Aplicação de Estratégias de Recomendação ao Domínio de Componentes de Software

Tássia Camões Araújo [1]

[1]Instituto de Matemática e Estatística
Universidade de São Paulo (USP)

EXAME DE QUALIFICAÇÃO DE MESTRADO
Programa: Ciência da Computação
Orientador: Prof. Dr. Arnaldo Mandel

Resumo. *A crescente expansão na oferta de serviços na rede mundial de computadores vem expondo cada vez mais seus usuários a inúmeras possibilidades de escolha. Esta visível pluralidade, somada à infinidade de interesses dos indivíduos envolvidos, acaba por demandar uma organização das informações de forma que seja possível aproximar destes usuários aquilo que supõe-se ser o que verdadeiramente necessitam. Tal suposição é auxiliada por sistemas computacionais de recomendação, que em geral utilizam o próprio comportamento do usuário como componente fundamental para decidir o que lhe deve ser apresentado como opções mais suscetíveis a aceitação. O presente trabalho propõe o desenvolvimento de um sistema que auxilia a recomendação de componentes de software. Vê-se em especial no universo dos softwares de código aberto uma enorme diversidade de opções que, pelo seu caráter majoritariamente não comercial, são apresentadas de forma menos ostensiva aos seus potenciais usuários. A distribuição Debian GNU/Linux reúne milhares de componentes de software nestas circunstâncias sob uma sólida infraestrutura, oferendo assim a este trabalho uma série de benefícios técnicos para os experimentos. Esta distribuição será portanto utilizada, onde propõe-se a implementação e posterior análise comparativa das técnicas abordadas de recomendação, que por fim será objeto de uma consulta pública aos usuários acerca da sua eficácia, a ser também compilada e comentada na finalização do trabalho.*

1. Introdução

A popularização de recursos computacionais e do acesso à Internet nas últimas décadas proporcionou um aumento expressivo na diversidade de serviços e conteúdo à disposição dos usuários. Um dos fatores para este aumento é que os usuários, que anteriormente eram considerados meros consumidores, apresentam-se atualmente como produtores de conteúdo. [Castells 2006] analisa este fenômeno e afirma que a maioria da população acredita que pode influenciar outras pessoas, atuando no mundo através da sua força de vontade e utilizando seus próprios meios. Isto pode ser observado no surgimento e proliferação de serviços criados e mantidos pelos usuários: blogs, enciclopédias colaborativas, como a Wikipedia[1], repositórios para compartilhamento de fotografia e vídeo, como Flickr [2] e Youtube[3], entre outros. Considerando a produção em termos de software, observa-se o exemplo das comunidades de software livre e/ou de código aberto (FOOS[4]), que propiciam a construção coletiva de uma ampla gama de softwares de qualidade em constante atualização e evolução, organizados na forma de um rossio [Simon and Vieira 2008].

A grande diversidade de opções disponíveis nestes ambientes, apesar de positiva, representa uma sobrecarga de informações que pode confundir o usuário final. [Iyengar 2010] afirma que "mais é menos" (em inglês, *"more is less"*), no sentindo de que quanto maior for a disponibilidade de escolhas, menor será a satisfação do usuário. O fato é que comumente o indivíduo possui pouca ou nenhuma experiência pessoal para realizar escolhas em determinado contexto [Cazella et al. 2010]. Sendo assim, recomendações de outras pessoas são de grande utilidade, pois reduzem as dúvidas e auxiliam o processo de escolha dentre as muitas alternativas apresentadas. No entanto, diante do número de usuários e do volume de conteúdo que usualmente deve ser considerado, recomendações no estilo "boca a boca" tornam-se ineficientes, pois exigem a comunicação direta entre os pares. A tecnologia assume então papel fundamental neste processo [Shardanand and Maes 1995].

Sistemas de recomendação emergiram como uma área de pesquisa independente na década de 90, tendo como fonte de estratégias para a automatização de recomendações soluções nas áreas de ciência cognitiva, teoria da aproximação, recuperação da informação, teorias de predição, administração e marketing [Adomavicius and Tuzhilin 2005]. O tema ganhou destaque com o crescimento do comércio eletrônico, onde apresentar o que o usuário tem interesse pode significar conquistar o cliente. Os Sistemas de Recomendação fazem a associação entre objetos de interesse e pessoas neles interessadas, filtrando as informações de forma a apresentar somente aquilo que seja relevante para o usuário. Além da agilidade para encontrar o que se deseja, tais sistemas possibilitam a personalização de serviços e conteúdos, que são apresentados de maneira individualizada a partir da identificação de interesses pessoais.

Este trabalho se insere no contexto de desenvolvimento de componentes de soft-

[1] `http://wikipedia.org`

[2] `http://flickr.com`

[3] `http://youtube.com`

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

ware, no qual os usuários são modelados como clientes e os componentes desenvolvidos como itens pelos quais os usuários têm interesse ou não. Assume-se que cada usuário tem um sistema operacional instalado e deve escolher quais aplicativos extras deseja obter para suprir suas necessidades pessoais. Diante da enorme quantidade de software disponível nas mais diversas áreas de aplicação, configura-se um cenário onde um sistema de recomendação traria benefícios imediatos ao usuário, por auxiliá-lo a tomar decisões acerca da configuração do seu ambiente de trabalho.

O objetivo principal deste trabalho é a experimentação de diferentes técnicas e estratégias para a construção de sistemas recomendadores no domínio de componentes de software, utilizando como ambiente de desenvolvimento a distribuição Debian GNU/Linux. Ao final deste estudo pretende-se integrar a estratégia que obtiver resultados mais satisfatórios à infraestrutura existente desta distribuição, aproximando os serviços atualmente disponíveis ao estado da arte em sistemas de recomendação.

