# Mineração de Informação em Linguagem Natural para Apoiar a Elicitação de Requisitos

Roxana Lisette Quintanilla Portugal
Departamento de Informática, PUC-Rio, Rio de Janeiro, Brasil
rportugal@inf.puc-rio.br

**Abstract.** Este trabalho descreve a mineração de informações em linguagem natural a partir do repositório de projetos GitHub. É explicada como o conteúdo de projetos semelhantes dada uma busca por domínio podem ser úteis para o reuso de conhecimento, e assim, ajudar nas tarefas de Elicitação de Requisitos. Técnicas de mineração de textos, regularidades independentes do domínio, e os metadados de GitHub são os métodos utilizados para selecionar projetos relevantes e as informações dentro deles. Uma abordagem para atingir nossa meta utilizando pesquisa exploratória é explicada, bem como descrevemos os resultados alcançados.

Dissertação de Mestrado apresentada como requisito parcial
para obtenção do grau de Mestre pelo Programa de
Pósgraduação em Informática da PUC-Rio.

Orientador: Prof. Julio Cesar Sampaio do Prado Leite

Rio de Janeiro, 2016

Pontifícia Universidade Católica
do Rio de Janeiro

**Roxana Lisette Quintanilla Portugal**

# Mineração de Informação em Linguagem Natural para Apoiar a Elicitação de Requisitos

**Dissertação de Mestrado**

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática da PUC-Rio.

Orientador: Prof. Julio Cesar Sampaio do Prado Leite

Rio de Janeiro

Abril de 2016

PUC-Rio - Certificação Digital Nº 1322105/CA

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

**Roxana Lisette Quintanilla Portugal**

# Mineração de Informação em Linguagem Natural para Apoiar a Elicitação de Requisitos

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

**Prof. Julio Cesar Sampaio do Prado Leite**
Orientador
Departamento de Informática - PUC-Rio

**Prof. Daniel Schwabe**
Departamento de Informática - PUC-Rio

**Prof. Eduardo Kinder Almentero**
Departamento de Informática - UFRRJ

**Prof. Marcio da Silveira Carvalho**
Coordenador Setorial do Centro
Técnico Científico – PUC-Rio

Rio de Janeiro, 19 de abril de 2016

PUC-Rio - Certificação Digital Nº 1322105/CA

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

**Roxana Lisette Quintanilla Portugal**

Atua no mercado de Engenharia de Software desde 2006. Graduou-se em Engenharia de Sistemas pela Universidade Privada do Norte de Perú em 2008. Ingressou no Mestrado em Informática na Pontifícia Universidade Católica do Rio de Janeiro em 2013.

PUC-Rio - Certificação Digital Nº 1322105/CA

Ficha Catalográfica

Portugal, Roxana Lisette Quintanilla

Mineração de informação em linguagem natural para apoiar a elicitação de requisitos / Roxana Lisette Quintanilla Portugal ; orientador: Julio Cesar Sampaio do Prado Leite. – 2016.
96 f. ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2016.
Inclui bibliografia

1. Informática – Teses. 2. Elicitação de Requisitos. 3. Reuso de Conhecimento. 4. Extração de informação. 5. Repositórios abertos. 6. Processamento de linguagem natural. I. Leite, Julio Cesar Sampaio do Prado. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 0004

## Dedicatória

Para meu pai *Victor Raúl Quintanilla Villafuerte*, Ele acendeu uma vela pra mim no dia da minha defesa e me disse claramente o quanto ele estava contente. Isso foi o maior presente desta jornada.

PUC-Rio - Certificação Digital Nº 1322105/CA

## Agradecimentos

A nossa mãe Maria, pois quando um dia eu disse a ela “leve-me onde quiser”, ela foi muito generosa em trazer-me para Rio de Janeiro.

Em especial a minha mãe, só ela pode reconhecer quando eu realmente quero algo e não hesita em me apoiar.

Para meus irmãos que sempre me mostram outros caminhos, e me lembram de não esquecer as coisas essenciais.

A meus amigos Amilcar Prada e Alvaro Elorrieta, por facilitar estes estudos, e por me guiar quando todo era tão desconhecido.

A meu orientador, por todos os desafios que me fizeram descobrir habilidades impensadas. Agradeço muito mais, pela sua qualidade humana nos momentos incertos que acompanharam a jornada.

