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, 25 de maio 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

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]. 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[4] 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 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

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

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

componentes de software em abstrações denominadas *pacotes*.

2.2 Empacotamento de programas

2.3 Sistemas gerenciadores de pacotes

2.4 Seleção de pacotes

O processo de desenvolvimento e manutenção de uma distribuição varia bastante de uma para outra e está diretamente ligado à constituição do projeto. Quando são criadas por empresas, costumam receber colaboração dos usuários de forma limitada, visto que as decisões-chave são tomadas dentro da organização. Este é o modelo de desenvolvimento descrito por [Raymond 1999] como catedral. Por outro 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. Nesses casos observa-se com mais clareza o fenômeno do consumidor produtor descrito anteriormente. 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 (ou seja, o desenvolvedor é também usuário).

A seleção e configuração dos aplicativos básicos de uma distribuição são de responsabilidade da equipe que a desenvolve, com diferentes níveis de interferência da comunidade, sendo uma questão crucial no projeto, dado que é um dos fatores de grande influência na escolha dos usuários pela distribuição. O impacto de tal seleção para os usuários finais 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].

Após a configuração de um sistema básico, a instalação de programas adicionais resulta na personalização do sistema para atender a demandas específicas dos usuários para o qual foi projetado. 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. 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

Segundo [Resnick and Varian 1997], os sistemas de recomendação aumentam a capacidade e eficácia de um processo de indicação bastante popular nas relações sociais. Existem recomendações produzidas exclusivamente por especialistas, a exemplo de indicações de filmes publicadas nos principais jornais e revistas do país por críticos de arte. Nos últimos anos, porém, a opinião e o comportamento de usuários não especializados passaram a ser considerados nas recomendações, por agregarem valor às mesmas. O que acontece de forma explícita, quando o próprio usuário escreve sua opinião ou avalia a qualidade de um item, ou implícita, quando suas preferências, comportamentos e transações são analisados e incorporados à recomendação de forma transparente ao usuário.

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.

3.1 Ações e desafios

Sistemas de recomendação são implementados nos mais diversos contextos e podem ser projetados com finalidades distintas. Abaixo estão descritas algumas ações relacionadas a estes sistemas identificadas por [Herlocker et al. 2004].

Anotação em contexto. Os primeiros sistemas de recomendação eram utilizados num cenário de informação estruturada (mensagens classificadas num contexto), e auxiliavam os usuários a selecionarem as mensagens a serem lidas dentro de cada contexto.

Encontrar itens relevantes. O sistema sugere itens para o usuário através de uma lista ordenada de forma decrescente, de acordo com a probabilidade do item ser considerado relevante pelo usuário. Esta é a ação mais comum entre os sistemas de recomendação comerciais, atraindo grande parte das pesquisas relacionadas com o tema.

Encontrar todos os itens relevantes. Em situações em que não se deseja ignorar nenhum item relevante, ao invés da recomendação de apenas alguns, todos os itens considerados relevantes devem ser retornados.

Sequência recomendada. Quando não somente os itens recomendados importa, mas também a ordem em que eles são apresentados, caracteriza-se a ação de recomendação de sequência.

Expressão de opinião. A recomendação em si muitas vezes não é o que atrai usuários desses sistemas. Alguns estão interessados simplesmente em emitir suas opiniões. Esta ação é comum em ambientes que disponibilizam um espaço para os usuários registrarem seus comentários e avaliações sobre os produtos.

Ajudar usuários. Alguns usuários utilizam sistemas de recomendação por acreditarem que a comunidade se beneficia da sua contribuição. Apesar de nem sempre aparecerem juntas, tal atividade está comumente relacionada com a expressão de opinião.

Navegação. Alguns usuários utilizam recomendadores mesmo quando não têm planos de consumir produto algum, apenas para navegar entre os itens disponíveis. Neste caso, aspectos como a interface, facilidade de uso e natureza da informação provida são de extrema importância.

O desenvolvimento de sistemas recomendadores têm como desafios questões inerentes ao problema de recomendação e sua representação computacional. As estratégias e técnicas propostas devem levar em conta tais questões e tentar contorná-las na medida do possível. Alguns destas questões 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.

3.2 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 **??**.

3.2.1 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 3.1 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 3.1: 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.

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 χ^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].

3.2.2 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 3.1. 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 ao invés de P porque geralmente não se sabe o valor exato das probabilidades, que são estimadas a partir dos dados de treinamento.

$$c_{MAP} = \arg\max_{c \in C} \hat{P}(c|x) \tag{3.1}$$

A equação 3.2 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 3.3).

$$c_{MAP} = \arg\max_{c \in C} \frac{\hat{P}(x|c)\hat{P}(c)}{\cancel{\hat{P}(x)}} \tag{3.2}$$

$$= \arg\max_{c \in C} \hat{P}(x|c)\hat{P}(c) \tag{3.3}$$

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:

$$\hat{P}(x|c) = \hat{P}(x_1, x_2, ..., x_n|c) = \hat{P}(x_1|c)\hat{P}(x_2|c) \ ... \ \hat{P}(x_n|c) \tag{3.4}$$