O presente texto está organizado como segue: a seção 2 traz uma introdução sobre sistemas de recomendação, seus desafios, as principais técnicas e estratégias utilizadas em seu desenvolvimento, além de métodos de avaliação destes sistemas; o ambiente de desenvolvimento deste estudo, distribuições GNU/Linux, é descrito na seção 3, seguida pela exibição de soluções existentes para a recomendação neste domínio (seção 4); na seção 5 é apresentada a metodologia de realização deste trabalho e, por fim, a seção 6 traz a conclusão preliminar deste projeto, até a presente etapa de execução.

2. Sistemas de recomendação

Os Sistemas de Recomendação aumentam a capacidade e eficácia de um processo de indicação que é bastante popular nas relações sociais [Resnick and Varian 1997]. Existem recomendações que são produzidas exclusivamente por especialistas na área, a exemplo de indicações de filmes por críticos de arte publicadas nos principais jornais e revistas do país. Nos últimos anos, porém, a opinião e o comportamento de usuários não especializados passaram a ser considerados nas recomendações, por agregarem valor às mesmas. Isto pode acontecer 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 aos usuários.

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.

2.1. Ações

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 deve sugerir 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. Muito comum em ambientes que disponibilizam um espaço para os usuários registrarem suas 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.

2.2. Desafios

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 usuários-item por usuário resulta numa matriz de relacionamentos esparsa, o que dificulta a localização de usuários com preferências semelhantes (relações de vizinhança), resultando em recomendações fracas.

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

Perda de transitividade de vizinhança. Usuários que têm comportamento semelhante a um determinado usuário u não necessariamente têm comportamento semelhante entre si. A captura deste tipo de relação pode ser desejável mas em geral esta informação não e 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.

Problema da 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".

Problema do 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.

2.3. 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 busca por informação e classificação.

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 normalmente é representada por um conjunto de *palavras-chave*. A partir dos termos da *query*, o buscador recupera os documentos da coleção que sejam relevantes para a mesma. 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.

O problema de recomendação pode ser visto como uma classificação, na qual os itens devem ser categorizados entre duas classes: relevantes ou 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. Os classificadores "inteligentes" são construídos a partir de um conjunto de dados tratados como exemplos para treinamento em algoritmos de aprendizado de máquina.

A seguir são apresentadas algumas técnicas para resolução destes problemas que também são utilizadas na construção de sistemas de recomendação, dando suporte às estratégias apresentadas na seção 2.4.

2.3.1. Medida tf-idf

Acrônimo para *term frequency - inverse document frequency*, o *tf-idf* é uma medida de peso clássica utilizada no desenvolvimento de buscadores para ordenar os itens por relevância.

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, que pode ser forte ou fraca, em diversos graus. Porém, para a recuperação de informações, além da presença também é importante conhecer a frequência dos termos em um 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 pontuação maior. Desta forma, para cada termo do documento é atribuído um peso proporcional ao número de ocorrências do mesmo, designado como $tf_{t,d}$ (*term frequency*). Em sua abordagem mais simples, $tf_{t,d}$ é igual ao número de ocorrências do termo t no documento d.

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

Por outro lado, 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. Por exemplo, supondo uma *query* composta por dois termos, um muito frequente e outro muito raro, dois documentos que contenham apenas um deles com a mesma frequência terão o mesmo valor de tf. Porém, enquanto este valor de tf se repete ao longo da coleção para o termo frequente, ele pode ser único para o termo raro, o que de fato diferencia tal documento dos outros. O idf_t (*inverse document frequency*) foi então introduzido para atenuar o efeito de termos usuais no cálculo de relevância, diminuindo o peso de um termo por um fator que cresce com sua frequência na coleção. Na equação (1), N representa o número de documentos da coleção e df_t (*document frequency*) é o número de documentos que contém o termo t.

$$idf_t = log\frac{N}{df_t} \tag{1}$$

A medida *tf-idf*$_{t,d}$ combina as definições de *tf* e *idf*, produzindo um peso composto (*tf-idf*$_{t,d} = tf_{t,d} \times idf_t$) com as seguintes propriedades [Manning et al. 2009]:

1. É alta quando ocorre muitas vezes em poucos documentos;
2. Diminui quando ocorre menos vezes num documento ou em muitos documentos;
3. É muito baixa quando o termo ocorre em quase todos os documentos.

No cálculo de relevância geralmente são desconsiderados alguns termos, designados como *stop words*, que são muito frequentes e pouco informativos acerca do teor do documento. Artigos e pronomes, por exemplo, normalmente figuram nesta categoria. Um exemplo de cálculo de pesos para um conjunto de documentos é ilustrado na figura 1.

Figura 1. Exemplo de cálculo de pesos para um conjunto de documentos

Existem diversas variantes para o cálculo da medida *tf-idf*$_{t,d}$, propostas ao longo do tempo com o intuito de resolver anomalias e aperfeiçoar a busca. Por exemplo, é pouco provável que a persença de uma palavra 20 vezes num documento tenha de fato 20 vezes mais representatividade que uma ocorrência única. O logaritmo pode ser incorporado no cálculo do *tf* para atenuar o crescimento do peso para valores crescentes de frequência. Outras abordagens alternativas utilizam normalizações por diversas medidas: pelo comprimento do documento, pelo comprimento médio dos documentos da coleção, pelo *tf* máximo ou médio entre os *tf*s de todos os termos do documento, entre outros. É comum também que elementos constantes sejam introduzidos para suavizar um ou outro fator da fórmula.

Representando cada documento como um vetor de termos do dicionário e seus respectivos pesos *tf-idf* no documento, um cálculo possível de pontuação de um documento para uma determinada *query* seria a soma dos pesos dos termos da consulta (equação (2)).

$$R_{d,q} = \sum_{t \in q} tfidf_{t,d} \tag{2}$$

Similaridade de cossenos

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 a similaridade de cossenos dos vetores que os representam, apresentada na equação (3). O numerador representa o produto escalar dos dois vetores e o denominador a distância euclidiana entre os mesmos.

$$sim(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|} \tag{3}$$

Dado um documento d, para encontrar os documentos de uma coleção que mais se assemelham mais a este, basta encontrar aqueles com maior similaridade de cossenos 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.