A Soeli Fiorini pelo apoio oportuno, e o ambiente acolhedor do LES.

Aos membros da minha banca pelos ensinamentos e paciência.

Ao grupo de Engenharia de Requisitos, cada um de vocês, desde seu próprio olhar me ensinaram que nossa área é muito rica.

A Hugo Roque e Diego Haz, amigos encontrados no ambiente particular da PUC-Rio, vocês me deram muita motivação.

A Edgard Sarmiento, pelas milhares de vezes que ele me disse que tudo ia dar certo.

A Joanna Pivatelli, pela vontade de fazer que este trabalho fique mais transparente em português.

À agencia de fomento CAPES e a PUC-Rio, por me dar a oportunidade de estudar em uma universidade tão maravilhosa.

PUC-Rio - Certificação Digital Nº 1322105/CA

PUC-Rio - Certificação Digital Nº 1322105/CA

“As palavras hão de ser como estrelas, que são muito distintas e claras, e altíssimas. Seu estilo pode ser muito claro e muito alto; tão claro que o entendam os que não sabem e tão alto que tenham muito que entender os que sabem. ”

Extraído do Sermão da sexagésima § 5. Ano 1655

Padre Antônio Vieira.

## Resumo

Quintanilla Portugal, Roxana Lisette; Sampaio do Prado Leite, Julio Cesar. **Mineração de informação em Linguagem Natural para apoiar a Elicitação de Requisitos**. Rio de Janeiro, 2016. 96p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Este trabalho descreve a mineração de informações em linguagem natural a partir do repositório de projetos GitHub. É explicada como o conteúdo de projetos semelhantes dada uma busca por domínio podem ser úteis para o reuso de conhecimento, e assim, ajudar nas tarefas de Elicitação de Requisitos. Técnicas de mineração de textos, regularidades independentes do domínio, e os metadados de GitHub são os métodos utilizados para selecionar projetos relevantes e as informações dentro deles. Uma abordagem para atingir nossa meta utilizando pesquisa exploratória é explicada, bem como descrevemos os resultados alcançados.

PUC-Rio - Certificação Digital Nº 1322105/CA

## Palavras-chave

Elicitação de requisitos; Reuso de conhecimento; Extração de informação; Repositórios abertos; Processamento de linguagem natural.

## Abstract

Roxana Lisette Quintanilla Portugal; Sampaio do Prado Leite, Julio Cesar. **Mining Information in Natural Language to Support Requirements Elicitation.** Rio de Janeiro, 2016. 96p. MSc. Dissertation - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This work describes the mining of information in natural language from the GitHub repository. It is explained how the content of similar projects given a search domain can be useful for the reuse of knowledge, and thus help in the Requirements Elicitation tasks. Techniques of text mining, regularities independent from domain, and GitHub metadata are the methods used to select relevant projects and the information within them. One approach to achieve our goal is explained with an exploratory research and the results achieved.


PUC-Rio - Certificação Digital Nº 1322105/CA


## Keywords

Requirements elicitation; Knowledge reuse; Information extraction; Open source repositories; Natural language processing.

## Sumário


1 Introdução 16

1.1. Conceitos Básicos 18

1.1.1. Elicitação de Requisitos 18

1.1.2. Universo de Informações 19

1.1.3. Fontes de Informação 20

1.1.4. Coleta de Fatos 20

1.1.5. Metáfora de GitHub 20

1.1.6. Mineração de Textos (TM) 21

1.1.7. Processamento de Linguagem Natural (NLP) 22

1.1.8. Reuso de Conhecimento 23

1.1.9. R Project 24

1.1.10. Atlas.ti 24

1.2. Desafíos da Pesquisa 24

1.3. Estratégia adotada 25

1.4. Resumo 26

2 Pesquisa Exploratória 27

2.1. Exploração do GitHub 27

2.1.1. Metáfora do GitHub 28

2.1.2. Artefato Readme 29

2.1.3. Cenários de uso do Readme 30

2.1.4. Restrições de GitHub 31

2.2. Abstração de Informações 31

2.2.1. Informação Relacionada a Requisitos 32

2.2.2. Abordagens para Filtrar Informações 32

2.3. Resumo 34

3 Coletando Informações 35

3.1. Processo para Minerar Informações 35


PUC-Rio - Certificação Digital Nº 1322105/CA

3.1.1. Collate 37
3.1.2. Distill 41
3.1.3. Collocate 42
3.2. Recuperando Informações Automaticamente 43
3.2.1. Desenho 43
3.2.2. Arquitetura 44
3.2.3. Estratégia 45
3.2.4. Relevância dos documentos 46
3.2.5. A Ferramenta 46
3.2.6. Uso de um Corpus de documentos 47
3.3. Resumo 49

