Recomendação em um Sistema de Gerenciamento de Chamados

Pedro Henrique Frozi de Castro e Souza[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

phfcsouza@inf.ufrgs.br

Resumo. *Este artigo apresenta um trabalho completo da aplicação de um sistema de recomendação em um software já existente, utilizando dados reais extraídos da industria. Utilizamos uma técnica de recomendação que usa filtragem baseada em conteúdo, apresentando todas as etapas necessárias para a sua aplicação no modelo real. Como resultado, este sistema foi aplicado com sucesso em um sistema de gerenciamento de chamados.*

1. Introdução

Atualmente, grande parte das empresas que desenvolvem sistemas customizados possuem uma variedade de ferramentas de relacionamento com os clientes. Este canal de comunicação é imprescindível para a manutenção, suporte e evolução do software, satisfazendo as constantes necessidades dos clientes. Essas ferramentas geralmente são conhecidas como *Help Desks* e permitem que usuários do sistema interajam de alguma forma com a empresa que criou o software. Estes sistemas podem possuir interações mais simples, como por exemplo a troca de mensagens entre cliente e fornecedora, como também podem possuir estruturas mais complexas, como por exemplo as que possibilitam gerar demandas para sistemas customizados. Dentre estas demandas, temos: Solicitação de melhorias, Implementações Legais, Suportes, Bugs, etc.

Muitas vezes, estes sistemas servem como principal fonte de conhecimento da empresa, sendo utilizado como ferramenta para análise de requisitos, código fonte, mudanças arquiteturais, etc. Este trabalho propõe utilizar esta fonte de conhecimento para, através de algoritmos usados para sistemas de recomendação, criar um modelo capaz de agrupar e recomendar informações históricas similares deste sistema.

2. Trabalhos Relacionados

Sistemas de recomendação voltado às atividades do ciclo de um *software* tem se

popularizado muito nos ultimos anos, funcionando como uma alternativa para auxiliar no trabalho de análise, desenvolvimento e testes. Em [Anvik, Hiew and Murphy, 2006], um trabalhos de vanguarda na área, os autores mostraram como utilizar algoritmos de classificação de texto em um sistema de gerenciamento de *bugs* para, durante o processo de triagem de *bugs,* indicar colaboradores mais aptos à resolvê-los. Em [Liu, Beng and Zhang, 2013], os autores buscam através de técnicas de apredizado de máquina, melhorar as buscas por *bugs* duplicados em um sistema de gerenciamento de *bugs*, auxiliando também o processo de triagem. Já o trabalho realizado em [Miranda, Aranha and Iyoda, 2012] propõe utilizar métodos de sistemas de recomendação para o encaminhamento de determinados casos de teste para os perfis de *testers* disponíveis.

3. Background

3.1. Sistema de Gerenciamento de Chamados

Para este trabalho, utilizamos uma plataforma de gerenciamento de chamados já existente[1] para aplicar as técnicas de recomendação. Esta plataforma consiste em sistema para o gerenciamento de customizações, correções e solicitações de suporte para equipes pequenas de desenvolvimento, auxiliando principalmente no reuso de produtos já utilizados pelos clientes. Nesta ferramenta, o cliente pode criar solicitações que serão atendidas por uma equipe de desenvolvimento em um determinado fluxo de avaliação.

Figure 1. Tela principal do sistema *Metis*, onde é mostrado para o usuário a lista de chamados abertos.

1 *Metis Project*, trabalho acadêmico de uma disciplina de graduação.

O sistema foi desenvolvido em Node.js[2], utilizando MongoDB[3] para o armazenamento de dados.

A estrutura das solicitações foram iniciamente construídas baseadas em um sistema de gerenciamento de chamados já utilizado no mercado. Esta estrutura agrupa os principais campos necessários para que cliente e empresa realize as interações. São eles: *Título*, campo de texto que corresponde ao título da solicitação; *Descrição*: Descrição completa da solicitação; *Tipo*: Tipo de solicitação realizada, podendo ser categorizada como melhorias, suportes, implementações legais e *Bug's*; *Usuário*: Usuário que criou o chamado.

O conjunto de textos formados pela descrição será o nosso *corpus*, e cada elemento deste conjunto será chamado de *documento* [Tobergte, 2013]. O documento possuirá informações que descrevem que tipo de problema ou solução o usuário está buscando, bem como os pontos de alteração do sistema que o usuário está solicitando. Abaixo podemos ver um exemplo de documento:

Tabela 1. Exemplo de documento que representa a solicitação.

