

Sistema que Recomenda Sistema: uma Solução para Recomendação de Componentes de Software

Tássia Camões Araújo ¹

¹Instituto 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, oferecendo 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 quantidade 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 Wikipedia¹, repositórios para compartilhamento de fotografia e vídeo, como Flickr² e Youtube³, entre outros. Considerando a produção em termos de software, observa-se o exemplo das comunidades de software livre, 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 sentido 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 maneiras distintas para usuários diferentes, a partir da identificação de interesses pessoais.

Este trabalho se insere no contexto de desenvolvimento de componentes de software, 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

¹<http://wikipedia.org>

²<http://flickr.com>

³<http://youtube.com>

para suprir suas necessidades pessoais. Diante da enorme quantidade de software disponíveis, 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]. Tradicionalmente as recomendações eram produzidas exclusivamente por especialistas na área em que se pretendia recomendar. A recomendação de filmes em cartaz é um exemplo deste tipo de indicação, publicada por críticos de arte 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 feitas pelos próprios usuários.

2.1. 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 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.

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.

[Perfis de usuário e privacidade]

2.2. 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 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 importância, 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 sua relevância 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, mais potencialmente relevantes para o 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. Porém, em situações onde se observa que as mesmas consultas são aplicadas repetidamente de tempos em tempos, é interessante que o sistema dê suporte a consultas permanentes. Desta forma a computação relativa a consulta pode ser realizada previamente, e apresentada sempre que um usuário requisitar. 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 o problema de classificação. Em linhas gerais, a classificação tem por objetivo, dado um conjunto de classes, determinar a qual (ou quais) classe(s) um determinado objeto pertence.

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 2.3.

2.2.1. 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 válido saber qual 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. Então 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 de t em d , mas este valor pode ser normalizado, por exemplo, pelo tamanho do documento.

O conjunto de pesos determinado pelos *tf*s dos termos de um documento pode ser visto como um resumo quantitativo do documento. Essa visão do documento é conhecida na literatura como “saco de palavras” (em inglês, “*bag of words*”), onde a disposição das palavras no documento é ignorada e apenas a quantidade de ocorrências para cada termo é mantida.

Por outro lado, alguns termos tem pouco ou nenhum poder de discriminação na determinação de relevância de um documento e a medida *tf* não considera isto. Por exemplo, numa coleção de documentos sobre filmes, o termo *diretor* aparece em quase todos os documentos. O idf_t (*inverse document frequency*) foi então introduzido para atenuar o efeito destes termos no cálculo, 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 .

$$idf_t = \log \frac{N}{df_t} \quad (1)$$

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 do teor do documento. Artigos e pronomes, por exemplo, normalmente figuram nesta categoria. A figura 1 apresenta um exemplo de cálculo de pesos para um conjunto de documentos.

