uma classe.

Os métodos apresentados acima são "gulosos", ou seja, assumem escolhas ótimas locais na esperança de serem ótimas globais. Como resultado, podem selecionar atributos que não acrescentam nenhuma informação para a classificação quando considerados outros previamente escolhidos. Apesar disto, algoritmos não gulosos são raramente utilizados devido a seu custo computacional [Manning et al. 2009].

## 3.9 Técnicas

O desenvolvimento de sistemas de recomendação tem suas raízes em áreas distintas e o problema computacional a ser tratado está fortemente relacionado com outros problemas clássicos, como classificação e recuperação de informação em documentos de texto.

A fim de obter a informação desejada, o usuário de uma ferramenta de busca deve traduzir suas necessidades de informação para uma consulta (*query*, em inglês), que geralmente é representada por um conjunto de *palavras-chave*. O desafio do buscador é recuperar os documentos da coleção que são relevantes para a consulta, baseando-se nos termos que a constituem. Ademais, visto que a busca pode retornar um número excessivo de documentos, é desejável que este resultado seja apresentado ao usuário em ordem decrescente de relevância, aumentando assim as chances de a informação desejada ser encontrada com rapidez. Para tanto, cada documento da coleção deve ter uma pontuação (peso) que indique seu grau de importância para a referida *query*. Traçando um paralelo com o problema de recomendação, a identidade e/ou o comportamento do usuário representaria a consulta ao sistema de busca, que provocaria o retorno dos itens de maior peso, ou seja, com maior potencial de aceitação pelo usuário.

Na busca por informação, assume-se que as necessidades do usuário são particulares e passageiras, e por isso a reincidência de *queries* não é muito frequente [Manning et al. 2009]. Porém, em situações onde se observa que as mesmas consultas são aplicadas com uma certa frequência, é interessante que o sistema suporte consultas permanentes. Desta forma a computação necessária pode ser realizada previamente e apresentada sempre que a consulta for requisitada. Se a classe de documentos que satisfazem a uma dessas *queries* permanentes é tida como uma categoria, o processo de realização das consultas prévias pode ser caracterizado como uma classificação. O problema da classificação diz respeito à determinação de relacionamentos entre um dado objeto e um conjunto de classes pré-definidas.

A recomendação pode ser vista como uma classificação, na qual os itens são categorizados entre duas classes: relevantes e irrelevantes – os relevantes seriam recomendados. Porém, a definição de consultas ou regras fixas para uma busca não é uma estratégia eficiente neste caso, porque a consulta estaria diretamente relacionada com a identidade do usuário e portanto deveria ser escrita especialmente para ele.

Todavia, a disciplina de inteligência artificial aborda a questão da classificação através de estratégias que não se baseiam em busca. Algoritmos de aprendizado de máquina são utilizados para a construção de modelos de classificação "inteligentes", que "aprendem" através da análise de exemplos. Os métodos supervisionados fundamentam-se na construção de um classificador que aprende na medida em que lhe são apontados exemplos de objetos classificados. São caracterizados como supervisionados porque as classes atribuídas aos objetos de treinamento são determinadas por um ser humano, que atua como um supervisor orientando o processo de aprendizado [Manning et al. 2009]. Por outro lado, algoritmos não supervisionados procuram identificar padrões de organização nos dados sem que haja uma classificação prévia dos exemplos.

A seguir são apresentadas algumas técnicas para tratamento destes problemas que também são utilizadas na construção de sistemas de recomendação, dando suporte às estratégias apresentadas na seção 3.6.

### 3.9.1 Medida *tf-idf*

Acrônimo para *term frequency - inverse document frequency*, *tf-idf* é uma medida de peso clássica utilizada em ferramentas de busca em texto para ordenação do resultado da consulta pela relevância dos documentos.

O universo da busca é uma coleção de documentos de texto. Um documento por sua vez é uma coleção de palavras, geralmente referenciadas como *termos do documento*. O conjunto de todas as palavras presentes nos documentos da coleção é denominado *dicionário* ou *vocabulário*. Desta forma, um documento $d$ composto por $n$ termos do vocabulário $V$ pode ser representado como $d = \{t_1, t_2, ..., t_n | 1 \leq i \leq n, t_i \in V\}$.

Contudo, alguns termos do vocabulário, designados como *stop words*, são normalmente desconsiderados no cálculo de relevância dos documentos por serem muito frequentes na coleção e, em decorrência disto, pouco informativos acerca do teor dos textos. Artigos e pronomes, por exemplo, geralmente figuram nesta categoria.

Outra consideração acerca da representação dos documentos como conjuntos de termos é a realização de normalizações morfológicas. Diferentes palavras que dizem respeito ao mesmo conceito podem ser utilizadas ao longo de uma coleção, por exemplo, os termos *casa*, *casinha* e *casas*. Em certos contextos, deseja-se que a busca por uma determinada variante retorne ocorrências de todas as outras possibilidades. Neste caso, os termos devem ser tratados em sua forma normalizada, eliminando variações como plurais e flexões verbais. Os processos mais comuns de normalização são: *stemming*, que reduz a palavra ao seu radical; e *lematização*, que reduz a palavra à sua forma canônica (por exemplo, verbos no infinitivo, substantivos no singular masculino etc). A figura 3.5 apresenta um documento de texto numa coleção hipotética[11] e a sua representação após eliminação de *stop words* e procedimento de *stemming*.

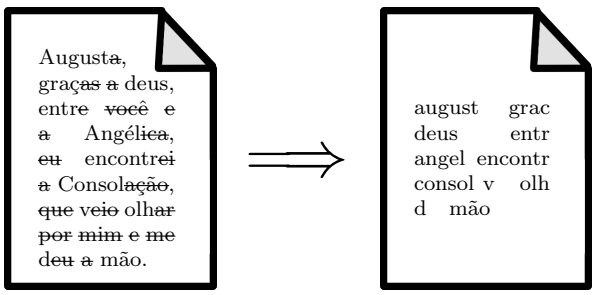


Figura 3.5: Eliminação de *stop words* e normalização do documento por *stemming*

[11] Os textos utilizados nos exemplos desta seção são excertos de letras de música de diversos compositores brasileiros.

A simples presença de um termo da *query* em um documento da coleção já é um indicativo de que o mesmo tem alguma relação com a consulta. No entanto, a quantidade de vezes que o termo ocorre é ainda mais informativo sobre sua relação com o conteúdo do documento. Intuitivamente, os documentos que referenciam os termos de uma *query* com mais frequência estão mais fortemente relacionados com a mesma, e por isso deveriam receber uma maior pontuação de relevância. O peso $tf_{t,d}$ (*term frequency*) quantifica esta noção intuitiva, relacionando documentos da coleção e termos do dicionário de acordo com a frequência destes nos documentos. Em sua abordagem mais simples, $tf_{t,d}$ é igual ao número de ocorrências do termo $t$ no documento $d$.

A figura 3.6 ilustra uma coleção de documentos, cujos valores de $tf_{t,d}$ para alguns termos do dicionário são apresentados na tabela 3.2. Por exemplo, a palavra *morena* ocorre duas vezes no documento (1), por isso $tf_{moren,1} = 2$. O cálculo do $tf$s já considera os radicais dos termos, resultantes de um processo de *stemming*. Em razão disto, $tf_{olh,2} = 3$, pois tanto a palavra *olho* quanto as diferentes flexões do verbo *olhar* contribuem para a contagem de frequência do termo *olh*. Na tabela 3.2, os radicais dos vocábulos são seguidos por algumas variações, a título de ilustração.

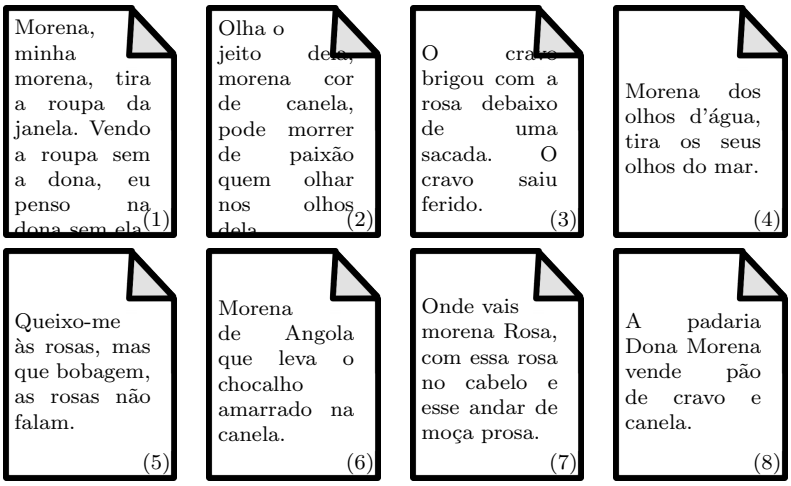


Figura 3.6: Coleção de documentos

Tabela 3.2: Frequência dos termos nos documentos da coleção

| $tf_{t,d}$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ***Termo*** \ ***Doc*** | **(1)** | **(2)** | **(3)** | **(4)** | **(5)** | **(6)** | **(7)** | **(8)** |
| moren {*a,o*} | 2 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| roup {*a,ão*} | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| don {*a,o*} | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| crav {*o,eiro*} | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 |
| canel {*a,eira*} | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| olh {*o,ar*} | 0 | 3 | 0 | 2 | 0 | 0 | 0 | 0 |
| ros {*a,eira*} | 0 | 0 | 1 | 0 | 2 | 0 | 2 | 0 |
| bob {*o,agem*} | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

O conjunto de pesos determinado pelos $tf$s dos termos de um documento pode ser entendido como um resumo quantitativo do mesmo. Esta visão do documento é comumente referenciada na literatura como "saco de palavras", onde a disposição das palavras é ignorada e apenas a quantidade de ocorrências para cada termo é considerada.

Uma medida de relevância baseada simplesmente na incidência dos termos da *query* nos documentos ($RI$) poderia ser calculada através da equação 3.1.

$$RI_{d,q} = \sum_{t \in q} tf_{t,d} \tag{3.1}$$

No entanto, alguns termos têm pouco poder de discriminação na determinação de relevância de um documento, por estarem presentes em quase todos os documentos. Ao passo que existem outros muito raros que quando presentes são forte indicativo de relevância. No contexto da coleção da figura 3.6, por exemplo, a *query* $\{morena, bobagem\}$ é composta por um termo muito frequente e outro muito raro. Coincidentemente, os documentos (3) e (5), contém apenas um dos dois elementos da consulta, porém ambos apresentam $tf_{t,d} = 1$ para os respectivos termos. Todavia, enquanto esta frequência se repete ao longo da coleção múltiplas vezes para o termo *morena*, ela é única para o termo *bobagem*, o que de fato diferencia o documento (5) dos demais.

O $idf_t$ (*inverse document frequency*) foi então introduzido para atenuar o efeito de termos muito comuns no cálculo de relevância, diminuindo o peso relacionado a um termo por um fator que cresce com sua frequência em documentos na coleção [Manning et al. 2009]. A equação 3.2 apresenta a forma clássica do $idf$, na qual $N$ representa o número de documentos da coleção e $df_t$ (*document frequency*) é o número de documentos que contém o termo $t$. É comum que o universo da busca seja uma coleção de documentos de altíssima dimensão, resultando em valores de $df_t$ muito discrepantes. O uso do log diminui a escala de valores, permitindo que frequências muito grandes e muito pequenas sejam comparadas sem problemas.

$$idf_t = \log \frac{N}{df_t} \tag{3.2}$$

Valores de $idf_t$ para a coleção da figura 3.6 são apresentados na tabela 3.3. Novamente os radicais dos termos são considerados: $idf_{morena} = idf_{moren} = \log \frac{8}{6} = 0.12$, enquanto $idf_{bobagem} = idf_{bob} = \log \frac{8}{1} = 0.9$.

Tabela 3.3: Valores de $idf_t$ para termos do dicionário

| ***Termo*** | moren | roup | don | crav | canel | olh | ros | bob |
|---|---|---|---|---|---|---|---|---|
| $idf_t$ | 0.12 | 0.9 | 0.6 | 0.6 | 0.42 | 0.6 | 0.42 | 0.9 |

A medida $tf\text{-}idf_{t,d}$ combina as definições de $tf$ e $idf$ (equação 3.3), produzindo um peso composto com as seguintes propriedades:

1. É alto quando $t$ ocorre muitas vezes em $d$ e em poucos documentos da coleção (ambos $tf$ e $idf$ são altos);
2. Diminui quando ocorre menos vezes em $d$ ($tf$ mais baixo) ou em muitos documentos da coleção ($idf$ mais baixo);
3. É muito baixa quando o termo ocorre em quase todos os documentos (mesmo para valores altos de $tf$, para termos muito comuns o peso $idf$ domina a fórmula, em decorrência do uso do log).

$$tf\text{-}idf_{t,d} = tf_{t,d} \cdot idf_t \tag{3.3}$$

A medida de relevância apresentada na equação 3.1 pode ser refinada para somar os pesos $tf\text{-}idf$ do documento $d$ com relação aos termos da *query* $q$, resultando na media $R_{d,q}$ apresentada na equação 3.4 [Manning et al. 2009].

$$R_{d,q} = \sum_{t \in q} tf\text{-}idf_{t,d} \tag{3.4}$$

A tabela 3.4 apresenta a ordenação dos documentos da coleção como resultado do cálculo de relevância por *tf-idf* para as consultas $q_1 = \{morena\}$, $q_2 = \{morena, bobagem\}$ e $q_3 = \{morena, dona, rosa\}$. Os valores $tf\text{-}idf_{t,d}$ foram obtidos a partir da equação 3.3, com os pesos das tabelas 3.2 e 3.3. Por exemplo, $tf\text{-}idf_{morena,1} = tf_{morena,1} \cdot idf_{morena} = 2 \cdot 0.12 = 0.24$.