Tipo	Título	Detalhamento
BUG	ERRO AO ENVIAR ANÁLISE	ERRO RELATADO PELO USUÁRIO: PROPOSTA ANTIGA, AO GRAVAR NA ETAPA 3 PARA ENVIAR PARA ANÁLISE OCORREU O ERRO ´CARO USUÁRIO OCORREU UM ERRO NO PROCESSAMENTO DE SUA SOLICITAÇÃO, PEDIMOS DESCULPAS PELO INCOVENIENTE, UMA MENSAGEM COM ERRO FOI AUTOMATICAMENTE ENVIADA PARA EQUIPE DE SUPORTE, POR FAVOR, SE O ERRO PERSISTIR ENTRE EM CONTATO CONOSCO. VER DETALHES.´ PRINT EM ANEXO.

Para a aplicação das técnicas que serão vistas na seção 4, foram necessárias algumas alterações na estrutura do documento que representa o chamado. Tivemos que incluir um *array* que representa as palavras encontradas no documento, chamada na estrutura de *ownTerms*. Também foi necessário criar um *array* de frequências, onde cada elemento representa a quantidade de vezes que determinada palavra do *corpus* aparece no documento. Abaixo podemos ver a estrutura do chamado após a alteração:

2 https://nodejs.org/.
3 https://www.mongodb.org/.

Tabela 2. Código que representa o objeto *Chamado* no sistema.

```
var reqSchema = new mongoose.Schema({
    title        : String,
    description  : String,
    priority     : Number,
    created      : { type: Date, default: Date.now },
    modified     : { type: Date, default: Date.now },
    tags         : [String],
    modules      : [String],
    category     : [String],
    type         : String,
    user         : {type: mongoose.Schema.ObjectId, ref: 'User'},
    whoSucceeded : {type: mongoose.Schema.ObjectId, ref: 'User'},
    forum        : [{type: mongoose.Schema.ObjectId, ref: 'Forum'}],
    visits       : { type: Number, default: 0 },

    ownTerms     : [String],
    freqTerms    : [Number]
});
```

3.2. Sistemas de Recomendação

Os sistemas de recomendação auxiliam na eficácia do processo natural de indicação do ser humano. A ideia é utilizar dados históricos baseados em preferência de usuário, dados de um conteúdo acessado, etc.

Existem dois tipos principais de técnicas utilizadas em sistemas de recomendação [Hopmans, 2015]:

- Filtragem Colaborativa: O sistema recomendará à um determinado perfil de usuário itens que foram avaliados por outro usuário de perfil similar;
- Filtragem de Baseada em Conteúdo: O sistema recomendará determinadas informações para o usuário baseado nas informações nas quais ele já avaliou.

Na imagem abaixo, podemos compreender melhor o funcionamento das duas principais técnicas utilizadas em sistemas de recomendação:

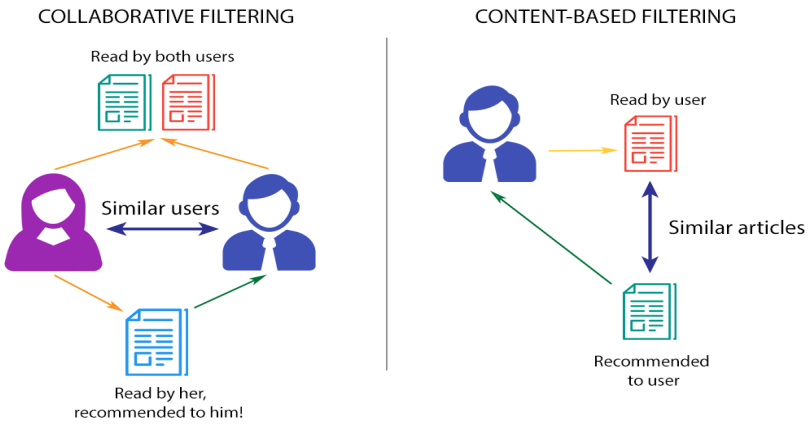

Figure 2. Quadro comparativo entre as duas principais técnicas de sistemas de recomendação.[4]

4 Imagem extraída de https://www.themarketingtechnologist.co/building-a-recommendation-engine-

Este trabalho utilizará uma técnica de filtragem baseada em conteúdo, que será extraído dos documentos gerados a partir do detalhamento dos chamados.

4. Abordagem

A abordagem utilizada neste trabalho utiliza as etapas de extração e pré-processamento do processo de mineração de dados para criação dos documentos que serão utilizados [Becker and Tumiman 2013]. Depois, utilizaremos um processo de vetorização dos documentos extraídos, considerando a representação numérica gerada para realização do cálculo de similaridade entre os documentos.