Portanto, a função de decisão pode ser reescrita através da equação 3.5. 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.

$$c_{MAP} = \arg\max_{c \in C} \ \hat{P}(c) \prod_{1 \leq i \leq n} \hat{P}(x_i|c) \tag{3.5}$$

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 3.6 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.

$$\hat{P}(c) = \frac{N_c}{N} \tag{3.6}$$

As probabilidades condicionais são estimadas como a frequência relativa do atributo x em objetos que pertencem à classe c. Na equação 3.7, 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.

$$\hat{P}(x|c) = \frac{T_{cx}}{\sum_{x' \in V} T_{cx'}} \tag{3.7}$$

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 3.5), 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 3.7:

$$\hat{P}(x|c) = \frac{T_{cx} + 1}{\sum_{x' \in V} T_{cx'} + 1} \tag{3.8}$$

O classificador bayesiano também é sensível a ruídos, logo, sua performance é igualmente beneficiada pelo processo de seleção de atributos descrito na seção 3.2.1.

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.

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.

3.2.3 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 3.1 apresenta um documento de texto numa coleção hipotética[1] e a sua representação após eliminação de *stop words* e procedimento de *stemming*.

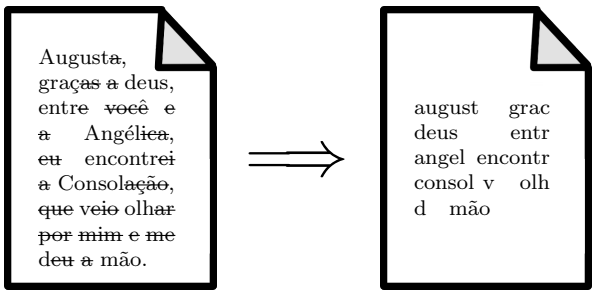

Figura 3.1: Eliminação de *stop words* e normalização do documento por *stemming*

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 3.2 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 tfs 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.

O conjunto de pesos determinado pelos tfs 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.

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

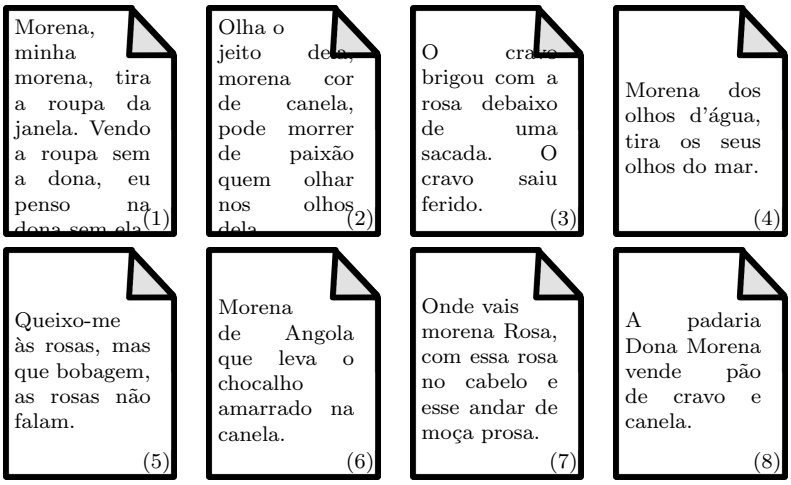

Figura 3.2: Coleção de documentos

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

Uma medida de relevância baseada simplesmente na incidência dos termos da *query* nos documentos (RI) poderia ser calculada através da equação 3.9.

$$RI_{d,q} = \sum_{t \in q} tf_{t,d} \tag{3.9}$$

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 3.2, 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 3.10 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.

$$idf_t = \log \frac{N}{df_t} \tag{3.10}$$

Valores de idf_t para a coleção da figura 3.2 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 3.11), 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).

$$tf\text{-}idf_{t,d} = tf_{t,d} \cdot idf_t \tag{3.11}$$

A medida de relevância apresentada na equação 3.9 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 3.12 [Manning et al. 2009].

$$R_{d,q} = \sum_{t \in q} tf\text{-}idf_{t,d} \tag{3.12}$$

A tabela 3.4 apresenta a ordenação dos documentos da coleção como resultado do cálculo de relevância por $tf\text{-}idf$ 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 3.11, 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 3.13).

$$tf\text{-}sub_{t,d} = \begin{cases} 1 + \log tf_{t,d} & \text{, se } tf_{t,d} > 0 \\ 0 & \text{, caso contrário} \end{cases} \tag{3.13}$$

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.

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 3.2 pode ser representada no modelo de espaço vetorial pela matriz da tabela 3.5.