Tabela 3.4: Ordenação dos documentos como resultado das consultas $q_1$, $q_2$ e $q_3$

| doc | $q_1$ | $R_{d,q}$ |
|---|---|---|
| | *morena* | |
| (1) | 0.24 | 0.24 |
| (2) | 0.12 | 0.12 |
| (4) | 0.12 | 0.12 |
| (6) | 0.12 | 0.12 |
| (7) | 0.12 | 0.12 |
| (8) | 0.12 | 0.12 |
| (3) | 0 | 0 |
| (5) | 0 | 0 |

| doc | $q_2$ | | $R_{d,q}$ |
|---|---|---|---|
| | *morena* | *bobagem* | |
| (5) | 0 | 0.9 | 0.9 |
| (1) | 0.24 | 0 | 0.24 |
| (2) | 0.12 | 0 | 0.12 |
| (4) | 0.12 | 0 | 0.12 |
| (6) | 0.12 | 0 | 0.12 |
| (7) | 0.12 | 0 | 0.12 |
| (8) | 0.12 | 0 | 0.12 |
| (3) | 0 | 0 | 0 |

| doc | $q_3$ | | | $R_{d,q}$ |
|---|---|---|---|---|
| | *morena* | *dona* | *rosa* | |
| (1) | 0.24 | 1.2 | 0 | 1.44 |
| (8) | 0.12 | 1.2 | 0 | 1.32 |
| (7) | 0.12 | 0 | 0.84 | 0.96 |
| (5) | 0 | 0 | 0.84 | 0.84 |
| (3) | 0 | 0 | 0.42 | 0.42 |
| (2) | 0.12 | 0 | 0 | 0.12 |
| (4) | 0.12 | 0 | 0 | 0.12 |
| (6) | 0.12 | 0 | 0 | 0.12 |

Existem diversas variantes para o cálculo dos pesos $tf_{t,d}$ e $idf_t$, propostas com o intuito de aperfeiçoar o processo de busca. Por exemplo, geralmente a presença de uma palavra 20 vezes num documento não tem de fato 20 vezes mais representatividade do que uma ocorrência única. Documentos distintos podem referenciar o mesmo conceito de forma concisa ou prolixa, e simplesmente este fato não deve ser motivo para pesos muito discrepantes com relação a uma *query*, visto que o teor do texto é o mesmo. A variante denominada *tf sub-linear* incorpora o logaritmo ao cálculo do *tf* para atenuar o crescimento do peso para valores crescentes de frequência (equação 3.5).

$$tf\text{-}sub_{t,d} = \begin{cases} 1 + \log tf_{t,d} & \text{, se } tf_{t,d} > 0 \\ 0 & \text{, caso contrário} \end{cases} \tag{3.5}$$

Outras abordagens alternativas utilizam normalizações por diversas medidas: comprimento do documento, comprimento médio dos documentos da coleção, *tf* máximo ou médio entre os *tf*s de todos os termos do documento, entre outros.

#### 3.9.1.1 Modelo de espaço vetorial

Uma coleção de documentos pode ser representada por um conjunto de vetores, sendo cada documento descrito como um vetor de termos do dicionário e os respectivos pesos *tf-idf* do documento. Tem-se como resultado uma visão da coleção como uma matriz de dimensões $M \times N$, na qual as linhas representam os $M$ termos do dicionário e as colunas os $N$ documentos da coleção. Esta representação, conhecida como *modelo de espaço vetorial*, é amplamente utilizada em soluções para recuperação da informação.

Assumindo que o vocabulário se restringe apenas aos termos para os quais os valores de *tf* e *idf* foram calculados (tabelas 3.2 e 3.3), a coleção de documentos da figura 3.6 pode ser representada no modelo de espaço vetorial pela matriz da tabela 3.5.

#### 3.9.1.2 Similaridade de cosseno

Medir a similaridade entre dois documentos pode ser útil, por exemplo, para disponibilizar o recurso "mais do mesmo", onde o usuário pede indicações de itens semelhantes a um que ele já conhece. Porém, se a diferença entre os vetores de pesos de dois documentos for usada como medida para avaliação de similaridade entre os mesmos, pode acontecer de documentos com conteúdo similar serem considerados diferentes simplesmente porque um é muito maior que o outro. Para compensar o efeito do comprimento dos documentos utiliza-se como medida de similaridade o cosseno do ângulo entre os vetores que os representam ($\theta$), apresentada na

Tabela 3.5: Representação da coleção no modelo de espaço vetorial

| $tf\text{-}idf_{t,d}$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ***Termo*** \ ***Doc*** | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) |
| moren | 0.24 | 0.12 | 0 | 0.12 | 0 | 0.12 | 0.12 | 0.12 |
| roup | 1.8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| don | 1.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0.6 |
| crav | 0 | 0 | 1.2 | 0 | 0 | 0 | 0 | 0.6 |
| canel | 0 | 0.42 | 0 | 0 | 0 | 0.42 | 0 | 0.42 |
| olh | 0 | 1.8 | 0 | 1.2 | 0 | 0 | 0 | 0 |
| ros | 0 | 0 | 0.42 | 0 | 0.84 | 0 | 0.84 | 0 |
| bob | 0 | 0 | 0 | 0 | 0.9 | 0 | 0 | 0 |

equação 3.6. O numerador representa o produto escalar dos dois vetores e o denominador a distância euclidiana entre os mesmos.

$$sim(d_1, d_2) = \cos(\theta) = \frac{V(\vec{d}_1) \cdot V(\vec{d}_2)}{|V(\vec{d}_1)||V(\vec{d}_2)|} \tag{3.6}$$

Dado um documento $d$, para encontrar os documentos de uma coleção que mais se assemelham a este, basta encontrar aqueles com maior similaridade de cosseno com $d$. Para tanto, pode-se calcular os valores $sim(d, d_i)$ entre $d$ e os demais $d_i$ documentos da coleção e os maiores valores indicarão os documentos mais semelhantes.

Considerando o fato de que *queries*, assim como documentos, são um conjunto me palavras, elas também podem ser representadas como vetores no modelo de espaço vetorial. A tabela 3.6 apresenta as consultas $q_1$, $q_2$ e $q_3$ neste espaço. Logo, a similaridade de cosseno também pode ser utilizada em buscas, considerando que os documentos mais similares a determinada *query* são os mais relevantes para a mesma (equação 3.7).

$$R_{d,q} = sim(d, q) \tag{3.7}$$

Tabela 3.6: Representação das queries no modelo de espaço vetorial

| $tf\text{-}idf_{t,d}$ | | | |
|---|---|---|---|
| ***Termo*** \ ***Query*** | $q_1$ | $q_2$ | $q_3$ |
| moren | 0.12 | 0.12 | 0.12 |
| roup | 0 | 0 | 0 |
| don | 0 | 0 | 0.6 |
| crav | 0 | 0 | 0 |
| canel | 0 | 0 | 0 |
| olh | 0 | 0 | 0 |
| ros | 0 | 0 | 0.42 |
| bob | 0 | 0.9 | 0 |

### 3.9.2 Okapi BM25

*Okapi BM25* é o modelo probabilístico considerado estado da arte em recuperação da informação [Pérez-Iglesias et al. 2009]. É amplamente utilizado no desenvolvimento de ferramentas de busca para os mais diversos domínios de aplicação. Tornou-se bastante popular

em virtude de seu destaque nas avaliações do TREC[12], sendo apontado como o melhor entre os esquemas de peso probabilísticos conhecidos [Betts 2007]. A título de ilustração, Xapian[13] e Lucene[14], bibliotecas livres para construção de motores de busca, são projetos de grande destaque na comunidade que utilizam o *BM25* como medida de pesos. O nome *Okapi* advém do primeiro sistema no qual foi implementado, denominado *City Okapi*, enquanto *BM* se refere à família de esquemas *Best Match*.

Embora seja comumente apresentado num contexto de busca em texto, o esquema não é específico para este domínio e pode ser usado na estimativa de relevância para qualquer tipo de recuperação de informação. A realização de consultas depende da descrição de itens e necessidades dos usuários, no entanto o modelo em princípio é compatível com inúmeras possibilidades de unidades descritivas [Jones et al. 2000]. Todavia, formalmente o modelo se refere a descrições de documentos como $D$ e de consultas como $Q$, ambas podendo ser decompostas em unidades menores. Cada componente é um atributo $A_i$, que pode assumir valores do domínio $\{presente, ausente\}$ ou valores inteiros não negativos, representando o número de ocorrências do termo no documento ou na *query*.

A busca no modelo probabilístico fundamenta-se no *Princípio de Ordenação por Probabilidade*, segundo o qual a maior eficácia de uma consulta num conjunto de dados é obtida quando os documentos recuperados são ordenados de maneira decrescente pela probabilidade de relevância em tal base de dados. [Robertson 1977]. No entanto, o ponto chave do Princípio é que a probabilidade de relevância não é o fim em si mesma, mas um meio de ordenar os documentos para apresentação ao usuário. Portanto, qualquer transformação desta probabilidade pode ser usada, desde que preserve a ordenação pela relevância [Jones et al. 2000].

#### 3.9.2.1 Modelo formal

Dado um documento descrito por $D$ e uma *query* $Q$, o modelo considera a ocorrência de dois eventos: $L = \{D \text{ é relevante para } Q\}$ e $\overline{L} = \{D \text{ não é relevante para } Q\}$. Para que a ordenação por relevância seja possível, calcula-se para cada documento a probabilidade $P(L|D)$. A aplicação do teorema de Bayes permite que $P(L|D)$ seja expressa em função de $P(D|L)$ (equação 3.8).

$$P(L|D) = \frac{P(D|L)P(L)}{P(D)} \tag{3.8}$$

Para evitar a expansão de $P(D)$, a chance de $(L|D)$ é utilizada em vez da probabilidade. Na verdade, o logaritmo da chance é aplicado (equação 3.9), considerando que esta é uma transformação que satisfaz o Princípio de Ordenação [Jones et al. 2000]. Ademais, dado que o último termo da fórmula é igual para todos os documentos, ele pode ser desconsiderado sem que isso altere a ordenação dos documentos. Desta forma, a equação 3.10 descreve uma pontuação por relevância referenciada como primária ($R\text{-}PRIM_D$).

[12] O *Text Retrieval Conference (TREC)* é uma conferência anual realizada pelo *U.S. National Institute of Standards and Technology (NIST)* que promove uma ampla competição em recuperação da informação de grandes coleções de texto com o intuito de incentivar pesquisas na área.

[13] http://xapian.org/

[14] http://lucene.apache.org/

$$\begin{aligned}
\log \frac{P(L|D)}{P(\overline{L}|D)} &= \log \frac{P(D|L)P(L)}{P(D|\overline{L})P(\overline{L})} \\
&= \log \frac{P(D|L)}{P(D|\overline{L})} + \log \cancel{\frac{P(L)}{P(\overline{L})}} \qquad (3.9) \\
\textit{R-PRIM}_D &= \log \frac{P(D|L)}{P(D|\overline{L})} \qquad (3.10)
\end{aligned}$$

Assim como o modelo de classificação *Bayes* ingênuo, o *BM25* assume que os atributos dos documentos são estatisticamente independentes de todos os outros. [Jones et al. 2000] justifica a suposição de independência de atributos pelos seguintes argumentos:

1. Facilita o desenvolvimento formal e expressão do modelo;
2. Torna a instanciação do modelo tratável computacionalmente;
3. Ainda assim permite estratégias de indexação e busca com melhor performance do que estratégias rudimentares, como o simples casamento de padrões aplicados a termos da *query* no documento.

De acordo com a suposição de independência, a probabilidade de um documento pode ser trivialmente derivada a partir das probabilidade de seus atributos. Logo, *R-PRIM*$_D$ poderia ser estimado como um somatório de probabilidades, cada uma relacionada a cada atributo da descrição $D$ (equação 3.11).

$$\begin{aligned}
\textit{R-PRIM}_D &= \log \prod_i \frac{P(A_i = a_i|L)}{P(A_1 = a_1|\overline{L})} \\
&= \sum_i \log \frac{P(A_i = a_i|L)}{P(A_1 = a_1|\overline{L})} \qquad (3.11)
\end{aligned}$$

No entanto, a fórmula 3.11 pressupõe a consideração de um componente para cada valor do atributo, por exemplo, para presença de um termo assim como para sua ausência. Uma alternativa mais natural seria considerar apenas valores para a presença, contabilizando a ausência como um *zero* natural. Desta forma, é subtraído da pontuação de cada documento o componente relativo a cada valor de atributo zerado (fórmula 3.12).

$$\begin{aligned}
\textit{R-BASIC}_D &= \textit{R-PRIM}_D - \sum_i \log \frac{P(A_i = 0|L)}{P(A_1 = 0|\overline{L})} \\
&= \sum_i \left( \log \frac{P(A_i = a_i|L)}{P(A_1 = a_1|\overline{L})} - \log \frac{P(A_i = 0|L)}{P(A_1 = 0|\overline{L})} \right) \\
&= \sum_i \log \frac{P(A_i = a_i|L)P(A_1 = 0|\overline{L})}{P(A_1 = a_1|\overline{L})P(A_i = 0|L)} \qquad (3.12)
\end{aligned}$$

Considerando $W_i$ como um peso para cada termo $t_i$ do documento (equação 3.13), *R-BASIC*$_D$ pode ser então reescrito em função deste peso, como na equação 3.14.

$$W_i = \log \frac{P(A_i = a_i|L)P(A_1 = 0|\overline{L})}{P(A_1 = a_1|\overline{L})P(A_i = 0|L)} \qquad (3.13)$$

$$\textit{R-BASIC}_D = \sum_i W_i \qquad (3.14)$$

No caso em que os atributos $A_i$ restringem-se a exprimir a presença ou ausência do termo $t_i$ (atributos binários), pode-se dizer que $P(A_1 = 0|L) = 1 - P(A_i = 1|L)$, o mesmo vale para $\overline{L}$. Portanto, considerando que $p_i = P(t_i \text{ ocorre } |L)$ e $\overline{p}_i = P(t_i \text{ ocorre } |\overline{L})$, a fórmula 3.13 pode ser usada como um peso para presença de termos. A pontuação de relevância para um documento seria então a soma dos pesos $w_i$ dos termos da *query* presentes no documento.

$$w_i = \log \frac{p_i(1 - \overline{p_i})}{\overline{p_i}(1 - p)} \tag{3.15}$$

A seguir será apresentada a interpretação do modelo formal a partir de informações disponíveis sobre a coleção de documentos, com o intuito de definir funções de peso eficazes para a ordenação por relevância.

#### 3.9.2.2 Incidência dos termos e atestação de relevância

A incidência dos termos nos documentos da coleção é uma informação que pode ser facilmente coletada e pode ser utilizada como parâmetro no cálculo da probabilidade de relevância. O popular $idf_t$ (equação 3.2) é uma medida plausível e, apesar de ter sido proposta baseada apenas na frequência de incidência dos termos, também pode ser derivada da equação 3.15 [Jones et al. 2000].

No entanto, apenas a incidência dos termos é uma base fraca para a estimativa de probabilidades de relevância. As estimativas podem ser refinadas através da consideração de dados acerca da real relevância ou irrelevância de documentos, obtidos por exemplo através de procedimentos de *atestação de relevância*[15].

A tabela de contingência da incidência dos termos é apresentada na tabela 3.7. $N$ representa o número total de documentos da coleção, enquanto $n$ representa o número de documentos nos quais o termo $t$ da *query* ocorre. Analogamente, $R$ é a quantidade de documentos relevantes para a consulta e $r$ a quantidade de documentos relevantes nos quais o termo ocorre.