4.1. Extração

Os dados utilizados neste trabalho foram extraídos de um sistema de *helpdesk* proprietário, sendo importados para o sistema *Metis* um total de 1685 chamados. Todos os chamados foram criados por um cliente específico entre janeiro de 2014 e janeiro de 2016. Estes dados foram organizados na estrutura de chamados definida para o sistema, tendo os campos de título, descrição e tipo preenchidos com valores reais.

Tabela 3. Quantidade de chamados importados por tipo identificado.

Tipo	Quantidade	%
BUG	860	51.01%
SUPORTE	368	21.83%
CORREÇÕES LEGADO	227	13.46%
MELHORIAS	198	11.74%
ORÇADOS	33	1.96%

Após a extração, identificamos um *corpus* com 41630 palavras, tendo aproximadamente 25 palavras por documento.

4.2. Pré-Processamento

Na etapa de pré-processamento, iremos identificar as palavras relevantes de cada um dos documentos do *corpus*, para a partir disso, criar o dicionário de termos que será necessário para as próximas etapas.

Remoção de Termos Comuns

Existem diversas estratégias para melhor selecionar os termos que serão realmente úteis em cada um dos documentos [Manning 2008]. Desta forma, é possível diminuir a quantidade de termos que deverão ser indexados no processo de *tokenização*, que extrairá os termos dos documentos [Manning 2008]. Neste trabalho, aplicamos os seguintes tratamentos para cada documento do *corpus*:

for-geek-setting-up-the-prerequisites-13/.

- Remoção de *Stopwords*;
- Remoção de caracteres especiais;
- Remoção de acentos das letras.

Não foi realizada uma remoção total dos números existentes no *corpus* devido à existência de códigos que serão relevantes à recomendação realizada, como por exemplo: código de leis; resoluções, circulares e regimentos de órgãos reguladores; etc.

Tokenização

Após a redução da quantidade de palavras, realizamos a *tokenização* dos documentos, transformando-os em vetores de palavras.

Descrição do Documento	Tokens Extraídos
POR FAVOR, HABILITAR NOVAMENTE A OPÇÃO DE IMPRIMIR RELATÓRIOS DO MÓDULO CDC EM RICH TEXT.	["FAVOR", HABILITAR", "NOVAMENTE", "OPCAO", "IMPRIMIR", "RELATORIOS", "MODULO", "CDC", "RICH", "TEXT"]

Tabela 4. Exemplo de tokens gerados para um determinado documento.

Dicionário de Termos

O dicionário de termos é gerado a partir dos *tokens* encontrados para os documentos, representando todos palavras relevantes existentes no *corpus*. Portanto, para sua concepção, é necessário percorrer cada um dos *tokens* gerados, verificando se o termo já existe no dicionário: Se existir, desconsideramos o *token*; caso não exista, incluímos o *token* no dicionário de termos. No final do processamento, encontramos 3621 termos diferentes.

4.3. Vetorização

Na etapa de vetorização criamos o modelo numérico do documento. Neste caso, utilizamos um modelo multinomial [Sá 2008], onde o nosso documento é representado por um vetor de frequências dos termos do dicionário. O algoritmo abaixo mostra como o cálculo de frequências foi realizado:

Tabela 5. Pseudo-código que representa o cálculo das frequências.

```
1. Para cada documento d em Corpus
2.        Para cada termo t em Dicionario
3.               Para cada token tk em d.Tokens
4.                      Se t == tk Então
5.                             d.VetorDeFrequencias[t]++
6.                      fim se
7.               fim tk
8.        fim t
9. fim d
```

4.4. Similaridade

O cáculo da similaridade entre os documentos é realizado a partir dos seus vetores de frequêcia. Existem diversas maneiras de calcular a similaridade entre os documentos, como por exemplo: produto escalar, distância euclidiana, distância euclidiana normalizada, diferença de cossenos (ângulo entre vetores), etc [Manning 2008]. Neste trabalho utilizamos a distância euclidiana. Basicamente, para cálcular a distância euclidiana basta calcular a *norma* entre os dois vetores considerados:

$$N = Quntidade\ de\ termos\ no\ dicionário$$

$$Similaridade(Doc1, Doc2) = || doc2 - doc1 ||$$

$$= [(termo1_{doc2}\text{-}termo1_{doc1})^2 + ... + (termoN_{doc2}\text{-}termoN_{doc1})^2]^{1/2}$$

```
1. Similaridade(doc1, doc2) <- Retorno
2.        SomaQuadrados <- 0
3.        Para cada termo t em Dicionario
4.              SomaQuadrados <- SomaQuadrados +
5.              (doc1.VetorDeFrequencias[t] - doc2.VetorDeFrequencias[t])^2
6.        fim t
7.        Retorno <- RaizQuadrada(SomaQuadrados)
8. fim Similaridade
```

Tabela 6. Pseudo-código que representa o cálculo da similaridade entre dois documentos.