Uma coleção de N documentos pode ser representada pelo conjunto de seus N vetores, resultando numa visão da coleção como uma matriz de termos por documentos. Tal matriz tem dimensões $M \times N$, onde as linhas representam os M termos da coleção. Esta representação, conhecida como *modelo de espaço vetorial*, é amplamente utilizada em soluções para recuperação da informação.

Considerando o fato de que *queries*, assim como documentos, são um conjunto de palavras, elas também podem ser representadas como vetores no modelo de espaço vetorial. Portanto a similaridade de cossenos pode ser utilizada na busca por documentos relevantes para uma consulta, que serão os que mais se assemelham à referida *query*. Após o cálculo de similaridade de cossenos entre os vetores da *query* e dos documentos da coleção, os maiores valores indicarão os documentos mais relevantes para a mesma.

2.3.2. Okapi BM25

O modelo *Okapi BM25* figurou por diversos anos como o melhor esquema de pesos probabilísticos nas avaliações do TREC[5] e é considerado o estado da arte em recuperação da informação [Pérez-Iglesias et al. 2009]. É uma evolução de outros dois esquemas, *BM11* e *BM15* (*BM* de *"Best Match"*), sendo fundamentado teoricamente na disciplina de probabilidade discreta, ao modelar a frequência de termos em documentos utilizando distribuições de Poisson. Seu nome advém do primeiro sistema no qual foi implementado, denominado *Okapi*.

O *BM25* utiliza os componentes estatísticos do *tf-idf*, porém numa combinação diferente, além de considerar informações como o comprimento dos documentos, a

[5]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.

frequência dos termos na *query* e outros fatores constantes. A fórmula do *BM25* é melhor compreendida através da análise de seus três componentes principais, que neste trabalho são denominados de WT, WD e WQ.

WT é a porção do peso que considera a frequência do termo no documento ($tf_{t,d}$), permitindo haver normalização pelo tamanho do documento. Na fórmula (4), l_d representa o comprimento do documento d e l_{avg} o comprimento médio dos documentos da coleção.

$$WT_{t,d} = \frac{tf_{t,d}(k_1 + 1)}{k_1((1 - b) + b\frac{l_d}{l_{avg}}) + tf_{t,d}} \tag{4}$$

A constante k_1 determina o quanto o peso do documento deve ser afetado por um acréscimo no valor de tf. Se $k_1 = 0$, este componente da formula é eliminado, portanto tf passa a não interferir no peso. Para valores altos de k_1, o peso passa a ter um crescimento linear com relação a tf. De acordo com experimentos do TREC, valores entre 1.2 e 2 são os mais indicados. Valores dentro deste intervalo implicam numa interferência de tf altamente não linear, isto é, após 3 ou 4 ocorrências o impacto de uma ocorrência adicional é mínimo [Jones et al. 2000].

O parâmetro $b \in [0, 1]$ foi introduzido para ajustar o grau de normalização pelo comprimento dos documentos. Se a configuração for $b = 1$, a normalização tem efeito completo (equivalente ao modelo *BM11*). Valores menores reduzem este efeito. Se $b = 0$, o comprimento do documento não afeta a pontuação final, (como no modelo *BM15*).

O segundo componente do peso (WD) considera informações da atestação de relevância provida pelo usuário (em inglês, *relevance feedback*[6]), quando disponível. Na equação (5), N é o número total de documentos na coleção, df_t é o número de documentos que contém o termo t, R é o número total de documentos relevantes e r é o número de documentos relevantes que contém o termo t.

$$WD_{rf} = log\frac{(r + 0.5)(N - df_t - R + r + 0.5)}{(R - r + 0.5)(df_t - r + 0.5)} \tag{5}$$

Se não houver informações provenientes da atestação de relevância, o idf_t clássico pode ser utilizado (equação (1)), ou ainda, uma variação de WD a partir do estabelecimento de que $R = r = 0$ na equação acima (equação (6)).

$$WD = log\frac{N - df_t + 0.5}{df_t + 0.5} \tag{6}$$

A terceira parte da expressão, definida aqui como $WQ_{t,q}$ (equação (7), permite a consideração da frequência do termo na *query*, que é apropriada para o caso de consultas

[6] *Relevance feedback* é o mecanismo através do qual o usuário avalia o resultado da consulta, marcando os itens retornados como relevantes ou irrelevantes, informação que é posteriormente utilizada para refinar a consulta

longas, podendo ser desconsiderada para *queries* com poucos termos.

$$WQ_{t,q} = \frac{(k_3 + 1)qtf_{t,q}}{k_3 + qtf_{t,q}} \tag{7}$$

De forma análoga ao *tf-idf*, o cálculo da pontuação do documento d para a *query* q no modelo *BM25* pode ser realizado através da fórmula (8).

$$R_{d,q} = \sum_{t \in q} WT_{t,d} \times WD_t \times WQ_{t,q} \tag{8}$$

O *BM25* é amplamente utilizado em diversos tipos de busca e coleções. Este modelo tem obtido sucesso especialmente nas avaliações do TREC, o que provocou sua adoção por muitos grupos [Jones et al. 2000]. A título de exemplo, o Xapian[7] e o Lucene[8], bibliotecas livres para construção de ferramentas de busca, são projetos de grande destaque na comunidade que utilizam o *BM25* como medida de pesos [Betts 2007] [Pérez-Iglesias et al. 2009].

2.3.3. Classificador bayesiano

Naive Bayes é uma solução para classificação que figura entre os algoritmos de aprendizado de máquina supervisionados. Cada exemplo utilizado na fase de treinamento deve relacionar um objeto, caracterizado por um conjunto de atributos, e uma classe. É dito supervisionado porque o ser humano que atribuiu classes aos objetos dos exemplos atua como supervisor, como um professor orientando o processo de aprendizado.