Tabela 3.7: Tabela de contingência da incidência dos termos

| | Relevante | Irrelevante | Incidência na coleção |
|---|---|---|---|
| $t$ ocorre | $r$ | $n - r$ | $n$ |
| $t$ não ocorre | $R - r$ | $N - n - R + r$ | $N - n$ |
| total de documentos | $R$ | $N - R$ | $N$ |

Portanto, a probabilidade de um termo $t$ ocorrer num documento, dado que este é relevante para a *query* é $p = \frac{r}{R}$. Analogamente, dado que o documento não é relevante, $\overline{p} = \frac{n-r}{N-R}$. Desta forma, a equação 3.15 pode ser redefinida como a fórmula 3.16, que exprime o logaritmo da razão de chances de um termo ocorrer em documentos relevantes e irrelevantes.

$$w = \log \frac{r(N - n - R + r)}{(R - r)(n - r)} \tag{3.16}$$

Por fim, o fator de correção 0.5 é acrescido a cada termo central da fórmula para evitar que o peso seja zerado quando algum destes termos for *zero*.

$$RW_t = \log \frac{(r + 0.5)(N\text{-}n\text{-}R + r + 0.5)}{(R\text{-}r + 0.5)(n\text{-}r + 0.5)} \tag{3.17}$$

Se não houver informações provenientes da atestação de relevância, o $idf_t$ clássico pode ser utilizado (equação 3.2), ou ainda, uma variação de $RW_t$ a partir do estabelecimento de que $R = r = 0$ (equação 3.18).

[15] Mecanismo através do qual o usuário avalia o resultado da consulta, marcando os itens retornados como relevantes ou irrelevantes – no idioma inglês, referenciado como *relevance feedback*.

$$RW_t = \log \frac{N\text{-}n + 0.5}{n + 0.5} \tag{3.18}$$

#### 3.9.2.3 Distribuição dos termos nos documentos

A incidência dos termos na coleção distingue os documentos com relação aos termos da *query* no que diz respeito apenas à ocorrência ou ausência dos mesmos. Usando apenas esta medida não é possível portanto diferenciar dois documentos em relação a um termo se o mesmo ocorre em ambos. No caso em que dados de frequência dos termos são providos nas descrições dos documentos, esta informação pode contribuir para a estimativa de relevância do de um documento.

Assume-se que cada termo é associado a um conceito, ao qual um determinado documento pode estar relacionado ou não. Logo, para cada conceito existe um conjunto de documentos que dizem respeito a ele e outro conjunto que não (complementar ao primeiro). A frequência de um termo em um documento caracteriza sua ocorrência quantitativamente, porém, uma frequência maior que *zero* não significa que o documento esteja necessariamente relacionado com conceito do termo. Diante da impossibilidade de se prever esta relação conceitual, considera-se a distribuição de frequências dos termos nos documentos como uma mistura de duas distribuições, uma para cada um dos conjuntos [Jones et al. 2000].

Essa distribuição pode ser entendida como originada num modelo de geração de texto: o autor se depara com as posições de palavras nos documentos e escolhe termos para ocupar tais posições. Se a probabilidade de escolha de cada termo for fixa e todos os documentos forem de igual comprimento, caracteriza-se uma distribuição de *Poisson* para frequências dos termos nos documentos. Assume-se probabilidades diferentes para o conjunto de documentos relacionados ao conceito do termo e para o conjunto dos que não são – esta é a razão para a mistura de duas distribuições [Jones et al. 2000].

A derivação deste componente do peso é mais extensa e por esta razão foi omitida neste texto. A fórmula resultante é complexa, no entanto [Robertson and Walker 1994] examina o comportamento da mesma e propõe uma aproximação que apresenta comportamento similar à original, expressa pela equação 3.19.

$$RD_{t,D} = \frac{tf_{t,D}(k_1 + 1)}{k_1 + tf_{t,D}} \tag{3.19}$$

A constante $k_1$ determina o quanto o peso do documento em relação ao termo deve ser afetado por um acréscimo no valor de $tf_{t,D}$. Se $k_1 = 0$, então $RD_{t,D} = 1$, e $tf_{t,D}$ não interfere no peso final. Para valores altos de $k_1$, o peso passa a ter um crescimento linear com relação a $tf_{t,D}$. De acordo com experimentos do TREC, valores entre 1.2 e 2 são os mais indicados, visto que implicam numa interferência altamente não linear de $tf_{t,D}$, ou seja, após 3 ou 4 ocorrências o impacto de uma ocorrência adicional é mínimo [Jones et al. 2000].

No entanto, a modelagem através distribuições de *Poisson* assume que todos os documentos têm mesmo comprimento, o que não acontece na prática. Porém, uma interpretação ligeiramente estendida do modelo permite a consideração de documentos com comprimentos distintos.

Os comprimentos dos documentos da coleção podem variar por inúmeros motivos. Todavia, nesta nova interpretação assume-se que quando dois documentos acerca do mesmo conceito têm tamanhos distintos, a razão é simplesmente que um é mais verboso que o outro. Em outras palavras, considera-se que a recorrência de palavras deve-se sempre à repetição em vez de, por exemplo, melhor elaboração do tema. Partindo desta suposição, é apropriado estender o modelo normalizando o valor de $tf_{t,D}$ em função do comprimento do documento (equação 3.20).

$$RD_{t,D} = \frac{\frac{tf_{t,D}}{NL}(k_1+1)}{k_1 + \frac{tf_{t,D}}{NL}} = \frac{tf_{t,D}(k_1+1)}{k_1 * NL + tf_{t,D}} \tag{3.20}$$

Dado que o comprimento dos documentos pode ser medido de diversas formas (quantidade de palavras, caracteres e *bytes*, considerando ou não *stop words*), considera-se a medida uniformizada para normalização, obtida pela razão entre o comprimento dos documentos e o comprimento médio dos documentos ($\frac{l_d}{l_{avg}}$). Ademais, uma normalização simples resultaria na mesma pontuação para um documento de comprimento $l$ no qual o termo ocorre $tf$ vezes e para outro de comprimento $2l$ que contém $2tf$ ocorrências do termo. Este comportamento pode ser indesejável por exemplo quando se considera que a recorrência de palavras está geralmente associada ao aprofundamento do conceito, em vez de mera repetição.

A fórmula proposta (3.21) permite que a normalização ocorra em diferentes graus, de acordo com o ajuste do parâmetro constante $b$ que assume valores no intervalo $[0, 1]$. Se a configuração for $b = 1$, a normalização tem efeito completo (equivalente ao esquema *BM11*). Valores menores reduzem este efeito, e se $b = 0$ o comprimento do documento não afeta a pontuação final, (como no modelo *BM15*).

$$NL = ((1\text{-}b) + b\frac{l_d}{l_{avg}}) \tag{3.21}$$

$$RD_{t,D} = \frac{tf_{t,D}(k_1+1)}{k_1((1\text{-}b) + b\frac{l_d}{l_{avg}}) + tf_{t,D}} \tag{3.22}$$

#### 3.9.2.4 Consultas longas

Em situações onde as consultas podem ser descritas por *queries* longas, por exemplo, o caso em que um documento pode ser utilizado como base para a consulta, a consideração da frequência do termo na *query* pode ser mais um fator contribuinte para a estimativa de relevância. O componente $RQ_{t,Q}$ também é derivado a partir da modelagem em distribuições de *Poisson*, porém aplicadas ao conjunto de *queries* em vez do conjunto de documentos. O resultado é um peso semelhante ao $RD_{t,D}$, porém com parâmetros constantes próprios (equação 3.23). Todavia, para o caso de *queries* com poucos termos, este componente do peso deve ser desconsiderado.

$$RQ_{t,Q} = \frac{(k_3+1)qtf_{t,Q}}{k_3 + qtf_{t,Q}} \tag{3.23}$$

#### 3.9.2.5 Estimativa de relevância

Finalmente, a relevância de um documento $D$ para uma consulta $Q$ pode ser obtida pelo somatório dos pesos dos termos da *query* com relação a $D$. O peso de cada termo é obtido pelo produto dos componentes apresentados anteriormente, como indica a equação 3.24.

$$R_{D,Q} = \sum_{t \in Q} RW_t \cdot RD_{t,D} \cdot RQ_{t,Q} \tag{3.24}$$

### 3.9.3 K-NN

*K-nearest neighbors (k-NN)*, em português *k vizinhos mais próximos*, é mais um algoritmo de aprendizado supervisionado para classificação. Este método baseia-se no conceito de vizinhança, que representa um conjunto de objetos que estão próximos no espaço de busca.

O *K-NN* não exige nenhum treinamento prévio com os dados de exemplo, que podem ser diretamente armazenados como vetores de atributos acompanhados por suas devidas classes.

A classificação de um novo objeto parte do cálculo da vizinhança do mesmo, que é composta por $k$ objetos.

A determinação da vizinhança está diretamente relacionada com o conceito de proximidade entre objetos, que pode ser expressa em termos de similaridade ou de distância entre os mesmos (quanto maior a distância, menor a similaridade). Existem diversas medidas para mensurar estes conceitos, deve-se adotar a métrica que melhor se adeque ao domínio da aplicação e conjunto de dados. A tabela 3.8 apresenta algumas dessas medidas, onde os objetos $X$ e $Y$ são representados por seus vetores $\vec{X} = (x_1, ..., x_n)$ e $\vec{Y} = (y_1, ..., y_n)$. A similaridade de cosseno mede a similaridade de dois vetores através do cosseno do ângulo entre os mesmos. O coeficiente de *Pearson* é equivalente ao cosseno do ângulo entre os vetores centralizados na média. E o coeficiente de *Tanimoto* é uma extensão da similaridade de cossenos que resulta no índice de *Jaccard* para atributos binários.

Tabela 3.8: K-NN: Medidas de distância e similaridade entre objetos

| | |
|---|---|
| Distância euclidiana | $D(X,Y) = \sqrt{(x_1\text{-}y_1)^2 + (x_2\text{-}y_2)^2 + ... + (x_n\text{-}y_n)^2}$ |
| Similaridade de cosseno | $sim(X,Y) = \frac{\vec{X}\cdot\vec{Y}}{\lvert\vec{X}\rvert\lvert\vec{Y}\rvert} = \frac{\sum_{1\leq i\leq n} x_i y_i}{\sqrt{\sum_{1\leq i\leq n} x_i^2}\sqrt{\sum_{1\leq i\leq n} y_i^2}}$ |
| Coeficiente de *Pearson* | $P(X,Y) = \frac{\sum_{1\leq i\leq n}(x_i\text{-}\bar{x})(y_i\text{-}\bar{y})}{\sqrt{\sum_{1\leq i\leq n}(x_i\text{-}\bar{x})^2}\sqrt{\sum_{1\leq i\leq n}(y_i\text{-}\bar{y})^2}}$ |
| Coeficiente de *Tanimoto* | $T(X,Y) = \frac{\vec{X}\cdot\vec{Y}}{\lvert\vec{X}\rvert^2 + \lvert\vec{Y}\rvert^2\text{-}\vec{X}\cdot\vec{Y}}$ |

Após a definição de vizinhança, a classe mais frequente entre seus $k$ vizinhos é atribuída ao novo objeto. Desta forma, a similaridade também pode ser entendida como grau de influência entre os objetos. Os objetos mais semelhantes a um novo objeto terão maior influência no cálculo de sua classificação.

### 3.9.4 Classificador bayesiano

*Bayes ingênuo* é uma solução para classificação que figura entre os algoritmos de aprendizado de máquina supervisionados. O classificador apoia-se num modelo probabilístico que aplica o teorema de Bayes com fortes suposições de independência de atributos – por esta razão o método é considerado ingênuo. Em outras palavras, a presença ou ausência de um atributo em um objeto de uma classe não estaria relacionada com a incidência de nenhum outro atributo.

A decisão acerca da classe a qual um objeto pertence é tomada de acordo com o modelo de probabilidade máxima posterior *(MAP)*, indicada na equação 3.25. Dado que $C$ é o conjunto de classes e $x$ objeto a ser classificado, a classe atribuída a este será a que apresentar maior probabilidade condicionada a $x$. $\hat{P}$ é utilizado em vez de $P$ porque geralmente não se sabe o valor exato das probabilidades, que são estimadas a partir dos dados de treinamento.

$$c_{MAP} = \arg\max_{c\in C} \hat{P}(c|x) \tag{3.25}$$

A equação 3.26 aplica o Teorema de Bayes para probabilidades condicionadas. Na prática, apenas o numerador da fração interessa, visto que o denominador é constante para todas as classes, portanto não afeta o arg max (equação 3.27).

$$c_{MAP} = \arg\max_{c \in C} \frac{\hat{P}(x|c)\hat{P}(c)}{\cancel{\hat{P}(x)}} \tag{3.26}$$

$$= \arg\max_{c \in C} \hat{P}(x|c)\hat{P}(c) \tag{3.27}$$

É neste ponto que a independência de atributos é importante. Considera-se que um documento $x$ pode ser caracterizado por uma série de atributos $x_i$ – no caso de documentos de texto, os atributos são os próprios termos. Assumindo que a ocorrência de atributos acontece independentemente, tem-se que:

$$\hat{P}(x|c) = \hat{P}(x_1, x_2, ..., x_n|c) = \hat{P}(x_1|c)\hat{P}(x_2|c) \ ... \ \hat{P}(x_n|c) \tag{3.28}$$

Portanto, a função de decisão pode ser reescrita através da equação 3.29. Cada parâmetro condicional $\hat{P}(x_i|c)$ é um peso que representa a qualidade do atributo $x_i$ como indicador da classe $c$, enquanto que $\hat{P}(c)$ é a frequência relativa da classe $c$.

$$c_{MAP} = \arg\max_{c \in C} \ \hat{P}(c) \prod_{1 \leq i \leq n} \hat{P}(x_i|c) \tag{3.29}$$

Os parâmetros são obtidos através da estimativa de maior probabilidade *(MLE)*, que corresponde ao valor mais provável de cada parâmetro de acordo com os dados de treinamento. A equação 3.30 traz a estimativa de $\hat{P}(c)$, onde $N_c$ é o número de objetos da classe $c$ e $N$ é o número total de documentos.

$$\hat{P}(c) = \frac{N_c}{N} \tag{3.30}$$

As probabilidades condicionais são estimadas como a frequência relativa do atributo $x$ em objetos que pertencem à classe $c$. Na equação 3.31, $T_{cx}$ é o número de ocorrências de $x$ em objetos de exemplo da classe $c$ e $V$ é o conjunto de atributos que os objetos podem apresentar.

$$\hat{P}(x|c) = \frac{T_{cx}}{\sum_{x' \in V} T_{cx'}} \tag{3.31}$$

No entanto, a estimativa *MLE* é zero para combinações atributo-classe que não ocorrem nos dados de treinamento. Considerando que as probabilidades condicionais de todos os atributos serão multiplicadas (equação 3.29), a simples ocorrência de uma probabilidade zerada resulta na desconsideração da classe na referida classificação. E de fato, dados de treinamento nunca são abrangentes o suficiente para representar a frequência de eventos raros de forma adequada [Manning et al. 2009]. Para eliminar *zeros*, adiciona-se 1 a cada termo da equação 3.31:

$$\hat{P}(x|c) = \frac{T_{cx} + 1}{\sum_{x' \in V} T_{cx'} + 1} \tag{3.32}$$

O classificador bayesiano também é sensível a ruídos, logo, sua performance é igualmente beneficiada pelo processo de seleção de atributos descrito na seção 3.8.