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

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 (θ), apresentada na equação 3.14. O numerador representa o produto escalar dos dois vetores e o denominador a distância euclidiana entre os mesmos.

$$sim(d_1, d_2) = \cos(\theta) = \frac{V(\vec{d}_1) \cdot V(\vec{d}_2)}{|V(\vec{d}_1)||V(\vec{d}_2)|} \tag{3.14}$$

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 3.15).

$$R_{d,q} = sim(d, q) \tag{3.15}$$

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

3.2.4 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[2], sendo apontado como o melhor entre os esquemas de peso probabilísticos conhecidos [Betts 2007]. A título de ilustração, Xapian[3] e Lucene[4], 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

[2] 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.

[3] http://xapian.org/

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

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].

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 3.16).

$$P(L|D) = \frac{P(D|L)P(L)}{P(D)} \tag{3.16}$$

Para evitar a expansão de $P(D)$, a chance de $(L|D)$ é utilizada ao invés da probabilidade. Na verdade, o logaritmo da chance é aplicado (equação 3.17), 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 3.18 descreve uma pontuação por relevância referenciada como primária ($\mathit{R\text{-}PRIM}_D$).

$$\begin{aligned} \log \frac{P(L|D)}{P(\overline{L}|D)} &= \log \frac{P(D|L)P(L)}{P(D|\overline{L})P(\overline{L})} \\ &= \log \frac{P(D|L)}{P(D|\overline{L})} + \log \frac{P(L)}{P(\overline{L})} \end{aligned} \tag{3.17}$$

$$\mathit{R\text{-}PRIM}_D = \log \frac{P(D|L)}{P(D|\overline{L})} \tag{3.18}$$

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, $\mathit{R\text{-}PRIM}_D$ poderia ser estimado como um somatório de probabilidades, cada uma relacionada a cada atributo da descrição D (equação 3.19).

$$\begin{aligned} \mathit{R\text{-}PRIM}_D &= \log \prod_i \frac{P(A_i = a_i|L)}{P(A_1 = a_1|\overline{L})} \\ &= \sum_i \log \frac{P(A_i = a_i|L)}{P(A_1 = a_1|\overline{L})} \end{aligned} \tag{3.19}$$

No entanto, a fórmula 3.19 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 3.20).

$$\begin{aligned} \textit{R-BASIC}_D &= \textit{R-PRIM}_D - \sum_i \log \frac{P(A_i = 0|L)}{P(A_1 = 0|\overline{L})} \\ &= \sum_i \Big(\log \frac{P(A_i = a_i|L)}{P(A_1 = a_1|\overline{L})} - \log \frac{P(A_i = 0|L)}{P(A_1 = 0|\overline{L})} \Big) \\ &= \sum_i \log \frac{P(A_i = a_i|L)P(A_1 = 0|\overline{L})}{P(A_1 = a_1|\overline{L})P(A_i = 0|L)} \end{aligned} \tag{3.20}$$

Considerando W_i como um peso para cada termo t_i do documento (equação 3.21), $\textit{R-BASIC}_D$ pode ser então reescrito em função deste peso, como na equação 3.22.

$$W_i = \log \frac{P(A_i = a_i|L)P(A_1 = 0|\overline{L})}{P(A_1 = a_1|\overline{L})P(A_i = 0|L)} \tag{3.21}$$

$$\textit{R-BASIC}_D = \sum_i W_i \tag{3.22}$$

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 3.21 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.

$$w_i = \log \frac{p_i(1 - \overline{p_i})}{\overline{p_i}(1 - p)} \tag{3.23}$$

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.

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 3.10) é uma medida plausível e, apesar de ter sido proposta baseada apenas na frequência de incidência dos termos, também pode ser derivada da equação 3.23 [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*[5].

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.

[5]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*.

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 3.23 pode ser redefinida como a fórmula 3.24, que exprime o logaritmo da razão de chances de um termo ocorrer em documentos relevantes e irrelevantes.

$$w = \log \frac{r(N-n-R+r)}{(R-r)(n-r)} \tag{3.24}$$

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*.

$$RW_t = \log \frac{(r+0.5)(N\text{-}n\text{-}R+r+0.5)}{(R\text{-}r+0.5)(n\text{-}r+0.5)} \tag{3.25}$$

Se não houver informações provenientes da atestação de relevância, o idf_t clássico pode ser utilizado (equação 3.10), ou ainda, uma variação de RW_t a partir do estabelecimento de que $R = r = 0$ (equação 3.26).

$$RW_t = \log \frac{N\text{-}n+0.5}{n+0.5} \tag{3.26}$$

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 3.27.

$$RD_{t,D} = \frac{tf_{t,D}(k_1 + 1)}{k_1 + tf_{t,D}} \tag{3.27}$$

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, ao invés por exemplo da 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 3.28).

$$RD_{t,D} = \frac{\frac{tf_{t,D}}{NL}(k_1 + 1)}{k_1 + \frac{tf_{t,D}}{NL}} = \frac{tf_{t,D}(k_1 + 1)}{k_1 * NL + tf_{t,D}} \tag{3.28}$$

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, ao invés de mera repetição.