O classificador *Naive Bayes* (Bayes ingênuo) tem como base um modelo probabilístico que aplica o teorema de Bayes com fortes suposições de independência de atributos – por isso é considerado ingênuo. Em outras palavras, a presença ou ausência de um atributo em uma classe não estaria relacionada com a incidência de nenhum outro atributo. De certa forma a suposição de independência é análoga no modelo de espaço vetorial, onde cada atributo é representado na matriz por uma dimensão ortogonal a todas as outras [Manning et al. 2009].

A decisão acerca da classe a qual um objeto pertence é tomada de acordo com o modelo de probabilidade máxima posterior (em inglês, *maximum a posteriori probability (MAP)*), indicada na equação (9). 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 condicionadada a x. $\hat{P}$ é utilizado ao invés de P porque geralmente não se sabe o valor exato das probabilidades, elas são estimadas a partir dos dados de treinamento.

$$c_{map} = \arg\max_{c \in C} \hat{P}(c|x) \tag{9}$$

A equação (10) aplica o Teorema de Bayes para probabilidades condicionadas.

[7] `http://xapian.org/`
[8] `http://lucene.apache.org/`

Na prática, apenas o numerador da fração interessa, visto que o denominador é constante para todas as classes, portanto não afeta o $argmax$ (equação (11)).

$$c_{map} = \arg\max_{c \in C} \frac{\hat{P}(x|c)\hat{P}(c)}{\hat{P}(x)} \quad (10)$$

$$= \arg\max_{c \in C} \hat{P}(x|c)\hat{P}(c) \quad (11)$$

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:

$$\hat{P}(x|c) = \hat{P}(x_1, x_2, ..., x_n|c) = \hat{P}(x_1|c) \times \hat{P}(x_2|c) \times ... \times \hat{P}(x_n|c) \quad (12)$$

Portanto, a função de decisão pode ser reescrita através da equação (13). 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 $\hat{P}(c)$.

$$c_{map} = \arg\max_{c \in C} \ \hat{P}(c) \prod_{1 \leq i \leq n} \hat{P}(x_i|c) \quad (13)$$

Os parâmetros são obtidos através da estimativa de maior probabilidade (em inglês, *maximum likelihood estimate (MLE)*), que é a frequência relativa e corresponde ao valor mais provável de cada parâmetro, de acordo com os dados de treinamento. A equação (14) 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.

$$\hat{P}(c) = \frac{N_c}{N} \quad (14)$$

As probabilidades condicionais são estimadas como a frequência relativa do atributo x em objetos que pertencem à classe c. Na equação (15), 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.

$$\hat{P}(x|c) = \frac{T_{cx}}{\sum_{x' \in V} T_{cx'}} \quad (15)$$

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 (13)), 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 (15):

$$\hat{P}(x|c) = \frac{T_{cx} + 1}{\sum_{x' \in V} T_{cx'} + 1} \tag{16}$$

Apesar de a independência de atributos não ser verificada para a maioria dos domínios de aplicação, na prática o Naive Bayes apresenta resultados satisfatórios. [Zhang 2004] atribui a supreendente boa performance deste método ao fato de que pode haver dependência forte entre dois atributos, desde que as dependências sejam distribuídas igualmente nas classes. Segundo este argumento, a mera existência de dependências entre atributos não prejudicaria a classificação, mas sim o modo como as dependências estão distribuídas ao longo das classes.

Variantes do modelo naïve Bayes

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

O primeiro modelo utiliza uma representação que de certa forma preserva informações espaciais sobre o objeto. Na classificação de documentos de texto, por exemplo, o modelo gera um atributo que corresponde a um termo do vocabulário para cada posição do documento. Já o modelo de *Bernoulli* produz um indicador para cada possível atributo (no caso de texto, termo do vocabulário), onde 1 representa a presença e 0 a ausência do atributo no objeto.

A escolha da representação de documentos adequada é uma decisão crítica no projeto de um classificador. 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. A figura 2 ilustra as peculiaridades de cada representação.

Seleção de atributos

Uma grande quantidade de atributos a ser considerados resulta em alta complexidade computacional, além de geralmente mascarar a presença de ruídos nos dados, que prejudicam a acurácia da classificação. A fim de amenizar estes problemas, é comum que seja feita uma seleção de atributos (em inglês, *feature selection*), que consiste em escolher alguns atributos que ocorrem no conjunto de treinamento e utilizar apenas estes como parâmetro para a classificação. Esta seleção equivale à substituição de um classificador complexo por um mais simples. Especialmente quando a quantidade de dados de treinamento é limitada, modelos mais fracos são preferíveis [Manning et al. 2009].

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:

Figura 2. Exemplo ilustrativo dos modelos multinomial e de Bernoulli

Informação mútua. Análise do 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, que acontece quando um objeto apresenta o atributo se e somente se o objeto pertence à classe;

Independência de eventos. Aplicação do teste estatístico X^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].

2.3.4. 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 com características semelhantes.

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, sendo composta por k objetos.

A determinação da vizinha está diretamente relacionada com o conceito de similaridade entre objetos e a medida de distância adotada, que vai depender do domínio da

aplicação e conjunto de dados. Algumas medidas comuns para distância entre dois objetos X e Y, representados por seus vetores $\vec{X} = (x_1, ..., x_n)$ e $\vec{Y} = (y_1, ..., y_n)$, são apresentadas nas equações na tabela 1.

Tabela 1. K-NN: Medidas de distância entre objetos

Distância euclidiana	$d(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}$
Similaridade de cossenos	$sim(X,Y) = \frac{\vec{X} \cdot \vec{Y}}{\lvert\vec{X}\rvert\lvert\vec{Y}\rvert} = \frac{\sum_{1 \leq i \leq n} x_i \times y_i}{\sqrt{\sum_{1 \leq i \leq n} x_i^2} \times \sqrt{\sum_{1 \leq i \leq n} y_i^2}}$
Coeficiente de Pearson	$p(X,Y) = \frac{\sum_{1 \leq i \leq n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{1 \leq i \leq n} (x_i - \bar{x})^2} \times \sqrt{\sum_{1 \leq i \leq n} (y_i - \bar{y})^2}}$

Após o cálculo de distância e definição de vizinhança, a classe mais frequente entre os k vizinhos é atribuída ao novo objeto. Neste ponto, a similaridade também pode ser usada como grau de influência entre os objetos. Os objetos mais semelhantes geralmente têm peso maior no cálculo da classificação.