Apesar de a independência de atributos não ser verificada para a maioria dos domínios de aplicação, na prática o Bayes ingênuo apresenta resultados satisfatórios. [Zhang 2004] atribui a surpreendente boa performance deste método ao fato de que a mera existência de dependências entre atributos não prejudicaria a classificação, mas sim o modo como

as dependências estão distribuídas ao longo das classes. Segundo o autor, desde que as dependências estejam distribuídas igualmente nas classes, não há problema em haver dependência forte entre dois atributos.

#### 3.9.4.1 Variantes do modelo Bayes ingênuo

As duas principais variantes de implementação do classificador bayesiano, denominadas de modelo *multinomial* e de *Bernoulli*, diferem fundamentalmente na maneira como os objetos são representados.

O primeiro modelo utiliza uma representação que considera informações espaciais sobre o objeto. Na classificação de documentos de texto, por exemplo, o modelo gera um atributo para cada posição do documento, que corresponde a um termo do vocabulário. Já o modelo de *Bernoulli* produz um indicador de presença ou ausência para cada possível atributo (no caso de texto, cada termo do vocabulário).

A escolha da representação de documentos adequada é uma decisão crítica no projeto de um classificador, visto que o próprio significado de um atributo depende da representação. No *multinomial*, um atributo pode assumir como valor qualquer termo do vocabulário, o que resulta numa representação do documento correspondente à sequência de termos do mesmo. Já para o modelo de *Bernoulli*, um atributo pode assumir apenas os valores 0 e 1, e a representação do documento é uma sequência de 0s e 1s do tamanho do vocabulário.

### 3.9.5 Agrupamento

Agrupamento (*clustering*, em inglês) é uma técnica de aprendizado de máquina não supervisionado. O algoritmo particiona a base de dados de forma a criar automaticamente grupos que reúnam usuários com comportamentos semelhantes.

A escolha do algoritmos a ser utilizadoo deve depender do tipo de dados disponível e nos objetivos específicos da aplicação. É aceitável experimentar diversos algoritmos no mesmo conjunto de dados porque a análise de grupos é geralmente utilizada como ferramenta exploratória ou descritiva, em contraste com testes estatísticos que são realizados para fins de confirmação ou inferência. Ou seja, geralmente não se quere provar ou disprovar uma hipótese pré-concebida; mas sim ver "o que os dados estão tentando nos dizer" [Kaufman and Rousseeuw 2005].

#### 3.9.5.1 Métodos partitivos

Os dados são classificados em k grupos que satisfazem os requisitos de uma partição: cada grupo contém no minimo um objeto e cada objeto pertence a exatamente um grupo;

(a) **K-Means**

Uma das técnicas mais populares de agrupamento é o *k-means*, que consiste basicamente nos seguintes passos:

1. Seleção de $k$ elementos considerados sementes;
2. Associação de cada elemento da base de dados com a semente mais próxima a ele;
3. Cálculo de novos pontos no espaço centrais para cada grupo, a partir dos elementos que o compõem.

O passo 3 é repetido até que não seja mais necessário calcular novos centróides. Desta forma, um sistema de recomendação pode realizar o agrupamento da base de dados de usuários e sugerir itens de acordo com o grupo em que o usuário se encontra ao final do processo de agrupamento. Devem ser recomendados os itens mais frequentes entre os outros usuários do grupo que este usuário específico não tenha.

(b) **K-Medoids**

Este é outro método partitivo, porém as partições giram em torno de medoids, que são os elementos que acumulam menor dissimilaridade com o restante dos objetos do grupo. É uma solução mais genérica do que o *K-Means* porque pode ser utilizada mesmo quando um centróide não pode ser definido (a depender do tipo das variáveis), já que se apoiam em coeficientes de dissimilaridades. Recomendado também quando há interesse em preservar os objetos representativos de cada grupo, úteis para fins de redução de dados ou caracterização.

(c) **Fuzzy**

Evita decisões estritas. (detalhar...)

(d) **Agrupamento em larga escala**

A distância entre pares de objetos de um grupo é geralmente armazenado em memória, alocando um espaço de memoria da ordem de $O(n^2)$.

Para aplicações com bases de dados grandes, o agrupamento é realizado numa amostra dos dados selecionada aleatoriamente e o restante dos dados é classificado de acordo com a promiximidade a cada cluster formado. Para melhores resultados, o processo deve ser realizado diversas vezes e a melhor solução de agrupamento deve ser mantida.

#### 3.9.5.2 Métodos hierárquicos

Podem ser aglomerativos ou divisivos. (detalhar...)

### 3.9.6 Apriori

A mineração de Dados, também referenciada como descoberta de conhecimento em bases de dados, é a área da ciência da computação destinada à descoberta de correlações e padrões frequentes num conjunto de dados. Informações extraídas de uma base de dados de transações de venda, por exemplo, têm alto valor para organizações que pretendem realizar processos de *marketing* guiados por informação – modelo denominado, em inglês, *market basket analysis*. Outros domínios de aplicação que também utilizam técnicas de mineração são: detecção de intrusão através da análise de *logs* de sistemas computacionais, pesquisas na área de saúde sobre a correlação entre doenças, sequenciamento de DNA etc [Hegland 2003].

Os padrões frequentes podem ser descritos por conjuntos de itens que ocorrem simultaneamente ou por implicações na forma $X \Rightarrow Y$, denominadas de *regras de associação*, sendo $X$ e $Y$ conjuntos de itens disjuntos ($X \cap Y = \emptyset$). *Suporte* e *confiança* são duas métricas para quantificar a força dos padrões de acordo com a sua representatividade no banco de dados de transações. O suporte de um conjunto de itens é a frequência com a qual ele ocorre numa base de dados. Para uma regra de associação $X \Rightarrow Y$, mede-se o suporte do conjunto de itens $X \cup Y$. A confiança de uma regra é medida pela frequência de $Y$ nos registros que contém $X$, representando o grau de co-ocorrência de $X$ e $Y$.

O *Apriori* é um algoritmo clássico para mineração de regras de associação sustentadas por medidas mínimas de suporte e confiança numa base de dados. Este problema é comumente decomposto em dois sub-problemas:

(1) Identificação de todos os conjuntos de itens que extrapolam um valor de suporte mínimo na base de dados (denominados de *conjuntos frequentes*).

(2) Produção de regras de associação a partir dos conjuntos frequentes, selecionando apenas as que satisfazem a condição de confiança mínima. Visto que as regras são partições binárias de conjuntos de itens, uma solução trivial para este problema é: para cada

subconjunto $S$ de um conjunto frequente $F$, gerar a regra $S \Rightarrow F - S$ e testar seu valor de confiança.

O *Apriori* foi o primeiro algoritmo a tratar do sub-problema *(1)*, que de fato é o mais desafiador, de forma mais eficiente. Uma solução ingênua para tal problema seria: listar todos os conjuntos candidatos (conjunto das partes do universo de itens) e selecionar os conjuntos frequentes a partir do cálculo de suporte para cada um. No entanto, esta é uma estratégia extremamente custosa visto que o conjunto das partes de um conjunto com $n$ elementos contém $2^n$ subconjuntos, inviabilizando o cálculo para domínios de aplicação com um universo de itens grande [Hegland 2003]. A figura 3.7 ilustra através de um diagrama de *Hasse* o conjunto das partes do universo de itens $U = \{a, b, c\}$.

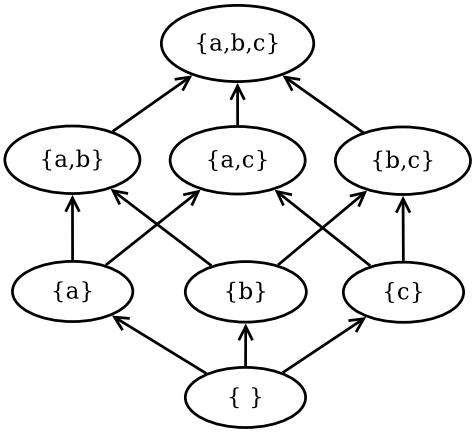


Figura 3.7: Conjunto das partes ilustrado por um diagrama de *Hasse*

A inovação do *Apriori* sobre a abordagem ingênua é a redução da quantidade de conjuntos candidatos pelo descarte de certos conjuntos que comprovadamente não são conjuntos frequentes. Desta forma o algoritmo consegue detectar todos os conjuntos frequentes sem a necessidade de calcular o suporte para todos os $2^n$ subconjuntos possíveis.

A descoberta de conjuntos frequentes acontece por níveis, como uma busca em largura no diagrama de *Hasse* começando pelos conjuntos unitários. Em vez de gerar os conjuntos candidatos a partir da base de dados, a cada nível da busca é feita uma combinação dos elementos para gerar os candidatos do nível seguinte. Neste ponto a solução se beneficia do seguinte princípio: qualquer subconjunto de um conjunto frequente também é um conjunto frequente. Portanto, só devem participar da nova combinação os elementos que apresentarem um suporte superior ao limite, pois um conjunto que não é frequente não será jamais subconjunto de um conjunto frequente [Agrawal and Srikant 1994].

A figura 3.8 ilustra a descoberta dos conjuntos frequentes em contraposição com o conjunto das partes do conjunto $U = \{a, b, c, d, e\}$. Neste exemplo, os subconjuntos $\{e\}$, $\{a, b\}$ e $\{b, d\}$ estão destacados por apresentarem suporte inferior ao limite. Consequentemente, todos os conjuntos dos quais estes são subconjuntos foram desconsiderados como conjuntos candidatos (nós com fundo cinza na figura). Portanto, apenas os nós com fundo branco teriam o suporte calculado.

A introdução do *Apriori* representou um marco para o desenvolvimento de soluções para mineração de dados, motivando o surgimento de inúmeras variantes baseadas no mesmo princípio. Entre elas, surgiram algumas propostas específicas para situações onde os dados têm características adicionais conhecidas como, por exemplo, base de dados particionada, dados que satisfazem à determinadas restrições ou que fazem parte de uma taxonomia conhecida [Hegland 2003].

Apesar de apresentar um processo inovador para geração de regras de associação, o *Apriori* também apresenta fraquezas, sendo a principal delas a necessidade de percorrer a base de

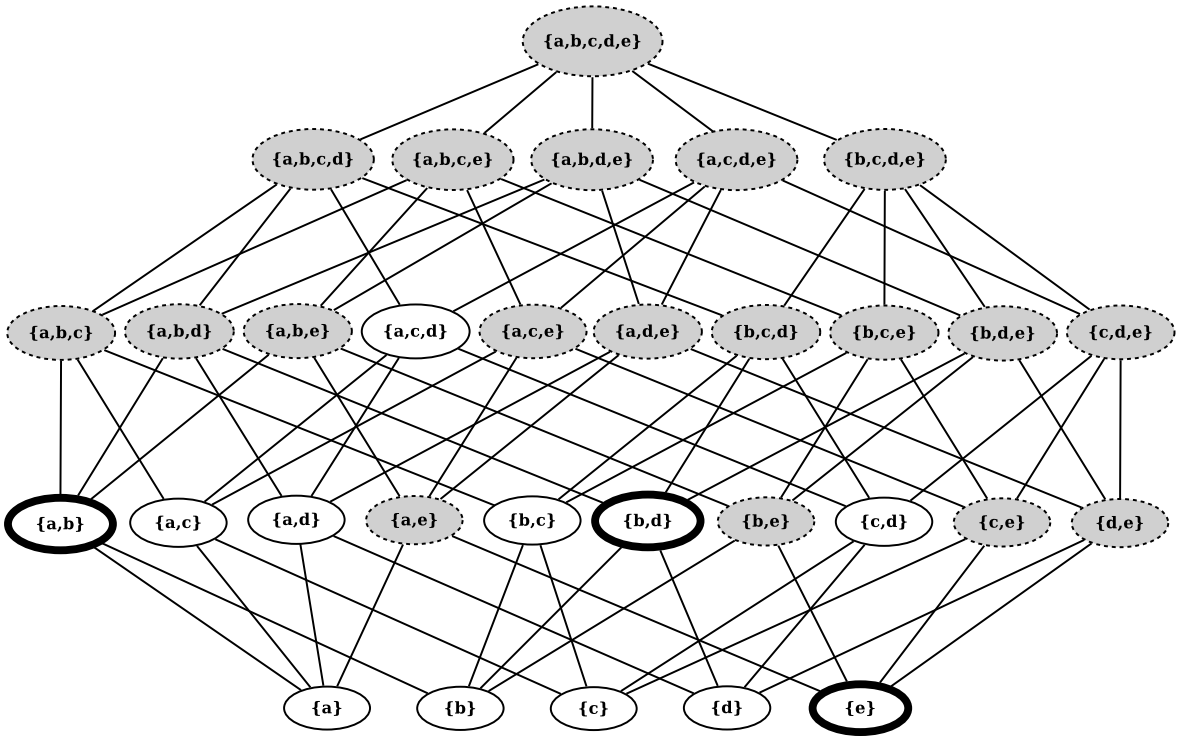


Figura 3.8: Geração de conjuntos candidatos pelo algoritmo Apriori

dados múltiplas vezes para cálculo de suporte e confiança dos conjuntos de itens. Algumas soluções alternativas fazem uso de estruturas de dados auxiliares para armazenar informações extraídas da base de dados numa única passagem, evitando desta forma repetidos acessos à mesma. Árvores de prefixos, árvores lexicográficas e matrizes binárias são algumas dessas estruturas [Kotsiantis and Kanellopoulos 2006].

## 3.10 Avaliação de recomendadores

A avaliação de sistemas de recomendação não é uma tarefa trivial, principalmente porque não há consenso sobre quais atributos devem ser observados e quais métricas devem ser adotadas para cada atributo [Herlocker et al. 2004]. Ademais, diferentes estratégias podem funcionar melhor ou pior de acordo com o domínio da aplicação e as propriedades dos dados. Por exemplo, algoritmos projetados especificamente para conjuntos de dados com um número muito maior de usuários do que de itens podem se mostrar inapropriados em domínios onde há muito mais itens do que usuários. Recomenda-se então que o processo de avaliação tenha início com a compreensão das ações para as quais o sistema foi projetado (ver seção 3.5), como guia para as decisões metodológicas ao longo dos experimentos.