A fórmula proposta (3.29) 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*).

$$NL = ((1\text{-}b) + b\frac{l_d}{l_{avg}}) \tag{3.29}$$

$$RD_{t,D} = \frac{tf_{t,D}(k_1 + 1)}{k_1((1\text{-}b) + b\frac{l_d}{l_{avg}}) + tf_{t,D}} \tag{3.30}$$

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* ao invés do conjunto de documentos. O resultado é um peso semelhante ao $RD_{t,D}$, porém com parâmetros constantes próprios

(equação 3.31). Todavia, para o caso de *queries* com poucos termos, este componente do peso deve ser desconsiderado.

$$RQ_{t,Q} = \frac{(k_3 + 1)qtf_{t,Q}}{k_3 + qtf_{t,Q}} \tag{3.31}$$

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 3.32.

$$R_{D,Q} = \sum_{t \in Q} RW_t \cdot RD_{t,D} \cdot RQ_{t,Q} \tag{3.32}$$

3.2.5 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 3.3 ilustra através de um diagrama de *Hasse* o conjunto das partes do universo de itens $U = \{a, b, c\}$.

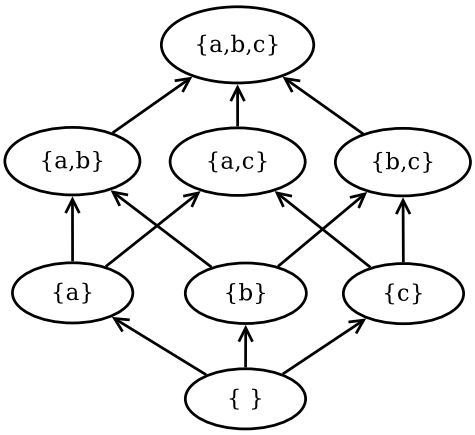

Figura 3.3: Conjunto das partes ilustrado por um diagrama de *Hasse*

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. Ao invés 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 3.4 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 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].

3.3 Composição de recomendações

Neste trabalho considera-se uma classificação mista dos seguintes autores. [Burke 2002] distingue as diferentes estratégias de recomendação a partir da fonte de dados de onde é

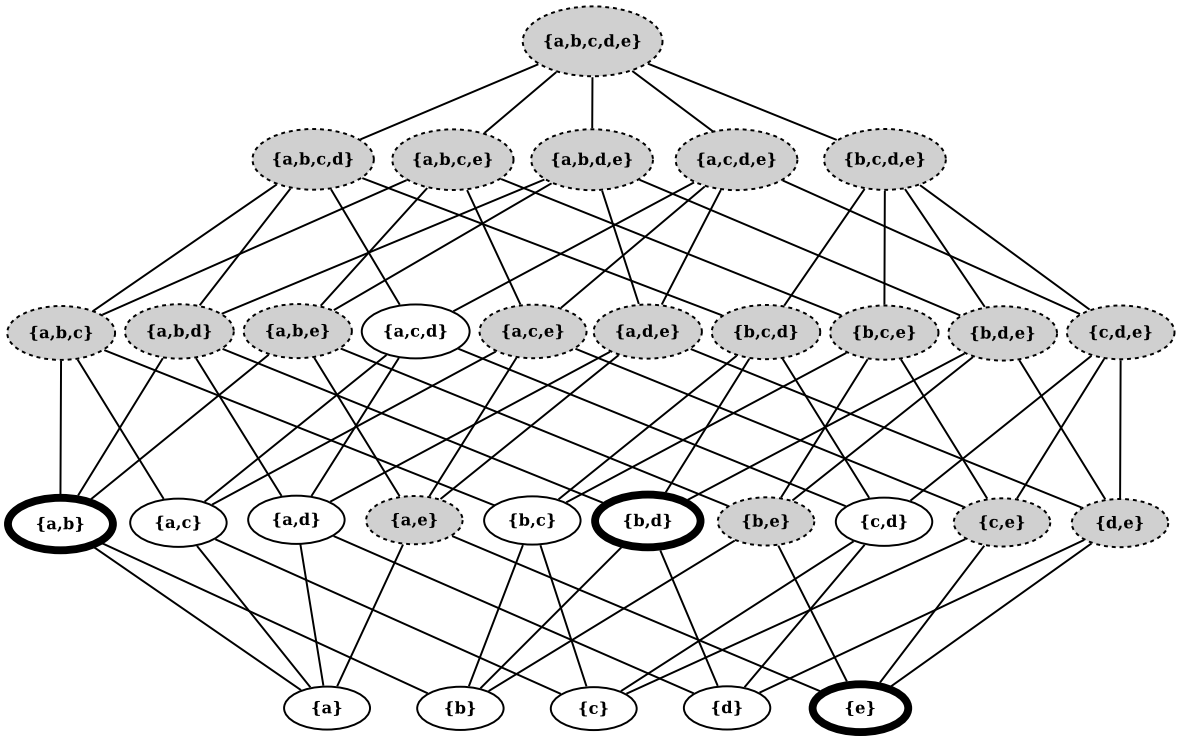

Figura 3.4: Geração de conjuntos candidatos pelo algoritmo Apriori

extraído o conhecimento para produzir as recomendações. [Cazella et al. 2010] propõe uma classificação um pouco mais abrangente, considerando por exemplo a recomendação por reputação dos itens, que por não oferecer grandes desafios computacionais é omitida por algumas taxonomias.