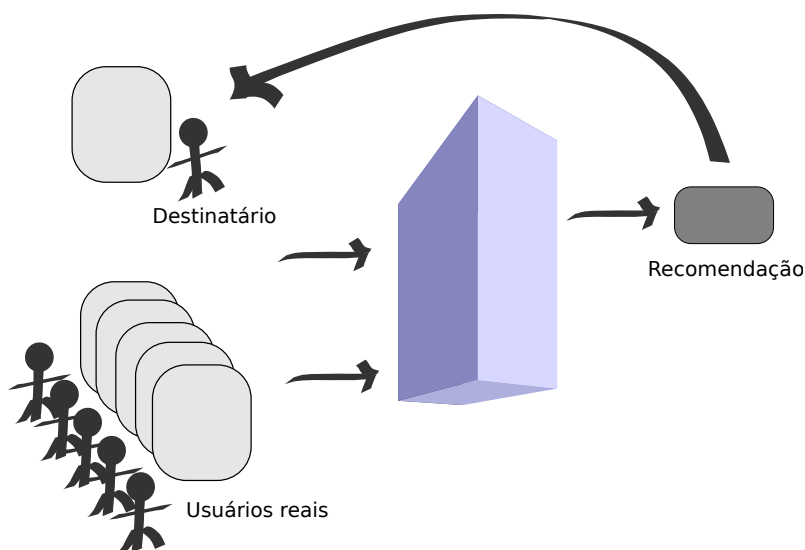


Figura 1. Exemplo de cálculo de pesos para um conjunto de documentos

Existem diversas variantes para o cálculo da medida $tf-idf_{t,d}$, propostas pouco a pouco pela comunidade de recuperação de informação com o intuito de resolver algumas anomalias e melhorar os resultados de busca. Por exemplo, é pouco provável que o aparecimento de uma palavra 20 vezes num documento tenha de fato 20 vezes mais representatividade que uma ocorrência única. Uma abordagem alternativa é incorporar o logaritmo no cálculo do tf . Outras modificações comuns são normalizações por diversas medidas: pelo comprimento do documento, pelo comprimento médio dos documentos da coleção, pelo tf máximo ou médio entre os tfs 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 (2).

$$R_{d,q} = \sum_{t \in q} tfidf_{t,d} \quad (2)$$

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.

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

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 similares.

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* e é amplamente utilizada em soluções para recuperação da informação.

No modelo de espaço vetorial, não apenas documentos da coleção, mas também as *queries* podem ser representadas como vetores, se consideradas como documentos pequenos. Portanto a similaridade de cossenos também 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 consulta.

2.2.2. Okapi BM25

O modelo Okapi *BM25* figurou por diversos anos como o melhor esquema de pesos probabilísticos nas avaliações do TREC⁴ [Jones et al. 2000]. É uma evolução de outros dois esquemas, *BM11* e *BM15* (*BM* de “*Best Match*”). Tem esse nome porque o primeiro sistema no qual foi implementado se chamava Okapi.

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

⁴O *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.

referenciados como WT , WD e WQ .

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

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.

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 (como no modelo *BM11*). Valores menores reduzem este efeito. Se $b = 0$, o comprimento do documento não afeta a pontuação final, como acontece no modelo *BM15*.

O segundo componente do peso (WD)

O idf_t é comumente calculado como na equação (7), onde N é o número total de documentos na coleção e df_t é o número de documentos que contém o termo t .

$$WD_1 = \log \frac{N}{df_t} \quad (5)$$

$$WD_2 = \log \frac{N - df_t + 0.5}{df_t + 0.5} \quad (6)$$

$$WD_3 = \log \frac{(r + 0.5)(N - df_t - R + r + 0.5)}{(R - r + 0.5)(df_t - r + 0.5)} \quad (7)$$

Consultas longas:

$$WQ = \frac{(k_3 + 1)qt f_{t,q}}{k_3 + qt f_{t,q}} \quad (8)$$

De forma análoga ao $tf-idf$, o cálculo no modelo *BM25* da pontuação do documento d para a *query* q pode ser realizado através da fórmula (9).

$$R_{d,q} = \sum_{t \in q} WT_{t,d} \times WD \times WQ \quad (9)$$

Atualmente já existem modelos que superaram o *BM25* em alguns testes mas este continua sendo o modelo mais popular, pelo menos no que diz respeito a implementações

livres. O Xapian⁵ e o Lucene⁶, bibliotecas livres para construção de ferramentas de busca, são exemplos de projetos de grande destaque na comunidade que utilizam o *BM25* como medida de pesos [Betts 2007] [Pérez-Iglesias et al. 2009].

2.2.3. Classificador bayesiano

O problema de recomendação pode ser visto como uma classificação entre duas classes, onde os itens devem ser categorizados em relevantes ou irrelevantes (os relevantes seriam recomendados). Na disciplina de inteligência artificial, a classificação é tratada com algoritmos de aprendizado de máquina. Naive Bayes é uma solução que figura entre os algoritmos de aprendizado supervisionado, onde o classificador é construído a partir de um conjunto de exemplos para treinamento. Cada exemplo traz um objeto classificado e seu conjunto de atributos.

Portanto, desde que os itens possuam metadados que os caracterize (atributos do objetos), um classificador bayesiano pode ser usado como base para recomendação.

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 equivalente no modelo de espaço vetorial, onde cada atributo é representado na matriz por uma dimensão ortogonal a todas as outras [Manning et al. 2009].