### 3.10.1 Seleção dos dados

A escolha do conjunto de dados adequado é fator chave para uma investigação consistente. Neste aspecto, as avaliações de recomendadores podem ser caracterizadas como (a) análises *offline*, que utilizam bases de dados previamente coletadas e (b) experimentos "ao vivo", realizados diretamente com usuários, seja num ambiente controlado (laboratório) ou em campo [Herlocker et al. 2004].

Análises *offline* geralmente são objetivas, com foco na acurácia das predições e performance das soluções [Vozalis and Margaritis 2003]. Inicialmente os dados são particionados em porções de treinamento e de testes. Utiliza-se como base os dados de treinamento para

prever recomendações para itens da porção de testes. Em seguida é feita a análise comparativa entre os resultados obtidos e os esperados. Algumas métricas comumente utilizadas na fase de análise serão apresentadas na seção 3.10.2. No entanto, tais análises são prejudicadas em conjuntos de dados esparsos. Não se pode, por exemplo, avaliar a exatidão da recomendação de um item para um usuário se não existe uma avaliação prévia do usuário para tal item.

Por outro lado, nos experimentos "ao vivo" os recomendadores são disponibilizados para uma comunidade de usuários, cujas avaliações são coletadas na medida em que são produzidas. Neste caso, além de análises objetivas como a acurácia e performance das soluções, pode-se avaliar fatores comportamentais como a performance, participação e satisfação dos usuários. A esparsidade dos dados tem efeito menor neste tipo de experimento, visto que o usuário está disponível para avaliar se os itens recomendados são ou não relevantes.

Quando não existem dados previamente disponíveis ou quando não são adequados para o domínio ou a ação principal do sistema a ser avaliado, pode-se ainda optar pelo uso de dados sintéticos. O uso de dados artificiais é aceitável em fases preliminares de testes, porém, tecer conclusões comparativas é arriscado visto que os dados produzidos podem se ajustar melhor para uma estratégia do que para outras [Herlocker et al. 2004].

### 3.10.2 Métricas

A tabela 3.9 apresenta métricas utilizadas para avaliação de sistemas de recomendação.

As métricas de acurácia medem o quanto as estimativas de relevância previstas pelo sistema se aproximam da real. Acurácia de classificação está relacionada com a frequência com a qual o sistema faz classificações corretas acerca da relevância dos itens, enquanto que a acurácia de predição pondera as diferenças entre as pontuações de relevância prevista e a real.

Além de medidas de acurácia, são apresentados outras métricas para mensurar qualidades que proporcionam maior grau de satisfação do usuário ao utilizar um sistema de recomendação.

Tabela 3.9: Métricas para avaliação de sistemas recomendadores

| ***Métrica*** | ***Descrição*** | ***Fórmula*** | ***Categoria*** |
|---|---|---|---|
| Precisão | Proporção de itens relevantes entre os selecionados como tal | $P = \frac{N_{\text{relevantes selecionados}}}{N_{\text{selecionados}}}$ | Acurácia de classificação |
| Recuperação | Proporção de itens selecionados entre todos os relevantes | $R = \frac{N_{\text{relevantes selecionados}}}{N_{\text{relevantes}}}$ | |
| Medida $F_1$ | Combinação de $P$ e $R$ numa mesma medida | $F_1 = \frac{2PR}{P+R}$ | |
| Curva *ROC* | Mede o poder de distinção entre itens relevantes e irrelevantes | Análise gráfica | |
| Erro absoluto médio *(MAE)* | Desvio absoluto médio entre pontuações previstas e reais | $\lvert\overline{E}\rvert = \frac{\sum_{i=1}^{N} \lvert p_i - r_i \rvert}{N}$ | Acurácia de predição |
| Erro quadrático médio | Desvio quadrático médio entre pontuações previstas e reais | $\lvert\overline{E}\rvert = \frac{\sum_{i=1}^{N} \lvert p_i - r_i \rvert^2}{N}$ | |
| Cobertura | Proporção de itens passíveis de serem recomendados entre todos os disponíveis | $R = \frac{N_{\text{passíveis de recomendação}}}{N}$ | Além de acurácia |
| Curva de aprendizado | Taxa de aprendizado dos algoritmos na análise dos dados de exemplo | $A = \frac{\Delta_{\text{acurácia}}}{\Delta_T}$ | |
| Novidade e surpresa | Qualidade do sistema de produzir recomendações não óbvias | Avaliada pelo usuário | |

*Capítulo 4*

# Trabalhos correlatos

No presente capítulo serão apresentadas iniciativas de desenvolvimento por parte da comunidade FOSS e trabalhos acadêmicos que se relacionam com a proposta deste trabalho. Algumas dessas iniciativas foram utilizadas como base de desenvolvimento, principalmente as relativas ao projeto Debian, enquanto outras serviram como fontes de referências e inspiração.

## 4.1 Iniciativas FOSS

### 4.1.1 Dados sobre pacotes Debian

O projeto Debian tem se destacado no universo das distribuições por suas iniciativas pioneiras no campo de gerenciamento de aplicações [Zini 2011b]. Diante da complexa e crescente estrutura do projeto, observa-se um esforço por parte dos desenvolvedores, principalmente da equipe responsável pelo controle de qualidade[1], de reunir, organizar e disponibilizar as informações ou meta-dados concernentes a esta estrutura [Nussbaum and Zacchiroli 2010].

A seguir serão detalhadas algumas destas iniciativas, que curiosamente foram desenvolvidas inicialmente num contexto extra-oficial e, ao passo que se mostraram úteis e eficientes, foram absorvidas pela comunidade de usuários e desenvolvedores.

#### 4.1.1.1 Debtags

*Debtags* é um esquema de classificação idealizado pelo desenvolvedor Debian Enrico Zini como uma maneira de categorizar pacotes alternativa às seções [Zini 2011a]. A principal motivação desta iniciativa foi a impossibilidade de relacionar pacotes a múltiplas seções. Um navegador web, por exemplo, não poderia ser categorizado como *network* e *web* simultaneamente. O uso de *tags* (em português, rótulos) possibilitaria a criação de uma coleção estruturada de metadados que poderia ser utilizada para implementar métodos mais avançados do que os existentes para apresentação, busca e manutenção do repositório de pacotes Debian.

O proposta foi apresentada na Debconf5 e foi paulatinamente sendo adotada por desenvolvedores em suas atividades, sendo atualmente utilizada como base de inúmeras ferramentas no Debian, tendo atingido a marca de 45% de pacotes categorizados[2].

Utilizando *Debtags*, os pacotes podem ser caracterizados por múltiplos atributos, que são (propositalmente) definidos num momento posterior à concepção do pacote. Dado que a base de tags é mantida de forma independente ao repositório, as modificações ao longo do tempo não trazem impacto algum ao desenvolvimento de pacotes. A atribuição de tags a pacotes é

[1] http://qa.debian.org
[2] Consulta realizada em junho de 2011

```
acx100-source: admin::kernel, implemented-in::c, role::source,
use::driver

adduser: admin::user-management, implemented-in::perl,
interface::commandline, role::program, scope::utility

apache2: implemented-in::c, interface::daemon, network::server,
network::service, protocol::http, protocol::ipv6, role::metapackage,
role::program, suite::apache, web::server, works-with-format::html,
works-with::text

apbs: field::chemistry, implemented-in::c, interface::commandline,
role::program, scope::utility

apcalc: field::mathematics, interface::shell, interface::text-mode,
role::program, scope::utility
```

Figura 4.1: Excerto da base do Debtags

realizada por colaboradores através do website do projeto[3] e revisada manualmente antes de ser incorporada à base de dados.

A base de dados é armazenada num arquivo texto no formato da figura 4.1.1.1. O conjunto de tags disponível faz parte de um vocabulário controlado[4], que também recebe contribuições de colaboradores. O esquema é estruturado para permitir a classificação por diferentes pontos de vista, que caracterizam as facetas.

Ao indicar novas tags para um pacote, o usuário é surpreendido com sugestões de outras tags que geralmente são aplicadas em conjunto com as já selecionadas. Esta é uma aplicação de recomendação com base em regras de associação descobertas a partir de análise da base de dados de tags. O algoritmo utilizado para produção das regras é o Apriori, descrito na seção 3.9.6.

O *Debtags* é uma poderosa ferramenta para a construção de estratégias de recomendação de pacotes baseadas em conteúdo. É fato que o conteúdo acerca de pacotes pode ser expresso em termos de atributos extraídos dos próprios pacotes, porém, a caracterização por meio de tags já fornece uma caracterização possível de ser utilizada e a custo baixo.

#### 4.1.1.2 Apt-Xapian-Index

#### 4.1.1.3 Popcon

O *Popcon (Popularity Contest)* é o concurso de popularidade entre pacotes Debian, criado pelo desenvolvedor Avery Pennarun em 1998 com o propósito inicial de auxiliar a escolha dos pacotes que fazem parte do primeiro CD de instalação[5] (os mais populares são selecionados). Atualmente o repositório de pacotes Debian pode ser obtido em 52 imagens de CDs ou 8 de DVD. Dado que comumente obtem-se apenas a primeira imagem por *download*, a A priorização de pacotes populares na primeira imagem tende a contribuir para a satisfação dos usuários, dado que geralmente apenas este é obtida por *download* e os demais pacotes são obtido diretamente do repositório por meio de uma conexão de rede.

[3] http://debtags.alioth.debian.org/todo.html
[4] http://debtags.alioth.debian.org/vocabulary/
[5] http://lists.debian.org/debian-devel-announce/2004/03/msg00009.html

Na instalação de um sistema Debian o usuário é convidado a participar do concurso. Se aceitar, o software cliente do *popcon* é instalado e envia periodicamente a lista de pacotes daquele sistema para um servidor central, indicando também quais deles foram utilizados recentemente. A figura 4.2 apresenta um exemplo de submissão do popcon. A primeira linha contém um hash que identifica um sistema unicamente no concurso (submissões consecutivas de um mesmo usuários são sobrescritas). Cada linha seguinte representa um pacote instalado no sistema, no formato `<atime> <ctime> <package-name> <mru-program> <tag>`, detalhado na tabela 4.1. Os campos relativos a tempo são indicados no formato *Unix time_t*[6].

```
POPULARITY-CONTEST-0 TIME:914183330 ID:b92a5fc1809d8a95a12eb3a3c84166dd
914183333 909868335 grep /bin/fgrep
914183333 909868280 findutils /usr/bin/find
914183330 909885698 dpkg-awk /usr/bin/dpkg-awk
914183330 909868577 gawk /usr/bin/gawk
[...]
[...]
[...]
END-POPULARITY-CONTEST-0 TIME:914183335
```

Figura 4.2: Exemplo de submissão do popcon

| **Campo** | **Descrição** |
|---|---|
| <package-name> | nome do pacote Debian que contém <mru-program> |
| <mru-program> | Programa, biblioteca ou cabeçalho contido no pacote que foi utilizado mais recentemente. |
| <atime> | Tempo de acesso do <mru-program> no disco, atualizado pelo kernel cada vez que o arquivo é aberto. |
| <ctime> | Tempo de criação do <mru-program> no disco, definido no momento de instalação do pacote. |
| <tag> | RECENT-CTIME: indica que <atime> é muito próximo de <ctime>, geralmente quando o pacote foi recentemente instalado ou atualizado; OLD: <atime> é anterior a um mês atrás, portanto o pacote não foi usado no último mês; NOFILES: o pacote não contém programas, portanto <atime>, <ctime> e <mru-program> são inválidos. |

Tabela 4.1: Descrição do formato de uma submissão popcon

As submissões recebidas são processadas diariamente e as estatísticas geradas são disponibilizadas no website do projeto[7]. Os dados "crus" (listas de pacotes antes de serem processadas) já foram utilizados anteriormente para fins de mineração de dados, como descrito nas seções 4.1.6 e **??**, porém ambos os trabalhos foram descontinuados.

A informação sobre o uso dos pacotes também tem sido utilizada como guia para o time de qualidade acerca de quais pacotes merecem atenção especial. Os times de trandução igualmente têm considerado estes dados para ordenar a lista de prioridades de acordo com a popularidade. Por outro lado, pacotes "problemáticos" (p. ex. pacotes com muitos *bugs RC*, órfãos ou com mantenedor em MIA) que apresentam baixa popularidade tendem a perder a

[6] Quantidade de segundos desde meia-noite de primeiro de janeiro de 1970 no horário GMT.

[7] http://popcon.debian.org

simpatia dos desenvolvedores, sendo esta um dos parâmetros para a decisão de remover um pacote do repositório (*low-popcon*).

Essa abordagem, no entanto, tem sido duramente criticada. Segundo Joey Hess[8], uma vantagem do Debian é justamente que não apenas programas populares são empacotados, mas os incomuns e específicos de um nicho de usuário também costumam estar disponíveis em pacotes. E de fato o *popcon* não mede o benefício de pacotes poucos usados estarem disponíveis no repositório, prontos para serem usados. Enquanto que ao remover pacotes que aparentemente não são populares, corre-se o risco de transformar o Debian numa distribuição homogênea, submetida à "tirania da maioria".

Existem ainda questões relativas a (1) representatividade desses dados, visto que alguns perfis de usuários dificilmente participam do concurso (p. ex. sistemas embarcados); e (2) acurácia de informações temporais, dado que `<atime>` e `<ctime>` podem ser inconsistentes caso o sistema de arquivos tenha sido montado com a opção `noatime`.

Todas essas ressalvas devem ser consideradas quando pretende-se utilizar os dados do *popcon*. No entanto, desde que as informações sejam manejadas de forma consciente e responsável, acredita-se que valiosas correlações possam ser reveladas após uma série de análises.

#### 4.1.1.4 UDD

Acrônimo para *Ultimate Debian Database*, o UDD[9] é uma iniciativa recente do time de qualidade criada com o intuito de reunir informações de diversos aspectos do Debian numa base de dados única [Nussbaum and Zacchiroli 2010].

O fluxo de dados do UDD é apresentado na figura 4.3. Existe um coletor para cada tipo de fonte de dados (p. ex. o sistema de acompanhamento de bugs[10] que implementa uma interface comum e esconde a complexidade e especificidade de acessar cada fonte de dados. Existe um processo central no UDD que invoca os coletores periodicamente, provocando a inserção dos dados na base única.

A pricipal motivação para o desenvolvimento do UDD foi auxiliar o time de QA em suas atividades, além de facilitar a colaboração com distribuições derivadas do Debian. Apesar de ser possível, dificilmente usuários consultariam esta base para tomar decisões acerca de que pacotes utilizar, visto que os dados armazenados no UDD geralmente são acessíveis por outros meios. No entanto, para fins de mineração de dados ou para o desenvolvimento de um recomendador automático, a possibilidade de acesso a dados de tamanha heterogeneidade numa fonte de dados única é uma grande vantagem.

### 4.1.2 Dados sobre pacotes *rpm*

Apesar de não estarem diretamente relacionadas com o desenvolvimento deste trabalho, a seguir são descritos alguns esforços recentes para coleta de dados sobre pacotes *rpm*.