3.3.1 Reputação dos itens

Popular entre serviços de venda como livrarias, sites de leilão e lojas de modo geral, esta estratégia consiste no armazenamento de avaliações dos produtos escritas por usuários, bem como na apresentação das mesmas no momento e local apropriado [Cazella et al. 2010]. A implementação desta solução é simples, visto que exige apenas a manutenção dos dados originais, não sendo necessária análise posterior alguma. No entanto, tem-se como premissa a imparcialidade dos usuários em suas opiniões, que de fato não pode ser verificada devido a seu caráter subjetivo e estritamente pessoal. Atualmente existem serviços especializados em reputação de produtos que não realizam venda associada, apenas disponibilizam as avaliações. É o caso do *Internet Movie Database* [6], apresentado na figura 3.5.

3.3.2 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]. O ponto chave desta estratégia é a caracterização dos itens, por exemplo, através da identificação de atributos (autores e temas de livros, por exemplo). A partir dos atributos dos itens, aplica-se técnicas de recuperação da informação (por exemplo, *tf-idf* e *BM25*) para encontrar itens semelhantes ou de classificação (por exemplo, *Bayes ingênuo* e *K-NN*) para encontrar itens relevantes. Em uma livraria, sugerir ao cliente outros livros do mesmo autor ou tema de livros previamente selecionados é uma estratégia amplamente adotada.