O k-NN também é sensível a ruídos, como atributos que prejudicam a acurácia de classificação quando são considerados. Portanto sua performance também é beneficiada pelo processo de seleção de atributos descrito na seção 2.3.3.

2.3.5. Apriori

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 regras de associação, que são implicações na forma $X \Rightarrow Y$, 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, enquanto para uma regra de associação $X \Rightarrow Y$, é o suporte do conjunto de itens $X \cup Y$. A *confiança* é medida pela frequêcia de Y nos registros que contém X, representando o grau de co-ocorrência de X e Y.

O *Apriori* é um algoritmo 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 sub-conjunto S de um comjunto frequente F, gerar a regra $S \Rightarrow F - S$ e testar seu valor de confiaça.

O *Apriori* foi o primeiro algoritmo eficiente para tratar do sub-problema *(1)*, que de fato é o mais desafiador. Uma possível solução ingênua 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 ilustra através de um diagrama de hasse o conjunto das partes do universo de itens $U = \{x, y, z\}$.

Figura 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 (figura 4). 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 fato de que qualquer subconjunto de um conjunto frequente também é um conjunto frequente. Portanto, só devem participam 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].

Algoritmos alternativos para mineração de regras

Figura 4. Descarte de conjuntos candidatos pelo algoritmo Apriori

Apesar de apresentar um processo inovador para geração de regras de associação, o *Apriori* ainda apresenta fraquezas, entre elas a necessidade de percorrer a base de dados repetidas vezes para cálculo de suporte e confiança [Kotsiantis and Kanellopoulos 2006]. Diversas variantes do algoritmo foram propostas com o intuito de superar algumas de suas desvantagens, entre elas estão o *AprioriTid*, que utiliza uma estrutura de dados auxiliar para evitar múliplas passagens na base de dados [Agrawal and Srikant 1994]; o *FDM*, que é uma versão do *Apriori* com suporte a paralelização [Cheung et al. 1996].

Abaixo estão relacionados alguns algoritmos apontados por [Kotsiantis and Kanellopoulos 2006] como soluções eficientes alternativas ao *Apriori*.

- *FT-Tree*: uma estrutura de árvore de prefixos é utilizada para armazenar informação sobre os padrões. Os conjuntos frequentes são descobertos e com apenas duas passadas na base de dados e sem a necessidade de geração de conjuntos candidatos [Han and Pei 2000];
- *TreeProjection*: a base de dados é projetada numa árvore lexicográfica de conjuntos de itens e o cálculo de suporte é realizado numa estrutura de matriz [Agarwal et al. 2000];
- *PRICES*
- *Matrix Algorithm*

2.4. Estratégias de recomendação

Sistemas de recomendação são comumente classificados em três grandes grupos, de acordo com a estratégia utilizada para o cálculo de recomendações: *(1)* baseada em conteúdo, *(2)* colaborativa e *(3)* híbrida. [Cazella et al. 2010] propõe uma classificação um pouco mais abrangente, na qual este trabalho se baseou.

2.4.1. 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 imparcialiadade 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* [9], apresentado na figura 5.

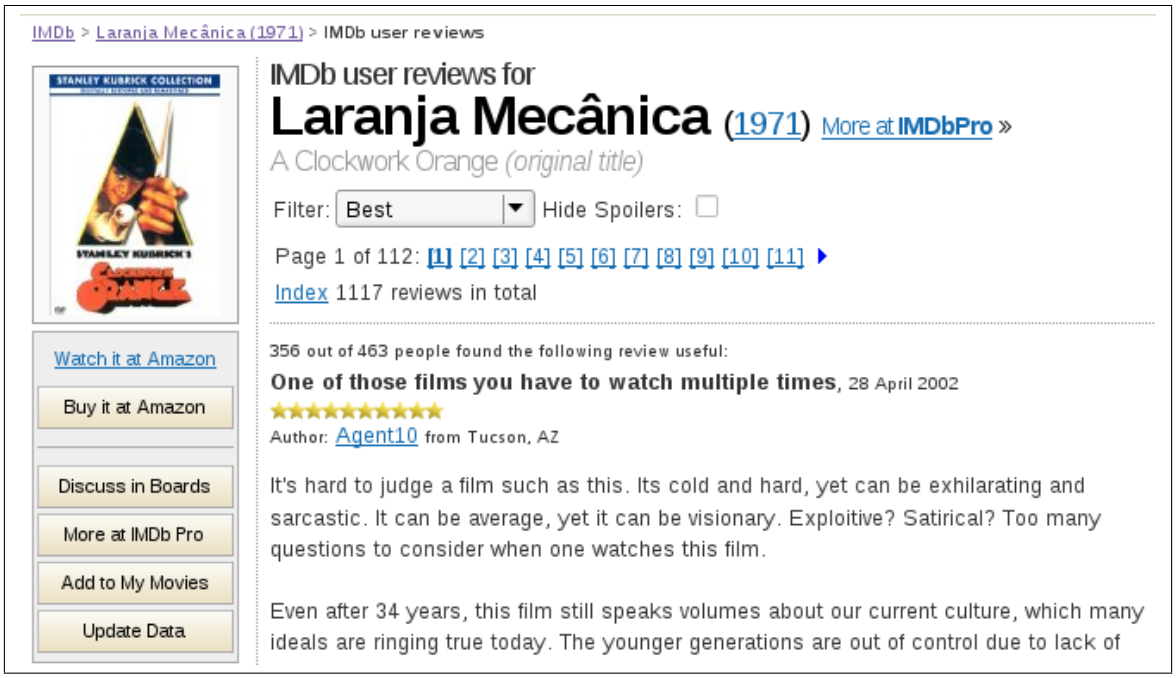

Figura 5. Avaliação de usuário no IMDb

2.4.2. 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, *naïve Bayes*) 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.