O modelo probabilístico de um classificador é a condicional $p(C|x_1, \dots, x_n)$, onde C representa a classe e x_i os n atributos. De acordo com o Teorema de Bayes, a probabilidade de um objeto $X = (x_1, \dots, x_n)$ ser da classe C é

$$p(C|x_1, \dots, x_n) = \frac{p(x_1, \dots, x_n|C) \times p(C)}{p(x_1, \dots, x_n)} \quad (10)$$

Na prática, apenas o numerador da fração acima interessa, visto que o denominador é constante para todas as classes – não depende de C e todos os valores de x_i são dados. O numerador é equivalente ao modelo de probabilidade conjunta: $p(C, x_1, \dots, x_n)$, que pode ser reescrito da seguinte maneira, de acordo com a definição de probabilidade condicional:

$$p(C, x_1, \dots, x_n) = p(C)p(x_1, \dots, x_n|C) = p(C)p(x_1, \dots, x_n, C)/p(c) \quad (11)$$

Definição de probabilidade condicional: $p(A|B) = \frac{p(A,B)}{p(B)}$

Apesar de esta suposição não ser verificada para a maioria dos domínios de aplicação, na prática o Naive Bayes apresenta resultados satisfatórios. [Zhang 2004] atribui a surpreendente boa performance deste método ao fato de que pode haver dependência

⁵<http://xapian.org/>

⁶<http://lucene.apache.org/>

forte entre dois atributos, desde que as dependências sejam distribuídas igualmente nas classes. Segundo este argumento, é a distribuição das dependências entre atributos ao através das classes que afeta a classificação, não as dependências por si só.

Now the "naive" conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for $i \neq j$. This means that

for $i \neq j$, and so the joint model can be expressed as

This means that under the above independence assumptions, the conditional distribution over the class variable C can be expressed like this:

where Z (the evidence) is a scaling factor dependent only on C , i.e., a constant if the values of the feature variables are known. Models of this form are much more manageable, since they factor into a so-called class prior $p(C)$ and independent probability distributions $p(F_i | C)$. If there are k classes and if a model for each can be expressed in terms of r parameters, then the corresponding naive Bayes model has $(k - 1) + n r$ parameters. In practice, often $k = 2$ (binary classification) and $r = 1$ (Bernoulli variables as features) are common, and so the total number of parameters of the naive Bayes model is $2n + 1$, where n is the number of binary features used for classification and prediction

Seleção de atributos

2.2.4. K-NN

K-nearest neighbors (k-NN), em português *k vizinhos mais próximos* é outro algoritmo de aprendizado supervisionado para classificação.

pearson correlation p/ achar a vizinhanca - dependencia linear entre duas variaveis (em estatística) seleção de atributos

2.2.5. Agrupamento

Agrupamento, *clustering* em inglês, é uma técnica de aprendizado de máquina não supervisionado, ou seja, tenta identificar algum padrão de organização dos dados sem que haja uma classificação prévia dos exemplos.

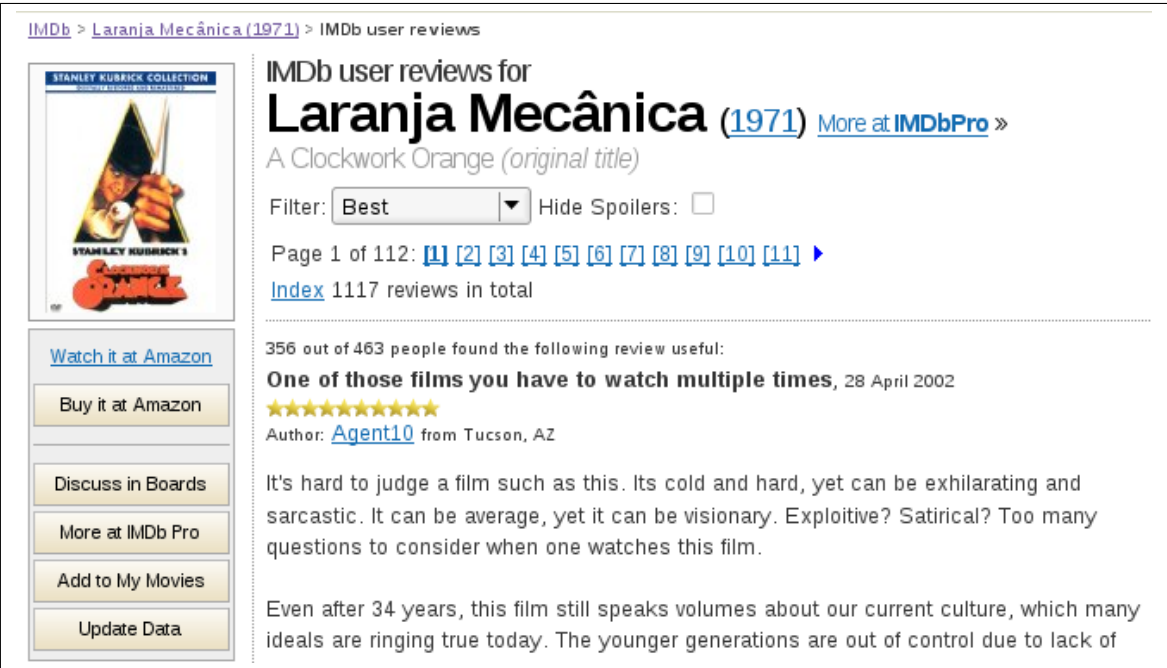
k-means - k sementes, agrupamento por similaridade, cálculo de novos centróides