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

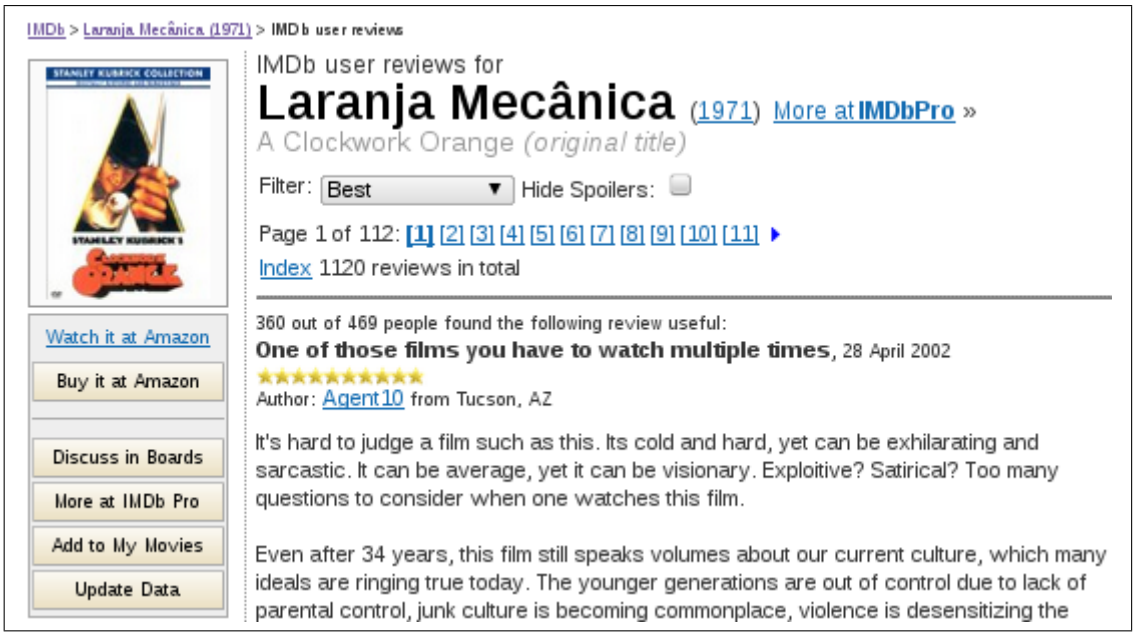

Figura 3.5: Avaliação de usuário no IMDb

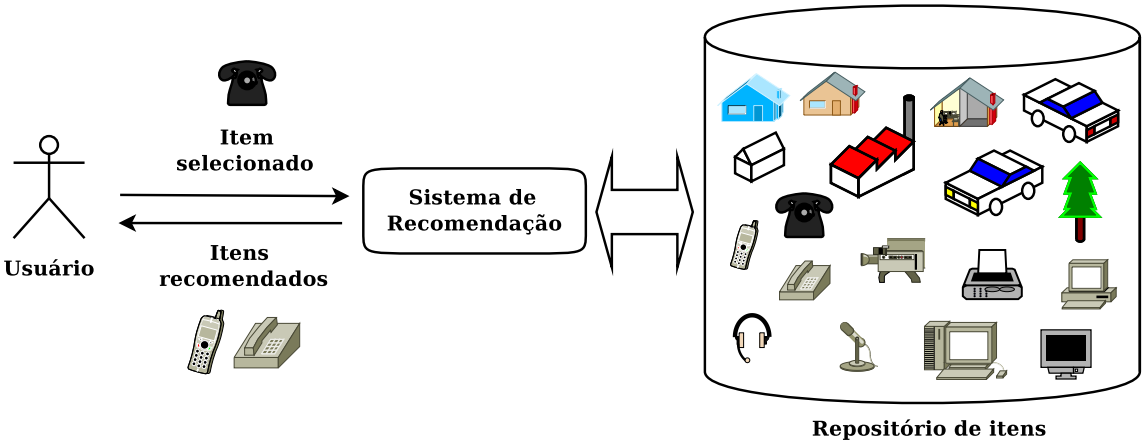

Figura 3.6: Cenário de uma recomendação baseada em conteúdo

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 automatizada. A superespecialização é outro problema indicado por [Adomavicius and Tuzhilin 2005], que diz respeito à abrangência das recomendações estar limitada a itens similares aos já escolhidos pelos usuários.

3.3.3 Recomendação colaborativa

Esta estratégia não exige o reconhecimento semântico do conteúdo dos itens, pois é fundamentado na troca de experiências entre indivíduos que possuem interesses em comum. A figura 3.7 ilustra o cenário da recomendação colaborativa.

A técnica *K-NN* é comumente utilizada neste tipo de solução. Define-se uma função que representa a proximidade entre os usuários. Com base nesta medida, a vizinhança de um determinado usuário é composta por outros k usuários que estiverem mais próximos a ele. Porém, ao invés de uma classe ser extraída da vizinhança, como descrito na seção 3.2.1,

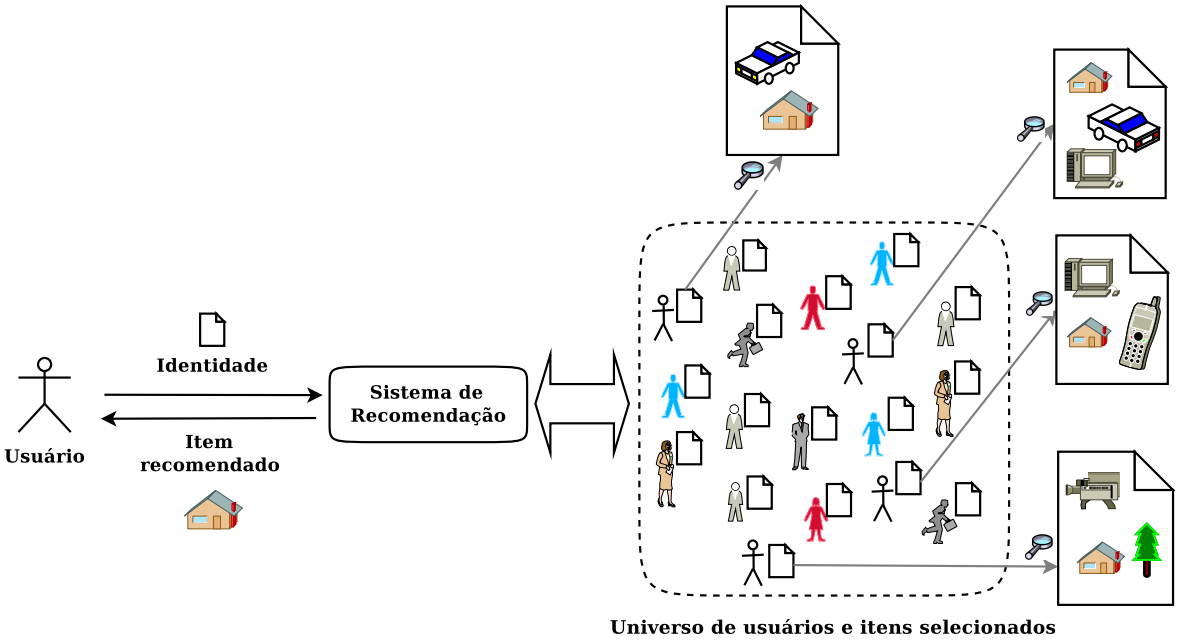

Figura 3.7: Cenário da recomendação colaborativa

uma recomendação para o usuário é produzida a partir da análise dos itens que os usuários vizinhos consideram relevantes. Geralmente os itens que ocorrem com maior frequência na vizinhança compõem a recomendação.

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

3.3.4 Baseada em conhecimento

Esta estratégia tem como princípio a descoberta de conhecimento a partir da análise de uma base de dados de transações, que registra a escolha dos usuários ao longo do tempo. Técnicas de classificação e mineração de dados são utilizadas para extrair correlações e padrões frequentes no comportamento dos usuários. Tal abordagem é frequentemente utilizada em recomendações implícitas, por exemplo, na definição do posicionamento de produtos numa prateleira ou a realização de propagandas dirigidas [Hegland 2003].

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. Uma das técnicas mais utilizadas é o *k-means*, que consiste basicamente em: (1) seleção de k usuários considerados sementes; (2) associar cada usuário da base de dados com a semente mais próxima dele; (3) calcular novos elementos centrais para cada grupo, entre os usuários que o compõem. O passo 3 é repetido até que não seja mais necessário calcular novos centróides. Desta forma, um sistema de recomendação poderia sugerir itens de acordo com as características de cada grupo [Cazella et al. 2010].