[9] `http://www.imdb.com/`

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.

2.4.3. Recomendação colaborativa

Esta estratégia não exige o reconhecimento semântico do conteúdo dos itens, pois é fundamentado na troca de experiências entre pessoas que possuem interesses em comum. A figura 6 ilustra o cenário da recomendação colaborativa.

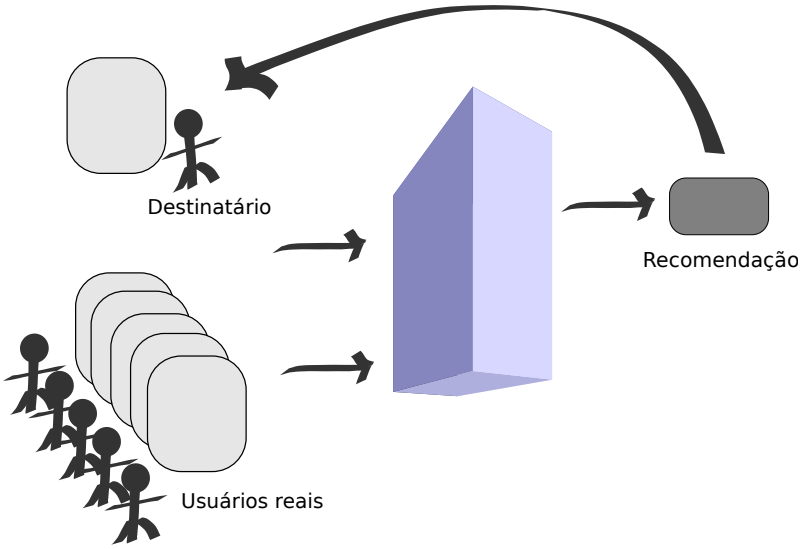

Figura 6. Cenário da recomendação colaborativa

A estratégia do K-NN é comumente utilizada neste tipo de solução. Define-se uma função que representa a distância entre os usuários. Com base nesta medida, a vizinhança de um usuário u é composta pelos que estiverem mais próximos a ele (com menor distância). Porém, ao invés de uma classe ser extraída da vizinhança, como descrito na seção 2.3.4, uma recomendação para o usuário u é produzida a partir da análise dos itens que a vizinhança de u considera 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 seus interesses semelhantes [Cazella et al. 2010].

2.4.4. Estratégia híbrida

Consiste basicamente na combinação das abordagens colaborativa e baseada em conteúdo, unindo o melhor das duas técnicas na tentativa de eliminar as fraquezas de cada uma [Cazella et al. 2010].

A figura 7 ilustra algumas possibilidades de combinação.

Figura 7. Recomendação híbrida

2.4.5. Regras de associação

A estratégia baseada em regras de associação tem como premissa o registro das escolhas dos usuários ao longo do tempo e utiliza técnicas de mineração de dados para extrair correlações e padrões frequentes no comportamento dos mesmos. Tal abordagem é frequentemente utilizada em recomendações implícitas, por exemplo, a definição do posicionamento de produtos numa prateleria ou a realização de propagandas dirigidas [Hegland 2003].

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"*.

As técnicas mais utilizadas na implementação desta classe de recomendadores são variações do algoritmos Apriori, apresentado na seção 2.3.5 [Kotsiantis and Kanellopoulos 2006]. Um exemplo de recomendação por associação é encontrado na loja virtual da empresa *Amazon*[10] (figura 8).

2.5. Avaliação de sistemas de recomendação

A avaliação de sistemas de recomendação não é uma tarefa trivial, principalmente visto que não há consenso sobre quais atributos devem ser observados e quais métricas devem ser adotadas para cada atributo. Ademais, diferentes estratégias podem funcionar melhor ou pior de acordo com o domínio da aplicação e as propriedades dos dados

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

Figura 8. Recomendação por associação na Amazon

[Herlocker et al. 2004]. 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 que o processo de avaliação tenha início com a compreensão das ações para as quais o sistema foi projetado (ver seção 2.1), como guia para as decisões metológicas ao longo dos testes.

2.5.1. Seleção dos dados

A escolha do conjunto de dados adequado é um fator chave para uma investigação consistente. [Herlocker et al. 2004] classificam as avaliações de sistemas de recomendação 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.

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 2.6. 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 e os dados são coletados na medida em que são produzidos. 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].

2.6. Métricas

Acurácia de predição. Medem o quanto pontuações previstas pelo recomendador se aproximam das pontuações reais dos usuários. Exemplos: Erro absoluto médio, erro quadrático médio, erro absoluto médio normalizado.

Acurácia de classificação. Medem a frequência com a qual o sistema faz classificações corretas no que tange a relevância dos itens. Exemplos: precisão, lembrança, medida F1 e curva ROC.

Além da acurácia. Cobertura, curva de aprendizado, novidade e surpresa, confiança, avaliação do usuário.

3. Distribuições GNU/Linux

Em 1983 Richard Stallman criou o projeto GNU[11] com o objetivo principal de desenvolver um sistema operacional livre em alternativa ao UNIX[12] – solução comercial amplamente difundida na indústria – e que fosse compatível com os padrões POSIX[13][14]. 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. Nem o próprio Linus imaginava que aquele projeto, desenvolvido sem grandes pretensões, teria a dimensão do que hoje conhecemos 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 (*General Public License*[15]) 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 distribuições GNU/Linux, que são variações do sistema operacional composto por milhares de aplicativos majoritariamente desenvolvidos pelo projeto GNU e pelo kernel Linux.