#### 4.1.2.1 Sophie

Ferramenta web[11] que permite a busca e análise do conteúdo de pacotes rpms de múltiplas distribuições. Provê também uma interface XML-RPC que possibilita a criação de outras ferramentas utilizando Sophie como base.

[8] http://kitenet.net/~joey/blog/entry/the_popcon_problem/
[9] http://udd.debian.org
[10] http://bugs.debian.org
[11] http://sophie.zarb.org/

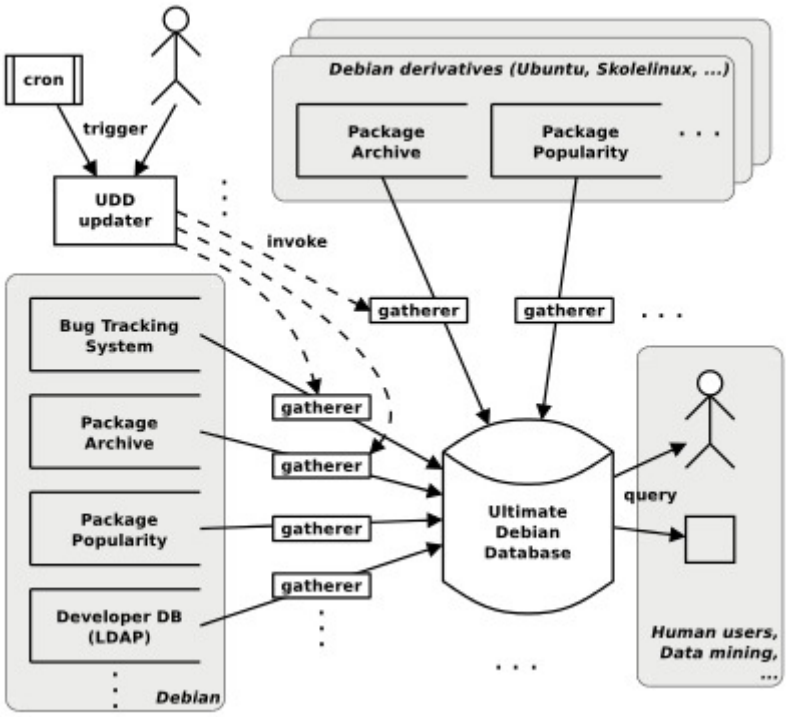


Figura 4.3: Fluxo de dados no UDD [Nussbaum and Zacchiroli 2010]

#### 4.1.2.2 mageia-app-db

Base de dados de aplicativos idealizada pelos desenvolvedores da distribuição Mageia[12], que atualmente também reúne esforços de colaboradores do openSUSE, Mandriva e Fedora. Ainda em fase de desenvolvimento, a ferramenta tem como principal objetivo facilitar a interação entre usuários, testadores e empacotadores de software. Além de acessar a coleção de aplicativos disponíveis na distribuição, o usuário será capaz de requisitar funcionalidades ou correções aos responsáveis pelos pacotes, além de acompanhar seus pedidos e receber notificações automáticas. Produção de recomendações de pacotes também estão previstas como funcionalidades do sistema.

### 4.1.3 Servidor para avaliação de aplicativos

Em inglês, *Ratings and Reviews Server*[13], é um serviço em desenvolvimento baseado em *django*[14] para armazenar avaliações de usuários sobre aplicativos, com pontuação e comentários. Quando estiver em produção será uma poderosa base de dados para a produção de recomendações sobre aplicativos, alternativa à base do popcon, que não foi projetada para fins de recomendação e se restringe a dados binários (um usuário possue um pacote instalado ou não).

### 4.1.4 Open Collaboration Service

Especificação aberta projetada para dar suporte a colaboração entre serviços web, permitindo o armazendamento de avaliações de usuários e outras informações do domínio de aplicação.

[12] http://mageia-app-db.tuxette.fr/projects/mageia-app-db/wiki
[13] http://launchpad.net/rnr-server
[14] https://www.djangoproject.com/

### 4.1.5 AppStream

Grande parte das distribuições GNU/Linux têm investido no desenvolvimento de interfaces para facilitar o gerenciamento de aplicativos e a forma como se obtém informações sobre os mesmos. Entre os dias 18 e 21 de janeiro 2011 aconteceu a primeira reunião sobre a temática com a presença de desenvolvedores de distribuições variadas (*Cross-distribution Meeting on Application Installer*). O encontro teve como principais objetivos a definição de padrões entre os diferentes projetos no que diz respeito a: procedimentos de instalação de aplicações; metadados associados aos pacotes; o modo como tais informações devem ser geradas e armazenadas; protocolo para manutenção de metadados dinâmicos; e a definição de quais metadados devem ser compartilhados entre as distribuições, em detrimento de outros considerados específicos de cada projeto [Freedesktop 2011].

### 4.1.6 Anapop/PopSuggest

O Anapop ou Popsuggest foi desenvolvido em 2007 por Enrico Zini*PopSuggest*[15] como uma ilustração das possibilidades de uso dos dados coletados pelo popcon. A ferramenta utilizava técnicas de recuperação da informação sobre a base de submissões do popcon para inferir quais pacotes diferenciavam determinado usuário dos demais, além de sugerir pacotes que usuários similares àquele tinham instalados.

## 4.2 Trabalhos acadêmicos

Nos últimos anos foram publicados, no Brasil e no exterior, inúmeros trabalhos nas áreas de mineração de dados e recuperação da informação aplicadas aos mais diversos domínios de aplicação. Nesta seção nos restringimos a comentar alguns deles que foram utilizados como referência, por tratarem especificamente do problema de recomendação, no domínio de aplicativos ou não.

### 4.2.1 *"Uma aplicação em sistemas de recomendação: sistema de recomendação para pacotes GNU/Linux"*

Trabalho final de graduação apresentado em junho de 2007 na Universidade Federal do Rio Grande do Sul [Pereira 2007]. A ferramenta foi desenvolvida como prova de conceito, não sendo porém integrado aos serviços da distribuição. Foram realizados experimentos de eficácia com cerca de 30 usuários, por meio do qual as diversas técnicas implementadas foram comparadas.

### 4.2.2 *"Projeto e criação de um sistema para produção de sugestões personalizadas para o Instalador Debian"*

Dissertação de mestrado[16] defendida em agosto de 2007 na Universität Paderborn, na Alemanha [Schröder 2007]. O trabalho foi apresentado na Conferência Internacional de Desenvolvedores Debian (Denconf7[17]), ainda em fase de desenvolvimento. A implementação era focada em técnicas de mineração de dados na base do popcon para produção de regras de associação.

[15] http://www.enricozini.org/2007/debtags/popcon-play/
[16] Tradução do título original do trabalho no idioma alemão.
[17] https://penta.debconf.org/~joerg/events/83.en.html

### 4.2.3 *"Uma plataforma de serviços de recomendação para bibliotecas digitais"*

Dissertação de mestrado defendida em Março de 2008 na Universidade Estadual de Campinas [Pedronette 2008]. O trabalho propõe uma plataforma de à produção de recomendações dirigida a serviços de bibliotecas digitais. Os experimentos realizados tiveram enfoque na interoperabilidade dos serviços, não havendo porém comparações entre estratégias diferenciadas de recomendação.

### 4.2.4 *"Arcabouço genérico baseado em técnicas de agrupamento para sistemas de recomendação"*

Dissertação de mestrado defendida em outubro 2010 na Universidade Estadual de Campinas que propõe um modelo genérico baseado em técnicas de agrupamento de dados para a construção de sistemas de recomendação [Pannagio 2010]. Um protótipo de recomendador foi implementado para validação da proposta e diversos experimentos foram realizados para fins de comparação de eficácia e eficiência entre técnicas de agrupamento distintas.

### 4.2.5 *"Uma arquitetura de personalização de conteúdo baseada em anotações do usuário"*

Tese de doutorado defendida em Março de 2011 no Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC-USP) [Manzato 2011]. O trabalho propõe uma arquitetura para serviços de personalização por meio da indexação automatizada de metadados extraídos de anotações colaborativas de usuários de um sistema multimídia. Como prova de conceito foram implementados dois sistemas recomendadores de vídeos que foram comparados por meio de métricas propostas na literatura.

*Capítulo 5*

# App-Recommender

Este capítulo apresenta o *AppRecommender* como proposta de um recomendador de aplicativos GNU/Linux.

## 5.1 Escolha da plataforma

A distribuição escolhida como base para o desenvolvimento deste trabalho foi o Debian GNU/Linux. No entanto, a independência de plataforma foi sempre levada em consideração na fase de desenvolvimento, com o intuito de que os resultados sejam facilmente adaptáveis para outros contextos. As seguir estão descritos os critérios que pautaram esta escolha.

1. **Esquema consistente de distribuição de aplicativos.** O gerenciamento de pacotes em sistemas Debian GNU/Linux é realizado através do *APT*, cujas funcionalidades foram apresentadas na seção 2.3.1. Apesar de atualmente outras distribuições oferecerem ferramentas similares, o *APT* é certamente uma das mais maduras, sendo inclusive apontada por alguns desenvolvedores como razão da explosão no surgimento de distribuições derivadas do Debian (herdeiras do esquema).

2. **Disponibilidade de dados estatísticos.** A base de dados do *Popcon* (seção 4.1.1.3) ultrapassou a marca de 100.000 colaboradores[1] em fevereiro de 2011. É a maior coleção de dados disponível atualmente sobre a utilização de pacotes Debian, compondo uma importante fonte de informação para a realização de estratégias colaborativas de recomendação.

3. **Possibilidade de integração dos resultados do trabalho.** Segundo o *contrato social Debian*[2], o desenvolvimento do projeto é guiado pelas necessidades dos usuários e da comunidade. Portanto, as iniciativas de colaboradores individuais, sejam eles desenvolvedores oficiais ou não, serão igualmente consideradas e passarão a fazer parte da distribuição se seguirem os princípios do projeto e forem considerados úteis para a comunidade.

4. **Popularidade.** O Debian é um projeto de destaque no ecossistema do software livre. Desde o lançamento da primeira versão de sua distribuição, em 1993, o projeto cresceu bastante em termos de componentes de software (atualmente provê mais de 25.000 pacotes), colaboradores e usuários. A *Distrowatch*, que tem 323 distribuições

[1] http://lists.alioth.debian.org/pipermail/popcon-developers/2011-February/001913.html
[2] http://www.debian.org/social_contract.pt.html

ativas em sua base de dados[3], classifica o Debian GNU/Linux entre as 10 distribuições mais populares[4]. O Debian aparece na quinta posição em suas estatísticas de páginas visitadas[5]. Já o *Linux Counter*[6] apresenta o Debian como a segunda distribuição mais popular entre as máquinas cadastradas que rodam o kernel Linux (16%), ficando atrás apenas do Ubuntu[7] (24%), que é uma distribuição derivada do Debian. Nas pesquisas da *W3Techs* sobre tecnologias para serviços web[8], o Debian aparece em segundo lugar, estando presente em 27% dos servidores. Na primeira posição está o CentOS com 31%.

De maneira geral, quando o projeto Debian é mencionado trata-se não somente do sistema operacional, mas de toda a infra-estrutura de desenvolvimento e coordenação que dá suporte ao trabalho de cerca de 900 desenvolvedores oficiais[9], além de outros milhares de colaboradores ao redor do globo. O trabalho é realizado de forma colaborativa, afinado pelo objetivo comum de produzir e disponibilizar livremente um sistema operacional de qualidade para seus usuários [Jackson and Schwarz 1998]. A interação entre os desenvolvedores acontece majoritariamente através da Internet, por meio de canais IRC e listas de discussão públicas. Não existe uma entidade formal ou qualquer tipo de organização que concentre, coordene ou defina as atividades do projeto. O que observa-se é um modelo de governança consolidado que emergiu naturalmente ao longo de sua história [O'Mahony and Ferraro 2007].

## 5.2 Caracterização do problema

Este trabalho tem como proposta principal o desenvolvimento de soluções para o problema da recomendação no contexto de componentes de software, em especial no âmbito de distribuições GNU/Linux. Neste cenário, os aplicativos são modelados como itens e os usuários da distribuição como clientes do recomendador.

Embora já faça parte da pauta de discussões entre desenvolvedores um esquema de pontuação de pacotes como medida de avaliação pessoal e comentários dos usuários, não há previsão para tal solução de fato ser implementada [Freedesktop 2011]. No entanto, a presença de um componente no sistema do usuário pode ser considerado como indicativo de relevância. A pontuação neste caso é binária – um item pode ser relevante ou irrelevante – e o processo de recomendação é caracterizado da seguinte maneira: dada a lista de pacotes instalados no sistema de determinado usuário (como representação de sua identidade), o recomendador deve retornar uma lista de pacotes sugeridos, que representam aplicativos de potencial interesse para tal usuário. Desta forma, a ação principal do sistema será *encontrar itens relevantes* (ver seção 3.5).

As recomendações devem ser produzidas a partir do comportamento do usuário. Neste trabalho não serão consideradas por exemplo informações registradas no BTS, PTS ou em listas de discussão. A computação será principalmente realizada com base em dados do *Popcon* (listas de pacotes de milhares de sistemas em produção), e do *Debtags* e *UDD* como fonte de metadados sobre os pacotes (atributos). A utilização de dados demográficos também está sendo considerada. Por exemplo, a declaração explícita por parte do usuário de que não tem interesse por determinado nicho de aplicativos eliminaria de antemão uma série de pacotes que a princípio seriam considerados. De certa forma, o uso de perfis pode possibilitar a realização de uma seleção de atributos específica para cada usuário.

[3] Consulta realizada em 24 de janeiro de 2011.
[4] http://distrowatch.com/dwres.php?resource=major
[5] http://distrowatch.com/stats.php?section=popularity
[6] http://counter.li.org/reports/machines.php
[7] http://www.ubuntu.com/community/ubuntu-and-debian
[8] http://w3techs.com/technologies/history_details/os-linux
[9] http://www.perrier.eu.org/weblog/2010/08/07#devel-countries-2010

### 5.2.1 Considerações acerca da dependência entre pacotes

Uma característica peculiar da recomendação neste contexto é que, diferentemente de outros domínios nos quais os itens não se relacionam entre si, os componentes de software objeto desta pesquisa podem declarar requisitos em seu conteúdo (dependência, sugestão, recomendação, conflito, substituição, quebra etc). Requisitos positivos representam relações de dependência, enquanto os negativos caracterizam relações de conflito entre os pacotes [Abate et al. 2009]. Por exemplo, se um componente $a$ depende de $b$, significa que $b$ deve ser instalado no sistema para que $a$ funcione como previsto. Por outro lado, se $a$ conflita com $c$, a instalação de ambos os aplicativos pode provocar um comportamento anômalo ou até comprometer o funcionamento de todo o sistema.