4 Filtragem das informações 50
4.1. Filtragem por *POS-tagging* 50
4.1.1. Identificação de termos frequentes 50
4.1.2. Identificação de substantivos 51
4.1.3. Organização de termos frequentes 52
4.2. Filtragem por Boilerplates 53
4.2.1. O processo 54
4.2.2. Organização de projetos usando ambas filtragens 57
4.3. Resumo 58

5 Analise das Informações 59
5.1. Motivação 59
5.2. Desenho do caso de estudo 60
5.3. Primeiro Teste 61
5.4. Segundo Teste 62
5.5. Teste de precisão 62
5.6. Notas e comentários 63
5.7. Ameaças 63
5.8. Resumo 64

PUC-Rio - Certificação Digital Nº 1322105/CA

6 Conclusões e trabalhos futuros 65
6.1. Conclusões 65
6.2. Trabalhos relacionados 65
6.3. Contribuições 66
6.4. Trabalhos futuros 67

Glossário 78
Referências Bibliográficas 73

Anexo 1 Caso de estudo “music application” 78
A1.1. IRR para o piloto do caso de estudo. 78
A1.2. IRR para o Primeiro Teste 87
A1.3. IRR para o Segundo Teste 91
A1.4. IRR para o teste de precisão 93

PUC-Rio - Certificação Digital Nº 1322105/CA

## Lista de Figuras


Figura 1. Modelo SADT Engenharia de Requisitos, Leite[1]. 19
Figura 2. GitHub *schema*, Rodriguez[25] 21
Figura 3. Ordem de exploração de Artefatos em GitHub 28
Figura 4. Dados brutos de Readme no GitHub 33
Figura 5. Modelo SADT do processo, nível 0 35
Figura 6. Modelo SADT de macro atividades do processo, nível 1. 36
Figura 7. Modelo SADT de atividades do Collate, nível 2. 37
Figura 8. Script para reduzir o Corpus de Readmes 39
Figura 9. Modelo SADT de atividades do Distill, nível 2. 41
Figura 10. Rede semântica de informações relacionadas a “real estate” 43
Figura 11. Combinação de abordagens para ancorar informações 43
Figura 12. Bibliotecas “*gems*” de Ruby utilizadas. 44
Figura 13. Principais Componentes da Arquitetura 45
Figura 14. Recuperação de readmes combinando opções de ordenação 45
Figura 15. Ordenação de projetos recuperados de GitHub 46
Figura 16.Traçabilidade dos Readmes para os seus repositórios. 46
Figura 17. Aplicação Web para a construção de Corpus de Readmes 47
Figura 18. Importação de documentos no Atlas.ti 47
Figura 19. Criação de *codes* e redes semânticas. 48
Figura 20. Exemplo de *dataset* no R Project 48
Figura 21. Readmes tratados no R Project 48
Figura 22. Sumario da matriz DTM para 308 Readmes 49
Figura 23. Termos frequentes no corpus de readmes de *Real Estate* 49
Figura 24. Termos frequentes. TF (esquerda) e TF-IDF (direita) 51
Figura 25. Nuvem de substantivos mais frequentes dos textos Readme 52
Figura 26. Proposta para organizar os termos frequentes 52
Figura 29. Rupp’s Boilerplates, Arora et al. [50] 53
Figura 28. Organização de abstrações utilizando ambas filtragens 57
Figura 29. Avaliação das frases no piloto de teste 60
Figura 30. Resumo de respostas do primeiro teste 61


PUC-Rio - Certificação Digital Nº 1322105/CA

Figura 31. Segundo teste do caso “music application” 62
Figura 32. R Teste de precisão de boilerplates em vários domínios 63