2.3. 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. Neste trabalho porém será apresentada a classificação proposta por [Cazella et al. 2010], que é um pouco mais detalhada.

2.3.1. Reputação do item

Bastante 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 e 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, o que não pode ser verificado devido a seu caráter subjetivo e estritamente pessoal. Atualmente existem serviços especializados em reputação de produtos, que não têm venda associada, apenas disponibilizam as avaliações. É o caso do *Internet Movie Database* ⁷, apresentado na figura 2.



IMDb > [Laranja Mecânica \(1971\)](#) > IMDb user reviews

IMDb user reviews for
Laranja Mecânica (1971) [More at IMDbPro](#) »
A Clockwork Orange (original title)

Filter: **Best** Hide Spoilers: ☐

Page 1 of 112: [\[1\]](#) [\[2\]](#) [\[3\]](#) [\[4\]](#) [\[5\]](#) [\[6\]](#) [\[7\]](#) [\[8\]](#) [\[9\]](#) [\[10\]](#) [\[11\]](#) ▶
[Index](#) 1117 reviews in total

356 out of 463 people found the following review useful:

One of those films you have to watch multiple times, 28 April 2002
★★★★★
Author: [Agent10](#) from Tucson, AZ

It's hard to judge a film such as this. Its cold and hard, yet can be exhilarating and sarcastic. It can be average, yet it can be visionary. Exploitive? Satirical? Too many questions to consider when one watches this film.

Even after 34 years, this film still speaks volumes about our current culture, which many ideals are ringing true today. The younger generations are out of control due to lack of

[Watch it at Amazon](#)
[Buy it at Amazon](#)
[Discuss in Boards](#)
[More at IMDb Pro](#)
[Add to My Movies](#)
[Update Data](#)

Figura 2. Avaliação de usuário no IMDb

2.3.2. Baseada em conteúdo

Este método 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 classificação 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 para definir a semelhança entre os mesmos. No caso de uma livraria, por exemplo, sugerir ao cliente outros livros do mesmo autor de livros previamente selecionados é uma boa estratégia.

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. Outro problema indicado por [Adomavicius and Tuzhilin 2005] é o da

⁷<http://www.imdb.com/>

superespecialização, onde apenas itens similares aos já escolhidos pelos usuários são recomendados.

2.3.3. Colaborativa

Esta abordagem não exige a compreensão ou reconhecimento do conteúdo dos itens, pois se baseia na troca de experiências entre pessoas que possuem interesses em comum.

Define-se uma função que representa a distância entre os usuários, e a vizinhança de um usuário é composta pelos que estiverem mais próximos dele (com menor distância). Uma recomendação para o usuário u é produzida a partir da análise dos itens da vizinhança de u . Os itens que aparecerem com maior frequência na vizinhança farão parte da recomendação.

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

2.3.4. Híbrida

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

2.3.5. Associação

A partir da análise de uma base de dados que relaciona clientes e itens selecionados pelos mesmos, utiliza-se diversas técnicas para inferir associações entre os itens, como por exemplo: "Clientes que selecionaram os itens A, B e C também selecionaram o item D". As associações são caracterizadas por um suporte mínimo, que diz respeito à confiabilidade das mesmas. Portanto, quando uma associação é extraída, não significa que ela pode ser verificada em todos os casos, e sim numa porcentagem mínima, definida pelo suporte. A implementação desta estratégia é mais complexa, pois exige técnicas mais sofisticadas para identificação de padrões no comportamento dos usuários, principalmente quando se trata de um grande volume de dados. Um exemplo de aplicação desta abordagem é encontrado no site da Amazon (figura 3).