As relações entre componentes também devem ser consideradas pelo recomendador de pacotes. Intuitivamente, numa mesma recomendação não faz sentido sugerir pacotes dependentes, visto que ao aceitar a recomendação de um determinado componente e prosseguir com a instalação do mesmo, o usuário já está implicitamente aceitando a recomendação de todas as suas dependências. No entanto, um recomendação que contenha por exemplo os pacotes $a$ e $b$, dado que $a$ depende de $b$, se $a$ obtiver uma alta estimativa de relevância em virtude de seus atributos (por estratégias baseadas em conteúdo), esta recomendação não deve ser descartada. No caso de pacotes "guarda-chuva", que possuem muitas dependências, é comum que o usuário se interesse apenas por uma de suas dependências.

No que diz respeito à mineração de conhecimento a partir de uma base de dados, um ponto importante a considerar é que o fato de pacotes com alguma relação de dependência estarem presentes concomitantemente em grande parte dos sistemas não é uma coincidência, e sim uma consequência direta da dependência. Por outro lado, teriam grande valor as associações que relacionassem componentes que por definição não apresentam relação alguma. Neste cenário, novos graus de relacionamento poderiam ser estabelecidos, baseados não na dependência mas na colaboração entre os componentes.

### 5.2.2 Considerações acerca da necessidade dos usuários

Um sistema de recomendação de pacotes tem como principal função sugerir novos pacotes a partir de escolhas anteriores do usuário, assumindo que os recomendados são os que melhor satisfazem às suas necessidades. Todavia, o conceito de *pacote* já é uma abstração, e nem sempre a necessidade de um usuário pode ser mapeada diretamente na instalação de pacotes específicos.

A necessidade do usuário diz respeito às funcionalidades que os aplicativos oferecem. Visto que aplicativos diferentes podem executar a mesma função, (resguardadas as devidas peculiaridades), o mais apropriado seria representar a necessidade ou desejo do usuário (analogamente, sua identidade) como um conjunto de funcionalidades, em vez de um conjunto de pacotes específicos.

Algumas funcionalidades de pacotes podem ser extraídas a partir da lista de pacotes virtuais[10]. Este conceito foi criado especialmente para situações em que diversos pacotes diferentes oferecem um conjunto de funcionalidades semelhantes. Pacotes virtuais não existem fisicamente no repositório, são apenas mencionados no campo *Provides* da definição de outros pacotes ("concretos"). Desta forma, quando uma dependência se refere a um pacote virtual, ela pode ser satisfeita com a instalação de qualquer pacote que provê o mesmo.

No entanto, a lista de pacotes virtuais é controlada e relativamente pequena. Acredita-se porém que uma análise detalhada da base do *Debtags* pode revelar novas funcionalidades como abstrações de conjuntos de *tags*, que poderão eventualmente ser utilizadas para refinar o cálculo das recomendações.

[10] http://www.debian.org/doc/packaging-manuals/virtual-package-names-list.txt

## 5.3 Proposta de Solução: AppRecommender

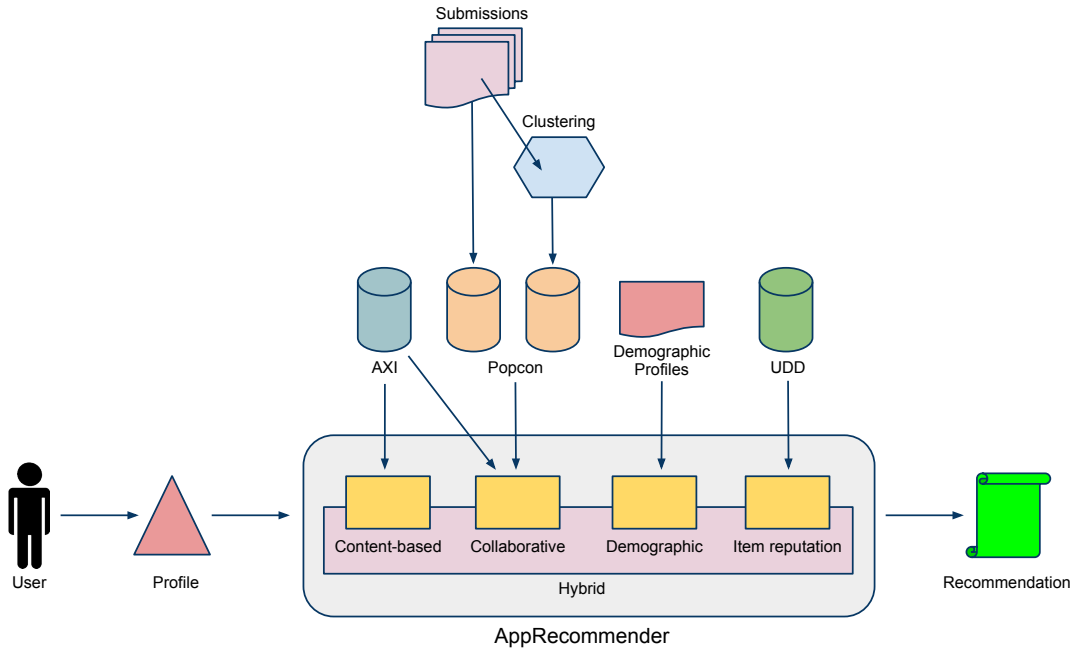


Figura 5.1: Fluxo de dados no AppRecommender

## 5.4 Coleta de dados

Os dados estatísticos disponíveis publicamente no site do *Popcon* não preservam os relacionamentos usuário-item, essenciais para a utilização de estratégias colaborativas. Por questões relativas à privacidade[11], as listas de pacotes originais submetidas pelos usuários não são publicadas, apenas as estatísticas geradas e um resumo de todas as submissões são disponibilizados diariamente. No entanto, este trabalho é desenvolvido com o apoio de desenvolvedores oficiais Debian, o que possibilitará o acesso aos dados necessários.

No entanto, as bases de dados do *UDD* e do *Debtags* são públicas, não havendo impedimento algum para acessá-las. Por fim, perfis de usuários podem ser formados com base no preenchimento facultativo de um formulário antes da utilização do recomendador.

## 5.5 Seleção de atributos

A seleção de atributos geralmente proporciona ganhos na eficiência e qualidade das recomendações, na medida em que diminui o ruído e o montante de dados a ser considerado. Além da aplicação de métodos clássicos de seleção, apresentados na seção 3.8, existem peculiaridades do domínio de componentes de software que podem ampliar tais benefícios se consideradas.

Seja qual for o sistema operacional ou distribuição GNU/Linux, existe um conjunto de componentes que fazem parte da instalação padrão, selecionados pela equipe de desenvolvimento. Considerando que os usuários do recomendador utilizam um sistema funcional, existem dois casos a considerar: (1) todo o conjunto de componentes da instalação padrão está instalado no sistema e (2) alguns componentes não estão presentes porque foram

[11] http://popcon.debian.org/FAQ

propositalmente removidos pelo usuário. Em ambos os casos a recomendação de tais pacotes certamente não interessaria ao usuário, portanto acredita-se que todos eles possam ser desconsiderados sem prejuízo para a recomendação.

Os dados coletados pelo *Popcon* também possuem informações temporais. A data de instalação do pacote no sistema é obtida através do atributo *ctime* do arquivo, que indica a data de sua criação no sistema de arquivos, enquanto a data de última utilização é indicada pelo *atime*, com a data do último acesso. Apesar destes dados não serem seguramente confiáveis[12], a possibilidade de uso dos mesmo será investigada. Seria interessante, por exemplo, atribuir pesos diferentes para pacotes que são usados com muita frequência pelo usuário em detrimento de outros que foram acessados pela última vez logo após serem instalados.

## 5.6 Estratégias implementadas

Optou-se por utilizar técnicas de busca como base das soluções, mesmo para estratégias herdadas de classificação, dado que a construção prévia de índices de busca otimiza o cálculo de dissimilaridades entre itens.

Ao final desta seção, na tabela **??**, será apresentado o conjunto de identificadores das diferentes estratégias adotadas, que a seguir são apresentadas em detalhes.

### 5.6.1 Repositórios de dados

Os repositórios utilizados na implementação estão descritos na tabela abaixo. A tabela 5.1 apresenta a fonte de dados de cada índice de busca e faz um paralelo com uma coleção de documentos de texto compostos por termos ou palavras-chave, o que em cada cenário representa um documento e o que representam termos. O modelo de espaço vetorial em cada caso pode representado por uma matriz de documentos por termos.

Tabela 5.1: Descrição de repositórios de dados utilizados

| **Identificador** | **Fonte dos dados** | **Documento** | **Termos** |
|---|---|---|---|
| axi | Pacote apt-xapian-index | pacote Debian | conjunto de metadados sobre o pacote (descrição, dependências etc) |
| txi | Base de Debtags | pacote Debian | tags |
| pxi | Popcon | submissão do popcon | pacotes instalados |

### 5.6.2 Baseadas em conteúdo

As técnicas baseadas em conteúdo foram implementadas com o suporte da biblioteca Xapian, que utiliza a técnica *BM25*, descrita na seção **??**. As recomendações podem ser produzidas com base na descrição dos pacotes já instalados pelo usuário ou por suas tags.

### 5.6.3 Colaborativas

O *Popcon* é a fonte de dados disponível atualmente para soluções colaborativas de recomendação. Neste contexto, listas de pacotes representam a identidade do usuário, que receberá

[12] http://popcon.debian.org/README

recomendações com base no que outros usuários com interesses semelhantes aos dele têm instalado em seus sistemas e ele não tem.

Como estratégias colaborativas, foram implementadas soluções baseadas nas técnicas *K-NN* e agrupamento por *k-medoid*. A composição da vizinhança do *K-NN*, no entanto foi realizada a partir de uma busca no repositório *pxi*. Já para a solução de agrupamento, foi necessário adotar medidas para aplicações em larga escala, dado que a base submissões do popcon ultrapassou a marca de 100.000 usuário, o que torna o processamento de todas as distância par-a-par extremamente custosa.

### 5.6.4 Híbridas

No sentido de refinar as recomendações produzidas através das estratégias colaborativas e baseadas em conteúdo, algumas implementações híbridas foram experimentadas. Além das listas de pacotes provenientes do *Popcon* e informações do *Debtags*, outros fatores foram considerados para composição das sugestões, alguns dos quais são descritos abaixo.

- **Áreas de interesse do usuário:** a composição de um perfil de usuário que indique suas áreas de interesse poderia refinar as sugestões e diminuir a ocorrência de anomalias como, por exemplo, a indicação de bibliotecas de desenvolvimento de software para um usuário que não é programador;

- **Popularidade dos pacotes:** aplicativos que figuram entre os mais populares no *Popcon* poderiam receber maior pontuação nos cálculos de relevância, ainda que como critério de desempate;

- **Bugs:** muitos relatórios de erro abertos para uma pacote são um indicativo de problema, principalmente se forem *bugs* graves. Devido ao número reduzido de pacotes em cada recomendação, deve-se dar prioridade à incluir aqueles que recebem atenção constante de seu mantenedor.

A consideração de um perfil de usuário caracteriza a estratégia com base em dados demográficos, enquanto que popularidade e *bugs* abertos podem ser entendidos como uma modelagem de reputação dos itens. Ambas as estratégias podem ser utilizadas para refinar os resultados obtidos com os recomendadores básicos, caracterizando uma hibridização em *cascata* (ver seção 3.6.6).

Neste contexto é possível também produzir recomendações de forma colaborativa com base em conteúdo, o que caracteriza um sistema híbrido *meta-nível*. Em vez de a identidade dos usuários ser representada por uma lista de itens, seria representada pela caracterização destes itens. No caso dos pacotes, a estratégia colaborativa seria realizada a partir das *tags* dos pacotes e não das listas de pacotes originais.

Outra possibilidade seria a elaboração de estratégias para *revezamento* entre os sistemas básicos, de acordo com o resultado do processo de seleção de atributos. O recomendador híbrido neste caso analisaria as características dos dados que seriam a entrada do algoritmo de recomendação e escolheria a técnica que melhor se adequaria a esta entrada.

Por fim, a hibridização por *combinação* teria implementação trivial dado que os recomendadores básicos já existissem, pois consiste basicamente na apresentação em conjunto dos resultados de múltiplos recomendadores. De certa forma, a combinação vai acontecer na apresentação do *survey* para o usuário.

## 5.7 Codificação

O desenvolvimento do *AppRecommender* foi majoritariamente realizado na linguagem de programação *Python*[13], principalmente pela facilidade de integração com outras ferramentas do Debian também desenvolvidas nesta linguagem. Ademais, a vasta documentação e grande variedade de bibliotecas de utilidade para o trabalho, a exemplo da *Python-cluster*[14] e *Xapian*[15], são fatores que contribuíram para esta escolha.

O código-fonte está licenciado pela GNU GPL e disponível em um repositório público[16]. O desenvolvimento obedeceu ao guia de estilo para código em python[17], fez uso de testes automatizados e padrões de projeto. A documentação é automaticamente gerada pelo Doxygen[18] e pode ser acessada no apêndice A deste documento. O diagrama UML do sistema está representado na figura **??**.

[13] http://www.python.org/
[14] http://python-cluster.sourceforge.net/
[15] http://xapian.org/
[16] http://github.com/tassia/AppRecommender
[17] http://www.python.org/dev/peps/pep-0008/
[18] http://www.stack.nl/~dimitri/doxygen/

*Capítulo 6*

# Validação da proposta

Este capítulo relata os procedimentos realizados com o intuito de validar a ferramenta desenvolvida, acompanhados pelos resultados obtidos em cada etapa. Os experimentos realizados tem caráter exploratório, no sentido de que não são realizados com o intuito de provar ou refutar uma hipótese, e sim de encontrar as soluções que mais se adequam ao domínio da aplicação e conjunto de dados utilizados como fontes de recomendação.

Os procedimentos de teste aconteceram em dois momentos distintos, apresentados nas seções a seguir.

## 6.1 Seleção de modelo

Esta etapa tem como principal objetivo a comparação entre diferentes modelos de recomendador, caracterizados por uma determinada configuração de estratégias, técnicas e outros parâmetros, a exemplo do tamanho da vizinhança para estratégias colaborativas, esquema de pesos utilizado em buscas, e abordagens distintas para seleção de atributos.

O metodologia utilizada neste momento basea-se *validação cruzada*, comumente adotada para estimativa de acurácia de modelos preditivos. Dado que tais estimativas são utilizadas primordialmente para fins de comparação entre modelos de recomendadores, a acurácia absoluta de cada estratégia é considerada menos importante, admitindo-se resultados enviesados[1] de acordo com a hipótese de que o viés afeta todos os modelos de forma similar (por exemplo, todas as estimativas são pessimistas ou todas são otimistas) [Kohavi 1995].