PUC-Rio - Certificação Digital Nº 1322105/CA

## Lista de Tabelas


Tabela 1. Relação entre capítulos e desafios 26
Tabela 2. POS-Tags com semântica para a filtragem de IRR 33
Tabela 3. Frequência de boilerplates no *corpus* “digital library” 54
Tabela 4. Número de mostras para cada grupo de readmes 55
Tabela 5. Classificação dos readmes segundo o seu conteúdo 55
Tabela 6. Classificação dos readmes pelo seu discurso 55
Tabela 7. Boilerplates identificados e sua frequência no *corpus* 56
Tabela 8. Tags para avaliação de frases 60
Tabela 9. Valorações feitas pelos 3 membros da equipe a 10 frases 61
Tabela 10. Teste de precisão de boilerplates em vários domínios 62


PUC-Rio - Certificação Digital Nº 1322105/CA

## Lista de Abreviaturas e Siglas

| | |
|---|---|
| API | Interface de Programação de Aplicações. |
| DSL | *Domain Specific Language* |
| IRR | Informações Relacionadas a Requisitos |
| IR | *Information Retrieval* |
| IE | *Information Extraction* |
| KDT | *Knowledge Discovery* |
| LAL | Léxico Ampliado da Linguagem |
| LDA | *Latent Dirichlet Alocation* |
| LSA | *Latent semantic analysis* |
| Md | Markdown |
| MPV | *Minimum Viable Product.* |
| NLP | *Natural Language Processing* |
| POS-tagger | *Part of Speech Tagger* |
| R | Uma linguagem e ambiente para computação estatística |
| SADT | *Structured analysis and design technique* |
| TF | *Term Frequent* |
| TF-IDF | *Term Frequency–Inverse Document Frequency* |
| TM | *Text Mining* |
| UdI | Universo de Informações |
| URL | *Uniform Resource Location* |

PUC-Rio - Certificação Digital Nº 1322105/CA

# 1
# Introdução

Este trabalho apresenta a visão de trabalhar sobre repositórios de projetos *open source* e de como adotá-los como fonte de informação útil para a descoberta de informações relacionadas a requisitos (IRR), visando a reutilização de conhecimento. Neste capítulo, apresentamos a motivação da nossa pesquisa e os desafios ao abordar técnicas de mineração de textos e processamento de linguagem natural. Em seguida, apresentamos os conceitos básicos para entendimento da dissertação. Finalmente, definimos a organização deste trabalho.

O processo de produção de software tem seu início quando o que se quer é definido. Esta definição, segundo Leite[1] é o ponto de partida do processo, ou seja, a construção de software só pode ser iniciada quando se tem bem estabelecido o que se quer produzir; isto pressupõe que há um trabalho iterativo para obter conhecimento e ajudar ao cliente no que realmente requere. A Engenharia de Requisitos (ER), como subárea da Engenharia de Software, está focada especialmente nesta primeira parte do processo, na qual há interação com aqueles que demandam software (clientes), e são guiados por um sub-processo definido por Leite[1] de: elicitar, modelar, analisar, y gerenciar. Na primeira atividade (elicitar) se tornam explícitos os conhecimentos para atender os objetivos dos clientes. Leite[1] aponta três atividades que compõem a elicitação: identificação de fontes de informação, coleta de fatos, y comunicação. Nosso trabalho é direcionado para essas duas primeiras atividades. Utilizamos o repositório de projetos GitHub como fonte de informação principal, e para coletar fatos, os artefatos existentes em cada projeto.

Projetos em repositórios como o GitHub são abundantes, porém não estão sendo explorados desde a visão de descoberta e reuso de conhecimento. Artefatos do GitHub tais como o *readme*, as *issues*, e as *issues comments* possuem informações em linguagem natural relevantes para o reuso na Elicitação de Requisitos. Nesse sentido, é importante a criação de um universo de tais informações (UdI)[1] que sirva de suporte para a criação de requisitos em aplicativos similares aos existentes no GitHub. Leite[1] aponta que a técnica de leitura de documentos, permite coletar informações (fatos) ao abstrair propriedades dos textos, os quais permitem um melhor entendimento da aplicação a ser construída. Este trabalho visa ajudar no problema que o Engenheiro de Requisito enfrenta para abstrair manualmente informações de documentação de sistemas similares, a fim de abstrair fatos que ajudem a alcançar um melhor entendimento sobre um domino. Isto pode resultar numa melhor elicitação e construção de requisitos do cliente.