2.4. Avaliação de sistemas de recomendação

A avaliação de sistemas de recomendação não é uma tarefa trivial, principalmente porque não há consenso entre os pesquisadores sobre quais atributos devem ser observados e quais métricas devem ser adotadas para cada atributo [Herlocker et al. 2004]. Ademais, diferentes estratégias podem funcionar melhor ou pior de acordo com o domínio da aplicação e as propriedades dos dados. Por exemplo, algoritmos projetados especificamente para conjuntos de dados com um número muito maior de usuários do que de itens podem se mostrar inapropriados em domínios onde há muito mais itens do que usuários.



Figura 3. Recomendação por associação na Amazon

2.4.1. Compreensão das ações

O processo de avaliação deve ter início com a compreensão das ações e finalidades para as quais o sistema foi projetado. Abaixo estão relacionadas algumas das ações 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 decidirem quais mensagens valiam a pena serem lidas dentro de cada contexto.

Encontrar itens relevantes . O sistema sugere itens para o usuário através de uma lista produtos ordenados por uma pontuação proporcional à probabilidade do item ser considerado relevante pelo usuário. Esta é a finalidade mais comum relacionada a sistemas de recomendação, que atrai grande parte das pesquisas relacionadas com o tema e está presente na maioria dos sistemas de recomendação comerciais.

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

Sequência recomendada . Quando não somente quais itens mas a ordem em que eles são apresentados importa, 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 . Recomendadores são normalmente avaliados de acordo com o quanto eles ajudam um usuário a tomar uma decisão de consumidor. Muitos usuários porém utilizam estes sistemas mesmo quando não têm planos de consumir nenhum produto, apenas para navegar através dos produtos. Neste caso, aspectos como a interface, facilidade de uso e natureza da informação provida são de extrema importância.

2.4.2. Seleção dos dados

Além da identificação das ações, outro fator importante para uma avaliação consistente é a escolha do conjunto de dados adequado. Em certos casos a avaliação pode ser realizada *offline* em cima de uma base de dados existente, já em outros, experimentos com usuários *ao vivo* são mais apropriados.

As análises *offline* geralmente são objetivas, com foco na acurácia das predições. Desde que existam dados prévios de avaliações dos usuários, diferentes estratégias podem ser utilizadas para prever outros dados, e os resultados são analisados utilizando-se uma ou mais métricas que serão apresentadas nas próximas seções. Desta forma pode-se avaliar de maneira rápida e pouco custosa diferentes conjuntos de dados e algoritmos. Em contrapartida, tais análises são prejudicadas pela esparsidade dos dados. Não se pode, por exemplo, avaliar a acurácia da recomendação de um item para um usuário se não existe uma pontuação prévia do usuário para tal item.

Por outro lado, nos experimentos *ao vivo*, seja num ambiente controlado ou em campo, os sistemas são disponibilizados para uma comunidade de usuários cujo comportamento é analisado no que diz respeito aos efeitos do sistema. Neste tipo de experimento, além de análises objetivas como a acurácia, pode-se avaliar fatores como a performance, participação e satisfação dos usuários.

Quando não existem dados prévios disponíveis ou quando existem porém não são considerados 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 podem se ajustar melhor para um algoritmo do que para outros [Herlocker et al. 2004].

2.5. Métricas

[Detalhar cada item]

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, 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 GNU⁸ com o objetivo de desenvolver um sistema operacional livre como alternativa ao UNIX⁹ – solução comercial amplamente difundida na indústria – e que fosse compatível com os padrões POSIX¹⁰ – família de

⁸<http://www.gnu.org>

⁹<http://www.unix.org/>

¹⁰<http://standards.ieee.org/develop/wg/POSIX.html>

normas definidas pelo IEEE com foco na portabilidade entre sistemas operacionais. 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. Porém, o núcleo do sistema (*GNU Hurd*) teve desenvolvimento mais lento.

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 agilizar 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*¹¹) e as equipes dos dois projetos começaram a trabalhar na adaptação do kernel Linux para o ambiente GNU. Este esforço conjunto possibilitou o surgimento das distribuições GNU/Linux, que são variações do sistema GNU/Linux.