A ideia central da validação cruzada é o isolamento de uma porção aleatória dos dados cuja relevância é conhecida, treinamento do modelo com os demais dados e posterior submissão da porção reservada para testes. A acurácia dos resultados é medida por meio da comparação dos resultados obtidos com os esperados (medida estatística de variância). A validação em rodadas (*k-fold cross-validation*) consiste basicamente nos seguintes passos:

1. O conjunto de dados original é particionado em $k$ subconjuntos;
2. Em cada uma das $k$ rodadas, apenas um dos subconjuntos é reservado para testar o modelo ao passo que todos os outros são usados como dados treinamento;
3. Ao final dos testes, os resultados são combinados para produzir uma única estimativa.

Supondo que o conjunto de pacotes instalados em um sistema é relevante para o mesmo, em cada uma das rodadas foi selecionado aleatoriamente um subconjunto de pacotes para testar o recomendador. As métricas de avaliação foram então calculadas a partir das sugestões geradas em relação à relevância conhecida.

[1] o termo viés é utilizado em estatística para expressar erro sistemático ou tendenciosidade

### 6.1.1 Ambiente de testes

Os experimentos de seleção de modelo foram realizados no servidor brucutu.ime.usp.br, acessível apenas a partir da rede interna do IME-USP. Detalhes de software e hardware estão descritos na tabela abaixo.

Tabela 6.1: Descrição de ambiente de testes

| | |
|---|---|
| Nome | brucutu.ime.usp.br |
| Sistema Operacional | Ubuntu GNU/Linux |
| Kernel | Linux 2.6.28-19 |
| Quantidade de processadores | 8 |
| Modelo | Intel(R) Xeon(R) CPU E5440 @ 2.83GHz (64 bits) |
| Memória volátil | 32GB |

## 6.2 Experimentos realizados

## 6.3 Resultados

## 6.4 Consulta pública

Dado que não existem avaliações prévias realizadas por usuários reais acerca da utilidade dos itens, os resultados da análise *offline* ficam, senão prejudicados, ao menos limitados. O conceito de utilidade de um item é subjetivo e apenas um indivíduo dotado de subjetividade é capaz fazer esta avaliação. Métricas como novidade e surpresa são dificilmente mensuráveis por meio de validação cruzada. Uma consulta pública amplamente divulgada seria uma forma de complementar os resultados daquela análise.

Algumas ferramentas foram avaliadas para a construção do *survey*, entre as quais o *LimeSurvey*[2]. No entanto, uma avaliação de acurácia da recomendação requer que a construção do questionário aconteça de forma dinâmica, com questões específicas para cada conjunto de aplicativos sugeridos para o usuário em questão, o que não foi possível realizar com este tipo de ferramenta. Sendo assim, foi desenvolvida uma interface web para o *AppRecommender*, com um módulo de avaliação integrado.

A consulta é guiada através dos seguintes passos:

1. O usuário é convidado a experimentar o recomendador de aplicativos por meio do envio de um envia uma lista de pacotes como representação de sua identidade. Esta lista pode ser indicada por um arquivo enviado ao servidor ou pelo preenchimento de um campo de texto no formulário.

2. A configuração do recomendador é completamente parametrizável, desde a escolha da estratégia de recomendação, até o tamanho do perfil e outros ajustes específicos de cada estratégia, a exemplo do tamanho da vizinhança (usuários considerados próximos). O usuário tem ainda a possibilidade de optar por uma configuração escolhida aleatoriamente dentre as disponíveis.

3. O sistema realiza a computação necessária para gerar recomendações de acordo com a configuração escolhida para o recomendador.

[2] http://www.limesurvey.org/

4. As sugestões são apresentadas ao usuário juntamente com informações detalhadas de cada item.

5. Se o usuário concordar com os termos da pesquisa, a validação da recomendação tem prosseguimento. Neste contexto, *-1* indica que a sugestão não foi útil, *0* é uma avaliação neutra e *1* indica que o usuário aprova a recomendação.

6. Os resultados da validação são armazenados no servidor para posterior análise.

### 6.4.1 Princípios

**Possibilidade de anonimato** Dado que a consulta pública foi instrumentalizada por meio de formulário online, o usuário podia escolher entre respondê-lo de forma completamente anônima ou fornecer dados de identificação. No caso de pesquisas realizadas por meio de mensagem eletrônica, as respostas dos usuários carregam seu email, o que pode causando desconforto no quesito privacidade.

**Amostra irrestrita** A pesquisa foi disponibilizada na Internet e comunicada à populacao alvo e todas as respostas recebidas foram consideradas. Dado que a caracterização Dado que a consulta foi realizada em seguida a uma recomendação, todos os usuários foram convidados sem que houvesse o privilégio de um perfil especifico de usuário. Em pesquisas de cunho social, por exemplo, que pretende alcancar a população em geral, o uso de surveys online privilegia o acesso a usuários numa determinada faixa etaria e com interesses específicos, tornando os resultados da pesquisa não representativos.

**Minimização de dados inválidos** O formulário de envio da avaliação é dotado de mecanismos de validação dos dados, evitando assim que sejam enviadas respostas que precisem ser inutilizadas posteriormente. Outro ponto neste quesito é que múltiplas respostas de um mesmo usuário não comprometem o resultado pesquisa, dado que são referentes a validações de múltiplos resultados de recomendação. Apenas a múltiplo envio de uma mesma avaliação é inutilizado (por exemplo, quando o usuário clica mais de uma vez no botão “enviar”).

**Simplicidade e objetividade** Questionários simples evitam baixas taxas de participação e erros de medida (decorrente do usuário não entender o que está sendo perguntado). Por outro lado, a objetividade é um sinal de respeito ao tempo do usuário.

**Incentivos à participação** A presente pesquisa ofereceu como incentivo a portunidade de o participante ser listado numa página de agradecimentos referenciada no texto da dissertação, além da aquisição de conhecimento acerca do tópico da pesquisa (todos os procedimentos são detalhadamente explicados no website.)

**Independência cultural** Internacionalização/localização do survey.

### 6.4.2 Métricas

**Acurácia da recomendação** de acordo com a seção 3.10.2

**Taxa de resposta** não há meios de saber quantos usuários tiveram acesso ao questionário (ou seu link) e decidiram não participar. Apenas o número de surveys completos é conhecido, não o numero de recusas. contadores das paginas podem dar uma estimativa, ainda que não confiaveis.

**Tempo de avaliação** Tempo decorrido entre a geração de uma recomendação e a respectiva validação por parte do usuário.

**Tempo de resposta** Registo da distribuição de resposta ao longo do tempo em que a pesquisa foi disponibilizada.

### 6.4.3 Ambiente de teste

Esta etapa de experimentos foi desenvolvida no Alioth[3], o servidor *forge* do projeto Debian, que provê uma infraestrutura de apoio ao desenvolvimento colaborativo de software, principalmente que tem alguma relação com o Debian.

### 6.4.4 Experimentos realizados

O *survey* foi disponibilizado no dia 30 de junho de 2011 e após 30 dias de consulta, XX usuários participaram da pesquisa.

### 6.4.5 Resultados

Apresentação e análise dos resultados.

## 6.5 Conclusão

[3]http://alioth.debian.org

*Capítulo 7*

# Considerações finais

Resumo geral do documento, principais contribuições do trabalho, deficiências, e limitações.

## 7.1 Trabalhos futuros

1. Análise de riscos de privacidade do AppRecommender;
2. Integração com AppStream;
3. Consideração de informações temporais para pontuação multivalorada
4. Comparação sistemática no repositório de pacotes debian da saída do AppRecommender com a lista de pacotes recomendados e sugeridos por cada pacote. Análise proposta por Joey Hess em 2008[1].
5. Testes com o algoritmo OPF para agrupamento [Rocha et al. 2009]

[1] http://www.mail-archive.com/debian-devel@lists.debian.org/msg260184.html

# Referências Bibliográficas


[Abate et al. 2009] Abate, P., Boender, J., Cosmo, R. D., and Zacchiroli, S. (2009). Strong Dependencies between Software Components. Technical report, MANCOOSI - Managing the Complexity of the Open Source Infraestructure.

[Adomavicius and Tuzhilin 2005] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749.

[Agrawal and Srikant 1994] Agrawal, R. and Srikant, R. (1994). Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499. Morgan Kaufmann Publishers Inc.

[Betts 2007] Betts, O. (2007). Xapian: BM25 Weighting Scheme. *Disponível em* http://xapian.org/docs/bm25.html.

[Burke 2002] Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12:331–370.

[Castells 2006] Castells, M. (2006). A Era da Intercomunicação. *Le Monde Diplomatique Brasil*, Agosto 2006.

[Cazella et al. 2010] Cazella, S. C., Reategui, E. B., and Nunes, M. A. (2010). A ciência da opinião: estado da erte em sistemas de recomendação. In *JAI: Jornada de Atualização em Informática da SBC*, pages 161–216.

[Cosmo et al. 2008] Cosmo, R. D., Zacchiroli, S., and Trezentos, P. (2008). Package upgrades in FOSS distributions: details and challenges. In *Proceedings of the 1st International Workshop on Hot Topics in Software Upgrades*, pages 7:1–7:5. ACM.

[Freedesktop 2011] Freedesktop (2011). Distributions Wiki: Cross-distro Meeting on Application Installer. *Disponível em* http://distributions.freedesktop.org/wiki/Meetings/AppInstaller2011.

[Hegland 2003] Hegland, M. (2003). Algorithms for association rules. In Mendelson, S. and Smola, A., editors, *Advanced Lectures on Machine Learning*, volume 2600 of *Lecture Notes in Computer Science*, pages 226–234. Springer Berlin / Heidelberg.

[Herlocker 2000] Herlocker, J. L. (2000). *Understanding and improving automated collaborative filtering systems*. PhD thesis.

[Herlocker et al. 2004] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22:5–53.

[Iyengar 2010] Iyengar, S. (2010). *The Art of Choosing*. Twelve.

[Jackson and Schwarz 1998] Jackson, I. and Schwarz, C. (1998). Debian Policy Manual. *Disponível em* http://www.debian.org/doc/debian-policy.

[Jones et al. 2000] Jones, K. S., Walker, S., and Robertson, S. E. (2000). A probabilistic model of information retrieval: development and comparative experiments (Parts 1 and 2). *Inf. Process. Manage.*, 36:779–840.

[Kaufman and Rousseeuw 2005] Kaufman, L. and Rousseeuw, P. J. (2005). *Finding Groups in Data: An Introduction to Cluster Analysis (Wiley Series in Probability and Statistics)*. Wiley-Interscience.

[Kohavi 1995] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*, pages 1137–1143. Morgan Kaufmann Publishers Inc.

[Kotsiantis and Kanellopoulos 2006] Kotsiantis, S. and Kanellopoulos, D. (2006). Association rule mining: a recent overview. *GESTS International Transactions on Computer Science and Engineering*, 32:71–82.

[Manning et al. 2009] Manning, C. D., Raghavan, P., and Schütze, H. (2009). *An Introduction to Information Retrieval*. Cambridge University Press.

[Manzato 2011] Manzato, M. G. (2011). *Uma arquitetura de personalização de conteúdo baseada em anotações do usuário*. PhD thesis.

[Nussbaum and Zacchiroli 2010] Nussbaum, L. and Zacchiroli, S. (2010). The Ultimate Debian Database: Consolidating Bazaar Metadata for Quality Assurance and Data Mining. *IEEE Working Conference on Mining Software Repositories*.

[O'Mahony and Ferraro 2007] O'Mahony, S. and Ferraro, F. (2007). The Emergence of Governance in an Open Source Community. *Academy of Management Journal*, 50(5):1079–1106.

[Pannagio 2010] Pannagio, R. L. Z. (2010). Arcabouço genérico baseado em técnicas de agrupamento para sistemas de recomendação. Master's thesis, Universidade Estadual de Campinas.

[Paul 2010] Paul, R. (2010). Shuttleworth: Unity shell will be default desktop in Ubuntu 11.04. *Disponível em* http://arstechnica.com/open-source/news/2010/10/shuttleworth-unity-shell-will-be-default-desktop-in-ubuntu-1104.

[Pedronette 2008] Pedronette, D. C. G. (2008). Uma plataforma de serviços de recomendação para bibliotecas digitais. Master's thesis, Universidade Estadual de Campinas.

[Pereira 2007] Pereira, D. (2007). Uma aplicação de sistemas de recomendação: sistema de recomendação para pacotes gnu/linux. Master's thesis, Universidade Federal do Rio Grande do Sul.

[Pérez-Iglesias et al. 2009] Pérez-Iglesias, J., Pérez-Agüera, J. R., Fresno, V., and Feinstein, Y. Z. (2009). Integrating the Probabilistic Models BM25/BM25F into Lucene. *CoRR*. *Disponível em* http://arxiv.org/abs/0911.5046.

[Raymond 1999] Raymond, E. S. (1999). *The Cathedral & the Bazaar*. O'Reilly Media.

[Resnick and Varian 1997] Resnick, P. and Varian, H. R. (1997). Recommender Systems. *Communications of the ACM*, 40(3):56–58.

[Robertson 1977] Robertson, S. E. (1977). The Probability Ranking Principle in IR. *Journal of Documentation*, 33(4):294–304.

[Robertson and Walker 1994] Robertson, S. E. and Walker, S. (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *In Proceedings of SIGIR'94*, pages 232–241. Springer-Verlag.

[Rocha et al. 2009] Rocha, L. M., Cappabianco, F. A. M., and Falcão, A. X. (2009). Data clustering as an optimum-path forest problem with applications in image analysis. *Int. J. Imaging Syst. Technol.*, 19:50–68.

[Schröder 2007] Schröder, A. (2007). Konzeption und erstellung eines systems zur generierung personalisierter installationsratschläge für das debian. Master's thesis, Universität Paderborn.

[Simon and Vieira 2008] Simon, I. and Vieira, M. S. (2008). O rossio não rival. *Disponível em* http://www.ime.usp.br/~is/papir/RNR_v9.pdf.

[Torvalds and Diamond 2001] Torvalds, L. and Diamond, D. (2001). *Just for Fun: The Story of an Accidental Revolutionary*. HARPER USA.

[Vozalis and Margaritis 2003] Vozalis, E. and Margaritis, K. G. (2003). Analysis of Recommender Systems' Algorithms. In *Proceedings of the 6th Hellenic European Conference on Computer Mathematics and its Applications*.

[Zhang 2004] Zhang, H. (2004). The Optimality of Naive Bayes. In *FLAIRS Conference*. AAAI Press.

[Zini 2011a] Zini, E. (2011a). A cute introduction to Debtags. *Disponível em* http://debtags.alioth.debian.org/paper-debtags.html.

[Zini 2011b] Zini, E. (2011b). Cross-distro Meeting on Application Installer. *Disponível em* http://www.enricozini.org/2011/debian/appinstaller2011.