Uma das situações na qual nasce nossa motivação, é o caso das Startups de software. Aqui o cliente é o consumidor final que possui uma variedade de necessidades a serem atendidas por algum aplicativo. Nesse contexto, os empreendedores têm a tarefa crítica de identificar e selecionar tais necessidades que atendam um perfil de consumidor, levando em conta, também, o tempo no mercado para o lançamento de um produto inovador (*time to market*). Paternoster[2]

PUC-Rio - Certificação Digital Nº 1322105/CA

tem percebido que no campo das Startups, o uso de técnicas para a elicitação são mínimas, pois, algumas vezes as características dos aplicativos são inventadas de acordo ao mercado, a concorrência, e no que o consumidor relata acerca de suas necessidades em um determinado contexto. A invenção de requisitos é também apontado por Potts[3] que a define como a criação de *marketable features* de um produto para consumidores imaginários com a finalidade de criar um produto suficientemente útil para um grupo de pessoas similares, isto é, com um padrão de necessidades em um determinado contexto. No ambiente das Startups existe também a cultura de identificar as necessidades do consumidor através da retroalimentação após a criação de um MVP[4] (Produto Mínimo Viável). Nessa linha, tem-se criado o marco de trabalho *Lean Startup*[5], que propõe lançamentos iterativos de MVP's com o objetivo de validar e aprender continuamente do cliente. Finalmente, Santala[6] trata essa criação de requisitos baseados na retroalimentação de clientes, e a define como Engenharia de Requisitos orientada ao mercado.

Sobre o descrito anteriormente cabe perguntar-se: *Em que medida a semi-automatização da leitura de documentos em linguagem natural existentes em projetos open source, pode acelerar o processo de obtenção de conhecimento sobre o domínio de uma aplicação?*

Neste trabalho, nos inspiramos na tese *A Design Theory for Requirementes Mining Systems*[7], dado que a necessidade de minerar, tem suas origens na quantidade de informação que um Engenheiro de Requisitos deve revisar para elicitar. Meth[7] menciona a definição de Vlas & Robinson[8] sobre a **identificação de requisitos**, que é útil especialmente para dois propósitos: primeiro, separar textos que descrevem requisitos, de textos que não são relevantes desde a visão de requisitos; e, segundo, a delimitação de cada requisito dentro de um documento, resultando em várias sentenças de requisitos. Essa identificação de requisitos, na verdade, mescla-se com a própria identificação de fontes de informação, isto é, segundo Leite et al.[9], selecionar estrategicamente a fonte para elicitar requisitos. Assim, no GitHub, é necessário identificar quais artefatos podem desempenhar melhor o papel de fonte de informação para o caso em questão.

Propomos que a obtenção de conhecimento em documentação de projetos deve ser feita pela identificação de projetos similares dado uma questão de busca (*query*), e na mineração de suas informações. Apesar que a mineração em projetos *open source* desde o ponto de vista de requisitos é um tema pouco explorado, acredita-se que as abordagens existentes[10,11,12] para abstrair informações em especificações de requisitos, podem ajudar na extração de tais informações. De outro lado, as técnicas existentes para processamento de Linguagem Natural que atravessam as áreas de *Information Retrieval* (IR), *Information Extraction* (IE) e *Text Mining* (TM), podem ajudar a semi-automatizar as tarefas de mineração.

Os fatos mencionados, em conjunto com uma pesquisa exploratória[13], criaram a seguinte hipótese:

*Informações relacionadas a requisitos (IRR) extraídas de projetos similares existentes em repositórios open source podem ajudar no reuso de conhecimento para apoiar a elicitação de requisitos de software.*

PUC-Rio - Certificação Digital Nº 1322105/CA

Para testar essa hipótese, os resultados da mineração de informações de projetos de GitHub foram avaliados, desde a perspectiva de reuso de conhecimento, por um projeto de software do tipo Startup (Capitulo 5).

## 1.1. Conceitos Básicos

Neste subitem, os conceitos básicos para o entendimento da dissertação são definidos. Primeiramente, definimos o processo de Elicitação de Requisitos. Em seguida, apresentamos dos conceitos utilizados em ER, como são o Universo de Informações e a Coleta de Fatos. A seguir a Metáfora de GitHub é explicada. Os mecanismos para filtrar informações são descritas nos títulos Mineração de Textos (TM) e Processamento de Linguagem Natural (NLP). O produto de este trabalho, que são informações as IRR, são descritas no título Reuso de Conhecimento. Finalmente, a descrição da ferramenta e os pacotes de software utilizados.