Para descoberta de regras de associação, as técnicas mais utilizadas são variações do algoritmos Apriori, apresentado na seção 3.2.5 [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". Um exemplo de recomendação por associação é encontrado na loja virtual da empresa *Amazon*[7] (figura 3.8).

Figura 3.8: Recomendação por associação na Amazon

3.3.5 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.

3.3.6 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 3.8 apresenta as principais técnicas de hibridização, segundo [Burke 2002].

Tabela 3.8: 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.

3.4 Avaliação

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

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

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 3.1), como guia para as decisões metodológicas ao longo dos experimentos.

3.4.1 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 3.4.2. 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].

3.4.2 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

4.1 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].

4.2 Informações sobre pacotes

4.2.1 Debian

O projeto Debian tem se destacado no universo das distribuições por suas iniciativas pioneiras no campo de gerenciamento de aplicações [Zini 2011]. 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 tabela 4.1 relaciona algumas destas iniciativas que estão diretamente ligadas ao gerenciamento de pacotes. Vale ressaltar que a maioria destas soluções foi inicialmente desenvolvida num contexto extra-oficial e ao passo que se mostraram úteis e eficientes foram absorvidas pela comunidade de usuários e desenvolvedores.

4.3 Trabalhos descontinuados

Ao longo do desenvolvimento deste trabalho, esforços anteriores de desenvolvimento na área de recomendação e mineração de dados sobre pacotes Debian foram identificados e estão descritos abaixo.

[1] http://qa.debian.org

Tabela 4.1: Gerência de informações sobre pacotes Debian

Solução	***Descrição***	***Estratégia de recomendação***
BTS *Bug Tracking System* http://bugs.debian.org	Sistema de acompanhamento de *bugs*, alimentado pelos usuários e desenvolvedores da distribuição. Reúne todo o histórico referente ao relatório e correção de erros em pacotes.	Reputação
Popcon *Popularity Contest* http://popcon.debian.org	Concurso de popularidade entre pacotes realizado diariamente, disponibiliza estatísticas de uso dos pacotes do repositório. Provê gráficos específicos por pacote, por arquitetura, por desenvolvedor etc.	
PTS *Package Tracking System* http://packages.qa.debian.org	Sistema de acompanhamento de pacotes, reúne informações relativas à manutenção dos pacotes: versão, mantenedor, *upstream*, *bugs* abertos por tipo, últimas atualizações no repositório etc.	
UDD *Ultimate Debian Database* http://udd.debian.org	Iniciativa recente do time de qualidade criada com o intuito de reunir informações de diversos aspectos do Debian numa base de dados única. Usuários avançados podem consultar esta base para tomar decisões acerca de que pacotes utilizar.	
Debtags Classificação de pacotes http://debtags.alioth.debian.org	Caracterização dos pacotes por múltiplos atributos, realizada manualmente por usuários e desenvolvedores, que auxilia a navegação e busca no repositório de pacotes.	Baseada em conteúdo
	Novas *tags* (atributos) são sugeridas a partir das *tags* já associadas ao pacote.	Associação

4.3.1 PopSuggest/Anapop

O *PopSuggest*[2] oferecia recomendações a partir de dados do *Popcon* como uma ilustração das possibilidades de uso dos dados coletados.

4.3.2 Debommender

O *Debommender*[3] foi desenvolvido como prova de conceito no âmbito de um trabalho de graduação, não sendo porém integrado aos serviços da distribuição.

4.3.3 Mineração de dados do Popcon

Trabalho apresentado por Alain Schroeder na Debconf7.

[2]http://www.enricozini.org/2007/debtags/popcon-play/
[3]http://ostatic.com/debommender

Capítulo 5

App-Recommender

Esta seção apresenta as fases de desenvolvimento deste trabalho, de acordo com seu planejamento até a presente data.

5.1 Escolha da plataforma

A distribuição escolhida como base para o desenvolvimento deste trabalho foi o Debian GNU/Linux. No entanto, a codificação será realizada com o maior grau possível de independência de plataforma, 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 (Advanced Packaging Tool)*[1]. 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.

2. **Disponibilidade de dados estatísticos.** O *Popcon (Popularity Contest)*[2] é a concurso de popularidade entre pacotes. Os usuários que aceitam participar do concurso enviam periodicamente a sua lista de pacotes instalados no sistema, que são armazenados no servidor do Popcon. Diariamente as listas recebidas são processadas e dados estatísticos acerca do uso dos pacotes são gerados e disponibilizados no website do projeto.