Distribuições como Debian, Fedora, Mandriva e Ubuntu, oferecem diferentes soluções de sistema operacional compostas pelo GNU/Linux e mais uma gama de softwares selecionados pela comunidade ou empresa responsável por seu desenvolvimento. 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 chamadas *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 conhecido como catedral [Raymond 1999]. Por outro lado, os projetos criados independentemente, formam ao longo do tempo uma comunidade de desenvolvedores interessados em colaborar, e estes são os únicos responsáveis pelo sucesso ou fracasso do projeto. Neste modelo, que recebe o nome de 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 Raymond 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 a cargo da equipe que a desenvolve, com diferentes níveis de interferência da comunidade, como vimos acima. Este é um ponto chave no desenvolvimento, pois é um dos fatores que mais influenciam a escolha dos usuários pela distribuição. Um exemplo do impacto que tal seleção têm na comunidade foi a recente polêmica gerada pelo anúncio de Mark Shuttleworth, criador do Ubuntu, de que o projeto abandonaria o Gnome como interface padrão do usuário [Paul 2010]. No caso do Debian, uma decisão como esta seria resul-

¹¹ Suporte legal para a distribuição livre de softwares

tado de longo período de discussão na lista debian-desktop. Polêmicas no Debian não são raras, por exemplo, quando o assunto é o tema do desktop padrão (papel de parede, cores e estilo das janelas, etc) que será lançado na próxima versão estável.

Softwares adicionais que atendem a necessidades específicas devem ser instalados pelo usuário ou administrador do sistema, após a configuração do sistema operacional. Este processo é facilitado pela infraestrutura de instalação de softwares provida pela distribuição (baseada em pacotes), mas a seleção dos programas fica a cargo do usuário.

3.1. Contexto do trabalho

A escolha da distribuição na qual o desenvolvimento do trabalho seria baseado foi pausada 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 disponibilizados pela distribuição; (4) Popularidade da distribuição. O Debian GNU/Linux foi selecionado pelos fatores apresentados a seguir.

1. O gerenciamento de pacotes em sistemas Debian GNU/Linux é realizado através do *APT (Advanced Packaging Tool)*. Ações como a busca, obtenção, instalação, atualização e remoção de pacotes são disparadas pelo APT, que num nível mais baixo faz uso do *dpkg*. O APT também gerencia conflitos entre pacotes, checando se novos pacotes a serem instalados conflitam com algum outro do sistema, e dependências, instalando todos os programas definidos como pré-requisitos antes da instalação de um pacote.
2. O *Popcon (Popularity Contest)*¹² é um 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 seu *contrato social perante a comunidade de software livre*¹³, o desenvolvimento do projeto Debian é guiado pelas necessidades dos usuários e da comunidade de software livre. Portanto, as iniciativas de colaboradores individuais, seja ele desenvolvedor oficial ou não, serão igualmente consideradas e passarão a fazer parte da distribuição se seguirem aqueles princípios e forem considerados de utilidade 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 671 distribuições em sua base de dados, classifica o Debian GNU/Linux entre as 10 distribuições mais populares¹⁴. O Debian aparece na quinta posição em suas estatísticas de páginas visitadas¹⁵. Já o *Linux Counter*¹⁶ apresenta o Debian como a segunda distribuição

¹²<http://popcon.debian.org>

¹³http://www.debian.org/social_contract.pt.html

¹⁴<http://distrowatch.com/dwres.php?resource=major>

¹⁵<http://distrowatch.com/stats.php?section=popularity>

¹⁶<http://counter.li.org/reports/machines.php>

mais popular entre as máquinas cadastradas que rodam o kernel Linux (16%), ficando atrás apenas do Ubuntu (24%). Nas pesquisas da *W³Techs* sobre tecnologias para serviços web¹⁷, 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 1500 desenvolvedores oficiais, 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].

Diante do exposto, optou-se pelo Debian GNU/Linux como ambiente de desenvolvimento, todavia, os resultados poderão facilmente ser adaptados para outros contextos, desde que as informações necessárias para o cálculo de recomendações estejam disponíveis (dados estatísticos).