Distribuições GNU/Linux, como Debian, Fedora, Mandriva e Ubuntu, oferecem diferentes "sabores" do sistema operacional, constituídos por aplicativos selecionados

[11] `hhttp://www.gnu.org`

[12] `http://www.unix.org/`

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

[14] `http://standards.ieee.org/develop/wg/POSIX.html`

[15] Suporte legal para a distribuição livre de softwares

por seus desenvolvedores. As distribuições reduzem a complexidade de instalação e atualização do sistema para usuários finais [Cosmo et al. 2008]. Os mantenedores da distribuição atuam como intermediários entre os usuários e os autores dos softwares (chamados de *upstreams*), através do encapsulamento de componentes de software em abstrações denominadas *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, mas 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 ficam sob a responsabilidade da equipe que a desenvolve, com diferentes níveis de interferência da comunidade, como descrito acima. Este é um ponto chave no desenvolvimento, pois é um dos fatores que mais influenciam a escolha dos usuários pela distribuição. O impacto que tal seleção tem para a comunidade de usuários 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].

Softwares adicionais para atender a demandas específicas dos usuários devem ser instalados pelos mesmos, após a configuração do sistema operacional. A infraestrutura de instalação de softwares provida pela distribuição, geralmente baseada em pacotes, facilita este processo [Cosmo et al. 2008]. Ainda assim, a seleção dos programas é de responsabilidade do usuário do sistema.

3.1. Escolha da plataforma

A escolha da distribuição na qual o desenvolvimento do trabalho será baseado foi pautada pelos seguintes critérios: *(1)* Existência de um esquema consistente de distribuição de software; *(2)* Existência de dados estatísticos referentes ao uso de pacotes por usuários e possibilidade de acesso aos mesmos; *(3)* Possibilidade de integração dos resultados do trabalho com os serviços dispobinilizados pela distribuição; *(4)* Popularidade da distribuição.

As seguir estão descritos os fatores que contribuiram para a opção pelo Debian GNU/Linux como plataforma de desenvolvimento.

1. O gerenciamento de pacotes em sistemas Debian GNU/Linux é realizado através do *APT (Advanced Packaging Tool)*[16]. Ações como a busca, obtenção, instalação, atualização e remoção de pacotes são disparadas pelo *APT*, que num nível mais

[16] `http://wiki.debian.org/Apt`

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.

2. O *Popcon (Popularity Contest)*[17] 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. Segundo o *contrato social Debian*[18], o desenvolvimento do projeto é guiado pelas necessidades dos usuários e da comunidade. Portanto, as iniciativas de colaboradores individuais, sejam eles desenvolvedores oficiais ou não, serão igualmente consideradas e passarão a fazer parte da distribuição se seguirem os princípios do projeto e forem considerados úteis para a comunidade.
4. 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[19], classifica o Debian GNU/Linux entre as 10 distribuições mais populares[20]. O Debian aparece na quinta posição em suas estatísticas de páginas visitadas [21]. Já o *Linux Counter*[22] apresenta o Debian como a segunda distribuição mais popular entre as máquinas cadastradas que rodam o kernel Linux (16%), ficando atrás apenas do Ubuntu[23] (24%), que é uma distribuição derivada do Debian. Nas pesquisas da *W³Techs* sobre tecnologias para serviços web[24], 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[25], 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 púlicas. 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].

Diante do exposto, optou-se pelo Debian GNU/Linux como ambiente de desenvolvimento, todavia, os resultados poderão facilmente ser adaptados para outros con-

[17] `http://popcon.debian.org`
[18] `http://www.debian.org/social_contract.pt.html`
[19] Consulta realizada em 24 de janeiro de 2011.
[20] `http://distrowatch.com/dwres.php?resource=major`
[21] `http://distrowatch.com/stats.php?section=popularity`
[22] `http://counter.li.org/reports/machines.php`
[23] `http://www.ubuntu.com/community/ubuntu-and-debian`
[24] `http://w3techs.com/technologies/history_details/os-linux`
[25] `http://www.perrier.eu.org/weblog/2010/08/07#devel-countries-2010`

textos, desde que as informações necessárias para o cálculo de recomendações estejam disponíveis.

4. Recomendação nas distribuições

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].

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[26], de reunir, organizar e diponibilizar as informações ou meta-dados concernentes a esta estrutura [Nussbaum and Zacchiroli 2010].

A tabela 2 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, mas ao passo que se mostraram úteis e eficientes foram absorvidas pela comunidade de usuários e desenvolvedores.

5. Desenvolvimento do trabalho

Nesta seção serão apresentadas as fases do desenvolvimento deste trabalho, de acordo como estão planejadas até presente data.

5.1. Caracterização do problema

Neste trabalho propõe-se desenvolver soluções para o problema da recomendação no contexto de componentes de sofware, em especial no âmbito de distribuições GNU/Linux. Neste cenário os pacotes são modelados como itens e os usuários da distribuição como clientes do recomendador.

Embora já seja pauta de discussões entre desenvolvedores um esquema de pontuação de pacotes como medida de avaliação pessoal 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 a recomendação é caracterizada 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 sugestão de pacotes, que representam pacotes de potencial interesse para tal usuário. Desta forma, a ação principal do sistema é definida como *encontrar itens relevantes* (ver seção 2.1).

[26] `http://qa.debian.org`

Tabela 2. Recomendadores no 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

Neste trabalho as recomendações devem ser geradas a partir do comportamento do usuário. Não serão consideradas informações registradas no BTS, fóruns ou em listas de discussão. Pretende-se calcular a recomendação a partir de dados do *Popcon*, que contém listas de pacotes instalados de milhares de sistemas em produção, e do *Debtags*, como fonte de metadados sobre os pacotes. O uso de perfis também está sendo considerado, através do preenchimento de um formulário simples com macro-temas de interesse do usuário.

Uma característica peculiar da recomendação neste contexto é que, diferentemente de outras instâncias do problema, onde os itens não se relacionam entre si, os componentes de software que são objeto desta pesquisa podem declarar explicitamente requisitos em seu conteúdo. Requisitos positivos representam relações de dependência, enquanto que os negativos caracterizam as relações de conflito entre os pacotes, podendo ser definidos em diversos graus: dependência, sugestão, recomendação, conflito, substituição, quebra etc [Abate et al. 2009]. Por exemplo, se um componente A depende de B, significa dizer 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 programas pode provocar um comportamento anômalo ou até comprometer o funcionamento de todo o sistema.