O sistema de recomedação calcula todas as similaridades entre os documentos, retornando os *K* mais próximos ao documento que está sendo analisado pelo usuário.

5. Avaliação

Como neste tarbalho apresentamos uma recomendação baseada em conteúdo de textos, onde os dados impordados não possuiam uma rotulação inicial, para medir a qualidade das recomendações geradas seria necessário realizar um processo de rotulação manual. Neste processo, o usuário poderia avaliar cada um dos *K* documentos que foram recomendados informando se os mesmos foram úteis ou não à sua consulta.

6. Conclusão e Trabalhos Futuros

Durante o estudo realizado para o desenvolvimento deste trabalho, verifiquei que as técnicas utilizadas em sistemas de recomendação atuais já estão muito mais avançadas em comparação com a técnica aqui apresentada. Para o modelo de documento proposto neste trabalho, a maior parte delas acaba utilizando um modelo híbrido de filtragem, que utiliza filtragem colaborativa além da filtragem baseada em conteúdo apresentada.

Além disso, o algoritmo utilizado para o cálculo da similaridade não é um dos mais indicados na literatura, que sugere principalmente o método da diferença entre cossenos dos vetores de frequência. A distância euclidiana pode ser ineficaz para comparação entre textos que possuem termos em comum, mas em frequencias muito distintas.

Não obstante, o trabalho deixa uma série de lacunas que podem ser preenchidas futuramente, como a inclusão da filtragem colaborativa e a melhora no algoritmo utilizado para cálculo da similaridade. Além disso, também podemos incluir um modelo de classificação baseada em *tags* que poderiam melhorar a qualidade das recomendações.

Além disso, os dados levantados neste trabalho também podem ser explorados nos campos de triagem de bugs, chamados e também nos processos de alocação de recursos.

Por fim, concluo que obtive um relativo sucesso, visto que este trabalho tinha como objetivo aplicar alguma técnica de sistemas de recomendação no âmbito da engenharia de software (SE), proposta da disciplina de *Têndencias em SE* para a graduação.

7. Referências

[Anvik, Hiew and Murphy, 2006] Anvik, J., Hiew, L., & Murphy, G. C. (2006). Who should fix this bug? Proceeding of the 28th International Conference on Software Engineering – ICSE. ’06, 2006, 361. http://doi.org/10.1145/1134285.1134336

[Liu, Beng and Zhang, 2013] Liu, K., Beng Kuan Tan, H., & Zhang, H. (2013). Has this bug been reported? Proceedings - Working Conference on Reverse Engineering, WCRE, 82–91. http://doi.org/10.1109/WCRE.2013.6671283

[Miranda, Aranha and Iyoda, 2012] Miranda, B., Aranha, E., & Iyoda, J. (2012). Recommender systems for manual testing: deciding how to assign tests in a test team. International Symposium on Empirical Software Engineering and Measurement, 201–210. http://doi.org/10.1145/2372251.2372289

[Hopmans, 2015] Hopmans, T. (2015). A recommendation system for blogs: Setting up the prerequisites (part 1). Retrieved June 23, 2016, from https://www.themarketingtechnologist.co/building-a-recommendation-engine-for-geek-setting-up-the-prerequisites-13/

[Tobergte, 2013] Tobergte, D. R., & Curtis, S. (2013). TopCat: Data Mining for Topic Identification in a Text Corpus Chris. Journal of Chemical Information and Modeling, 53(9), 1689â†1699. http://doi.org/10.1017/CBO9781107415324.004

[Becker and Tumiman 2013] BECKER, K. ; TUMITAN, D. “Introdução à Mineração de Opiniões: Conceitos, Aplicações e Desafios”. In: Joao Eduardo Ferreira. (Org.). Lectures of the 28th Brazilian Symposium on Databases. 1ed.Pernambuco: CIN - UFPE, 2013, v. , p. 27-52.

[Manning 2008] Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich. (2008) Introduction to Information Retrieval, Cambridge University Press.

[Sá 2008] Sá, H. R. de. (2008). Seleção De Características Para Classificação De Texto.

Bortis, G., & Van Der Hoek, A. (2013). PorchLight: A tag-based approach to bug triaging. Proceedings - International Conference on Software Engineering, 342–351. http://doi.org/10.1109/ICSE.2013.6606580