3. **Possibilidade de integração dos resultados do trabalho.** Segundo o *contrato social Debian*[3], o desenvolvimento do projeto é guiado pelas necessidades dos usuários e da comunidade. Portanto, as iniciativas de colaboradores individuais, sejam eles

[1] http://wiki.debian.org/Apt
[2] http://popcon.debian.org
[3] http://www.debian.org/social_contract.pt.html

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 ativas em sua base de dados[4], classifica o Debian GNU/Linux entre as 10 distribuições mais populares[5]. O Debian aparece na quinta posição em suas estatísticas de páginas visitadas[6]. Já o *Linux Counter*[7] apresenta o Debian como a segunda distribuição mais popular entre as máquinas cadastradas que rodam o kernel Linux (16%), ficando atrás apenas do Ubuntu[8] (24%), que é uma distribuição derivada do Debian. Nas pesquisas da *W3Techs* sobre tecnologias para serviços web[9], 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[10], 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 Pacotes Debian

5.3 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 3.1).

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

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.

5.3.1 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.

5.3.2 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, ao invés de um conjunto de pacotes específicos.

Algumas funcionalidades de pacotes podem ser extraídas a partir da lista de pacotes

virtuais[11]. 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.

5.4 Estratégias de recomendação

(a) **Baseada em conteúdo**

Pretende-se implementar as técnicas *BM25* e *Bayes ingênuo* para recomendações de pacotes com base na descrição dos pacotes já instalados pelo usuário.

A caracterização dos pacotes através de *Debtags* é essencial para a implementação de estratégias baseadas em conteúdo. Fazendo um paralelo com uma coleção de documentos de texto compostos por palavras-chave, o cenário neste contexto é de uma coleção de descrições de pacotes compostas por *tags*. O modelo de espaço vetorial neste caso é representado por uma matriz de pacotes por *tags*.

(b) **Colaborativa**

Através de estratégias colaborativas, pretende-se implementar recomendadores que analisem as escolhas de pacotes de outros usuários para compor a sua lista de sugestões, com base na técnica *K-NN*.

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

(c) **Híbrida**

No sentido de refinar as recomendações produzidas através das estratégias colaborativa e baseada em conteúdo, algumas implementações híbridas serão experimentadas. Além das listas de pacotes provenientes do *Popcon* e informações do *Debtags*, outros fatores podem ser 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.

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

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 3.3.6).

Neste contexto é possível também produzir recomendações de forma colaborativa com base em conteúdo, o que caracteriza um sistema híbrido *meta-nível*. Ao invés 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.

5.5 Codificação

O desenvolvimento de software será majoritariamente realizado na linguagem de programação *Python*[12], 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 *NLTK*[13] e *Xapian*[14], são fatores que contribuíram para esta escolha.

[12] http://www.python.org/
[13] http://www.nltk.org/
[14] http://xapian.org/

Capítulo 6

Validação da proposta

6.1 Procedimentos de teste

Uma avaliação preliminar das diferentes estratégias de recomendação, diferentes abordagens de seleção de atributos, bem como o ajuste de parâmetros dos algoritmos será realizada através de rodadas de validação cruzada. Dado que o conjunto de pacotes instalados em um sistema é sabidamente relevante para o mesmo, pode-se selecionar aleatoriamente um conjunto de pacotes de teste e utilizar o conjunto restante para treinamento dos algoritmos. As métricas de avaliação são então aplicadas a partir dos resultados obtidos para uma série de testes.

As soluções que apresentarem melhores resultados nos testes preliminares serão avaliados por meio de uma consulta pública. Diante da inexistência de dados que possibilitassem a análise *offline* da acurácia de recomendações (avaliação real), optou-se pela realização de experimentos diretamente com usuários por meio de um *survey* eletrônico. Algumas ferramentas estão sendo avaliadas para a construção do *survey*, entre elas, o *LimeSurvey*[1].

A consulta será guiada através dos seguintes passos:

1. O usuário envia uma lista de pacotes, como representação de sua identidade. Se desejar, fornece informações adicionais para composição de um perfil baseado em seus interesses pessoais, além de indicar a quantidade de pacotes que deseja receber como sugestão;
2. O sistema realiza a computação necessária para gerar recomendações utilizando diferentes estratégias;
3. As recomendações são apresentadas ao usuário, juntamente com informações detalhadas de cada item e explicação acerca dos procedimentos realizados;
4. O usuário avalia as recomendações apresentadas;
5. A análise desta recomendação é realizada com base na aplicação de algumas métricas apresentadas na seção 3.4.2.

Ao término do período de aplicação do *survey*, os dados de avaliações individuais serão compilados numa análise de esfera global. Os gráficos e considerações resultantes serão apresentados na versão final deste trabalho.

6.1.1 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

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

questões relativas à privacidade[2], 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.

6.1.2 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 3.2.1, 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 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[3], 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.

6.2 Ambiente de testes

6.3 Experimentos realizados

6.4 Análise dos resultados

6.5 Comparação com trabalhos correlatos

6.6 Conclusão

[2] http://popcon.debian.org/FAQ
[3] http://popcon.debian.org/README

Capítulo 7

Conclusões

Considerações finais: resumo geral do documento, principais contribuicoes do trabalho, deficiencias/limitacoes e trabalhos futuros.

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.

[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.

[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.

[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.

[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.

[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.

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