4. Recomendação nas distribuições

Diante da complexa e crescente estrutura do projeto Debian, observa-se um esforço por parte dos desenvolvedores, principalmente da equipe responsável pelo controle de qualidade¹⁸, de reunir, organizar e disponibilizar as informações ou meta-dados concernentes a esta estrutura [Nussbaum and Zacchiroli 2010]. A tabela 1 relaciona algumas destas iniciativas que estão diretamente ligadas ao gerenciamento de pacotes. Vale ressaltar que a maioria destas soluções foram inicialmente desenvolvidas num contexto extra-oficial, mas a medida que se mostraram úteis e eficientes foram absorvidas pela comunidade de usuários e desenvolvedores.

[Investigar iniciativas externas ao Debian: central de aplicativos ubuntu]

5. Metodologia

O desenvolvimento deste trabalho será pautado pelos seguintes passos: compreensão do problema; estabelecimento de estratégias para resolver o problema; desenvolvimento; análise dos resultados. Nas seções seguintes são apresenta

5.1. Caracterização do problema

Ao trazer o problema da recomendação para o contexto de distribuições GNU/Linux, considera-se que os componentes de software ou pacotes são itens e os usuários da distribuição são clientes. Em termos de entrada e saída do sistema de recomendação: Dada a lista de pacotes de um determinado usuário (destinatário da recomendação), deve-se retornar uma lista de pacotes recomendados, que representam pacotes de potencial interesse para o usuário destinatário.

¹⁷http://w3techs.com/technologies/history_details/os-linux

¹⁸<http://qa.debian.org>

Tabela 1. Soluções de recomendação no Debian

<i>Solução</i>	<i>Descrição</i>	<i>Estratégia de recomendação</i>
BTS	Sistema de rastreamento de bugs, que é alimentado pelos usuários	Reputação
Popcon	<i>Popularity Contest</i> publica diariamente o resultado do concurso de popularidade entre pacotes	
Packages.qa	Criado pelo time de qualidade, reúne informações relativas à manutenção do pacote	
UDD	<i>Ultimate Debian Database</i> é outra iniciativa do time de qualidade, que reúne informações de diversos aspectos do Debian numa base de dados única. Usuários avançados podem consultar esta base para tomar decisões acerca de que pacotes usar.	
Debtags	Classificação de pacotes em produção, porém, sem esquema de recomendação	Baseada em conteúdo
Debtags	No cadastro de tags para pacotes, novas tags são sugeridas a partir das tags já associadas ao pacote	Associação

Outro ponto a ser ressaltado é que as recomendações devem ser geradas a partir do comportamento do usuário (ação), e não da opinião. Neste trabalho não serão consideradas informações registradas por usuários no BTS 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 meta-dados sobre os pacotes.

Uma característica importante desta instância do problema de recomendação é que, diferentemente da situação usual onde os itens não se relacionam entre si (produtos na prateleira do supermercado, por exemplo), os componentes de software objetos desta pesquisa (pacotes debian) podem declarar explicitamente requisitos em seu conteúdo [Abate et al. 2009]. Requisitos positivos representam relações de dependência enquanto que os negativos representam conflitos, que podem ser definidos em diversos graus: dependência, sugestão, recomendação, conflito, substituição, quebra, etc. Por exemplo, se um componente *a* depende de *b*, significa dizer que *b* deve estar 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.

As relações entre os componentes são consideradas pelo sistema de gerenciamento de pacotes (no Debian, o APT), que recebe do usuário um pedido de modificar a instalação do sistema de alguma maneira – por exemplo, a instalação de um novo componente – e tenta satisfazer o pedido a partir do conhecimento de onde os componentes se encontram (repositórios de pacotes) e das relações entre os componentes. O gerenciador então promove a instalação de todas as dependências de um pacote antes de instalá-lo, e não permite a instalação de pacotes que conflitem com outros já instalados no sistema.