O sistema de gerenciamento de pacotes (no caso do Debian, o *APT*), ao receber um pedido de modificação da configuração do sistema – por exemplo, a instalação de um novo componente – 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.

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. Uma exceção seria o caso em que as características intrínsecas do componente sejam a causa da recomendação e o usuário possa se interessar apenas pela dependência.

Outro ponto a considerar no caso de busca por associação 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 da dependência. Por outro lado, teriam grande valor as regras de associação mineradas a partir da análise dos dados estatísticos relacionando componentes que não apresentam relação alguma de dependência. Neste cenário, novos graus de relacionamento poderiam ser estabelecidos, baseados não na dependência mas na colaboração dos componentes.

5.2. Metodologia

A primeira fase deste trabalho foi dedicada ao estudo dos métodos comumente utilizados para a produção de recomendação em diversos domínios de aplicação. Nesta seção serão apresentados quais estratégias e técnicas foram selecionadas e estabelecidas como meta de desenvolvimento da presente proposta.

5.2.1. 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 questões relativas à privacidade[27], 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.

5.2.2. Seleção de atributos

A seleção de atributos geralmente proporciona ganhos na eficiência e qualidade das recomedações, na medida em que diminui o ruído e a ordem de grandeza dos dados. Existem peculiaridades do domínio de componentes de software que se consideradas podem ampliar estes ganhos.

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 todos eles podem ser desconsiderados sem prejuízo.

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[28], 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.

Na última etapa da seleção de atributos serão utilizados métodos clássicos apresentados na seção 2.3.3.

5.2.3. Estratégias de recomendação

(a) **Baseada em conteúdo**
Pretende-se implementar as técnicas *BM25* e *naïve Bayes* 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

[27] `http://popcon.debian.org/FAQ`
[28] `http://popcon.debian.org/README`

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 caso, 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**
A primeira estratégia híbrida a ser implementada será uma recomendação colaborativa a partir das *tags* dos pacotes de cada usuário (como representação de conteúdo). Outras soluções híbridas podem ser projetadas à medida que os recomendadores das estratégias puras forem sendo implementados e posteriormente combinados.

Estratégias mistas podem considerar também novos fatores para composição das sugestões, alguns dos quais são descritos abaixo.

Áreas de interesse: a composição de um perfil de usuário que indique suas áreas de interesse diminuiria a ocorrência de anomalias como, por exemplo, a indicação de bibliotecas de desenvolvimento de software para um usuário que não é programador;

Popularidade: pacotes que figuram entre os mais populares no *Popcon* devem receber maior pontuação nos cálculos, ainda que este seja apenas um critério de desempate;

Bugs: muitos bugs abertos são um indicativo de problemas, 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.

5.3. Codificação

O desenvolvimento de software será majoritariamente realizado na linguagem de programação *Python*[29], principalmente pela facilidade de integração com outras ferramentas do Debian também desenvolvidas nesta liguagem. Ademais, a vasta documentação e grande variedade de bibliotecas de utilidade para o trabalho, a exemplo da *NLTK*[30] e *Xapian*[31], são fatores que contribuiram para esta escolha.

5.4. Avaliação

Diante da inexistência de dados que possibilitem a análise *offline* de acurácia das recomendação, 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*[32].

[29] `http://www.python.org/`
[30] `http://www.nltk.org/`
[31] `http://xapian.org/`
[32] `http://www.limesurvey.org/`

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;
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 das métricas apresentadas na seção 2.6.

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.

5.5. Plano de execução

O desenvolvimento deste trabalho está previsto para acontecer entre os meses de janeiro e julho de 2011. Considerando o macro objetivo do projeto, metas intermediárias foram estabelecidas e apresentadas na tabela 3.

Tabela 3. Plano de execução

Atividade	***Janeiro***	***Fevereiro***	***Março***	***Abril***	***Maio***	***Junho***	***Julho***
Preparação e realização da qualificação	X	X					
Implementação de diferentes estratégias e técnicas	X	X	X	X			
Testes e ajustes na implementação				X	X		
Preparação e aplicação do *survey*					X	X	
Escrita da dissertação					X	X	X

6. Conclusão

A conclusão do trabalho será pautada pela análise dos resultados obtidos com a aplicação do *survey*, seguida por uma proposta de integração de um recomendador com a infraestrutura de pacotes do projeto Debian.

Referências

[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.

[Agarwal et al. 2000] Agarwal, R. C., Aggarwal, C. C., and Prasad, V. V. V. (2000). A tree projection algorithm for generation of frequent itemsets. *Journal of Parallel and Distributed Computing*, 61:350–371.

[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`.

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

[Cazella et al. 2010] Cazella, S. C., Reategui, E. B., and Nunes, M. A. (2010). A Ciência da Opinião: Estado da Arte em Sistemas de Recomendação. In *JAI: Jornada de Atualização em Informática da SBC*, pages 161–216.

[Cheung et al. 1996] Cheung, D. W., Han, J., Ng, V. T., Fu, A. W., and Fu, Y. (1996). A fast distributed algorithm for mining association rules. In *Proceedings of the fourth international conference on Parallel and distributed information systems*, DIS '96, pages 31–43. IEEE Computer Society.

[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`.

[Han and Pei 2000] Han, J. and Pei, J. (2000). Mining frequent patterns by pattern-growth: methodology and implications. *SIGKDD Explor. Newsl.*, 2:14–20.

[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 Part 2. *Inf. Process. Manage.*, 36:809–840.

[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.

[Shardanand and Maes 1995] Shardanand, U. and Maes, P. (1995). Social information filtering: Algorithms for automating 'word of mouth'. In *Proceedings of the Conference on human factors in Computing Systems*.

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

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

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

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

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