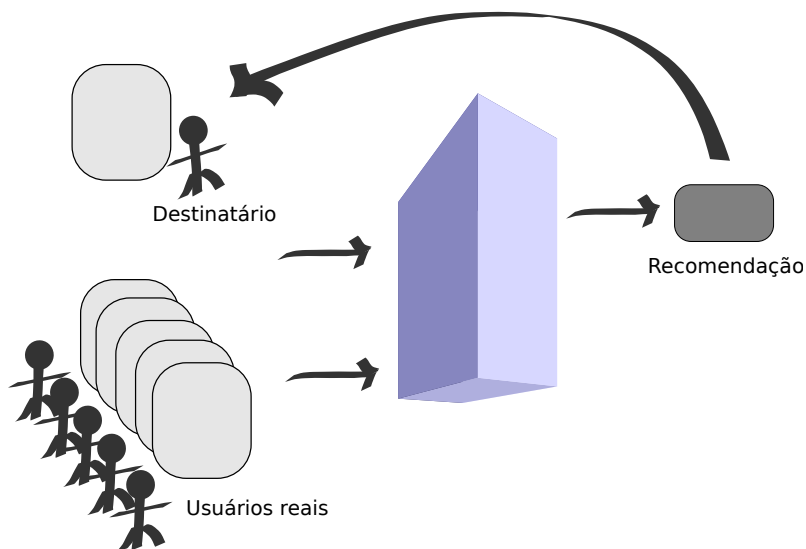


Figura 4. Cenário de uma Recomendação

Portanto, o cálculo da recomendação de pacotes deve levar em conta as relações de dependência entre pacotes e desconsiderar pacotes dependentes nas buscas por associações, visto que as dependências de um pacote sempre (ou quase sempre) estão presentes nos sistemas com tal pacote instalado.

5.2. Desenvolvimento de estratégias

Para o cálculo da recomendação pretende-se experimentar as estratégias:

Baseada em conteúdo a partir da classificação provida pelo *debtags*;

Colaborativa a partir dos dados do *popcon*;

Híbrida com uma mesclagem das duas abordagens.

Na implementação de tais estratégias, pretende-se utilizar algumas das técnicas mencionadas na seção 2.2 e fazer uma análise comparativa para avaliar qual delas é mais viável para o sistema em produção.

5.3. Desenvolvimento

O desenvolvimento de software será majoritariamente realizado na linguagem de programação Python e C, pela disponibilidade de bibliotecas de utilidade para o trabalho.

5.4. Avaliação

De acordo com os pontos destacados na seção 2.4, o plano de avaliação do sistema de recomendação deve ser definido com base na identificação da ação principal do sistema, que para este trabalho é *encontrar itens relevantes*.

Como já mencionado, a construção do recomendador será baseada nos dados do *Popcon* e *Debtags*. Todavia os dados do *Popcon* devem ser pré-processados a fim de serem utilizados pelos algoritmos de recomendação. As listas de pacotes enviadas pelos usuários contém todos os pacotes do sistema, entre aplicativos, bibliotecas e até o kernel Linux. Para este estudo, apenas os aplicativos para usuário final são relevantes, pois são

eles que devem compor a recomendação. Os pacotes que fazem parte do sistema básico também devem ser desconsiderados, pois em tese todos os usuários já os possuem visto que são essenciais para o funcionamento de qualquer sistema Debian. Na literatura este pré-processamento é denominado de *seleção de atributos*, que é realizado com o intuito de diminuir o ruído nos dados e reduzir sua ordem de grandeza.

Diante disponibilidade limitada de dados, que inviabiliza a medição *offline* da acurácia de recomendação, optou-se por realizar experimentos *ao vivo* com usuários, por meio de um *survey* eletrônico. Pretende-se disponibilizar um site onde o usuário seja guiado nos seguintes passos:

1. O usuário envia a sua lista de pacotes instalados, como representação de identidade;
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, além de explicações acerca do processo realizado;
4. O usuário avalia as recomendações apresentadas;
5. Com base na avaliação do usuário, aplicar as seguintes métricas: erro absoluto médio, precisão, lembrança, novidade e surpresa e confiança.

Ao final da etapa de avaliação, os dados de avaliação submetidos por todos os usuários serão compilados em gráficos e comentários apresentados na versão final da dissertação 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 de desenvolver um sistema de recomendação de pacotes, metas intermediárias foram estabelecidas e apresentadas na tabela 2.

Tabela 2. Plano de execução

<i>Atividade</i>	<i>Janeiro</i>	<i>Fevereiro</i>	<i>Março</i>	<i>Abril</i>	<i>Maior</i>	<i>Junho</i>	<i>Julho</i>
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 <i>survey</i>					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 do sistema de recomendação com a infraestrutura 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 Infrastructure.
- [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.
- [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.
- [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.
- [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.
- [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.ars>.

- [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.