### 1.1.1. Elicitação de Requisitos

Visto como processo, a elicitação se enquadra na primeira parte do processo da ER[1]: elicitar, modelar, analisar. Nesse sentido, a elicitação vem a ser um dos processos mais críticos, pois, como afirmado por Goguem & Linde[14], as tarefas da elicitação não podem ser resolvidas puramente de maneira tecnológica; isso ocorre, devido ao contexto social que é mais crucial do que nas fases de desenho ou programação. No processo de produção de software, tem-se como mantra que "O sistema atenderá as necessidades dos usuários", tal objetivo, no nosso entender, só pode ser alcançado se a ER não se limita apenas ao entendimento da interação do usuário e computador, senão também, à interação do usuário com o seu ambiente.

Das várias definições sobre requisitos, a seguinte definição de Zave & Jackson[15] representa a nossa perspectiva para este trabalho:

> **Requisitos só existem no ambiente.** A primeira distinção necessária para a ER é capturada por dois modos gramaticais. Afirmações em "modo indicativo" descrevem o ambiente tal como é na ausência da máquina ou independentemente das ações da máquina; tais afirmações são usualmente chamadas de *assumptions* ou *domain knowledge*. Afirmações em "modo optativo" descrevem o ambiente como gostaríamos que fosse e como nós esperamos que seja, quando a máquina estiver ligada ao meio ambiente. Afirmações optativas são comumente chamadas de requisitos. A habilidade para descrever o ambiente em modo optativo torna necessário a descrição de máquina.

Baseados na definição, esta pesquisa busca a extração de afirmações em **modo indicativo,** que exponha o conhecimento abstraído por aplicativos de software sobre um domínio. Existem também a possibilidade de encontrar afirmações em **modo optativo** acerca do comportamento do sistema, por exemplo, encontrar as frases do tipo: *"the system should be able to..."* entre outras.

PUC-Rio - Certificação Digital Nº 1322105/CA

### 1.1.2.
### Universo de Informações

Segundo Leite[1] o universo de informações (UdI) é o contexto geral no qual o software deverá ser desenvolvido e operado. O UdI inclui todas as fontes de informação e todas as pessoas relacionadas ao software. Essas pessoas são também conhecidas como os atores desse universo. O UdI é a realidade circunstanciada pelo conjunto de objetivos definidos pelos que demandam o software.

É usual na ER, que o UdI seja formado de documentação como questionários, entrevistas, atas, entre outros. Nossa exploração, está direcionada à abundância de documentos existente em GitHub. É importante diferenciar que, cada projeto pode conter documentos digitais próprios, mas não é garantido que isto ocorra em todos os projetos, e caso existam, a estrutura de tais documentos é diversa. Isto é, existem documentos de tipo **estruturado**, tal como um *script* de banco de dados; documentos de tipo **semiestruturado**, tal como os arquivos com marcações do tipo *html;* e documentos de tipo **não-estruturado**, como, por exemplo, os manuais de usuário. Sem embargo, o ambiente de desenvolvimento colaborativo de GitHub estimula aos projetos a criar outro tipo de documentas, que chamaremos de artefatos de aqui em diante, como são os: *readmes*, as *issues* e seus *comments*. Tais artefatos, embora não sejam obrigatórios, possuem a qualidade de ter uma função comunicativa, e uma estrutura de documento padrão. A (Figura 1) ajuda a situar o UdI nas atividades da ER:

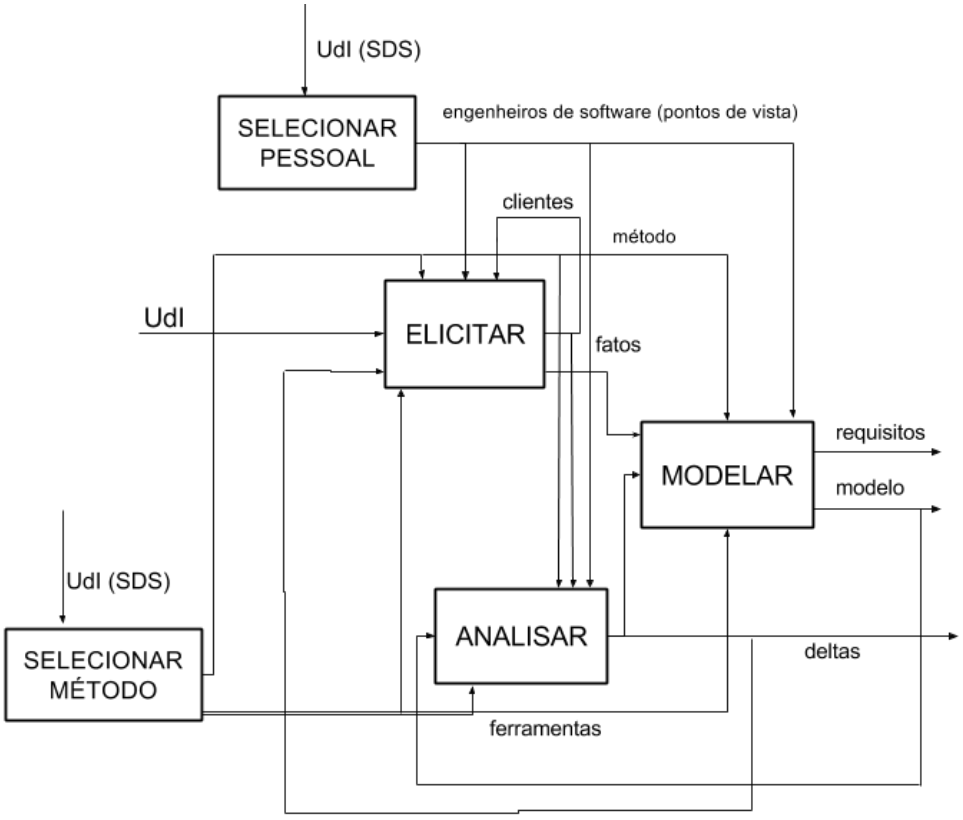



**Figura 1. Modelo SADT Engenharia de Requisitos, Leite[1].**

PUC-Rio - Certificação Digital Nº 1322105/CA

### 1.1.3.
### Fontes de Informação

Leite[1,9] e Zowghi & Colin[16] apontam como uma atividade importante do processo de elicitação, a seleção de fontes de informação. Em tal atividade, é usual que os clientes indiquem os fatos do contexto da aplicação. Sem embargo, para cada situação, outras possíveis fontes de informação devem ser identificadas, seja pela exploração própria do entorno, ou seguindo uma estratégia[9]. Para o caso dos artefatos de GitHub como fontes de informação, a definição de Zowghi & Colin[16] expõe os ganhos de explorá-los:

> **"**Documentação existente acerca de sistemas atuais e processos de negócio incluem manuais, formulários e relatórios, podem prover informação útil acerca da organização e ambiente, tanto como os requisitos para novos sistemas e a justificativa de sua importância**"**

### 1.1.4.
### Coleta de Fatos

Após a identificação de fontes de informação, a seleção das informações uteis (fatos) se torna necessária. Dita atividade, segundo Leite[1] toma estratégias oriundas de ciências sociais, das ciências cognitivas (com forte relacionamento com a Inteligência Artificial), e outras próprias da Engenharia de Software. A seguinte definição de Leite[17] explica o tipo de fatos que podemos extrair do GitHub, isto é, frases, frases de requisitos, e termos relevantes. É importante notar, que dita extração deve ter um mecanismo para manter o rastro da fonte.

> Os fatos podem ser descritos de várias maneira: listas, frases, frases de requisitos ("*o sistema deve...*"), tabelas, grafos conceituais, definições de termos, pequenos parágrafos, desenhos explicativos, enfim qualquer maneira que o Engenheiro de Requisito tenha usado para descrever suas anotações após ou durante o uso de técnicas de elicitação. O conteúdo desse baú, a pesar de não estruturado, deverá ter uma maneira de identificar cada fato "jogado" no baú; por exemplo, um contador que é aumentado de um a cada descrição de fato que é jogada no baú.

Para a coleta de fatos, a leitura de documentos é uma técnica utilizada pelos Engenheiros de Requisitos para elicitar fatos. Nesse sentido, os trabalhos[10,11,12] evoluem tal técnica para uma leitura semi-automatizada, apojados em técnicas de *text-mining*. Em tais trabalhos, são abstraídos (conceitos, palavras frequentes, tópicos) em documentos de elicitação, e em documentos de especificações de requisitos. Assim, uma abstração importante viria a ser o vocabulário utilizado para o domínio de aplicações semelhantes, assim como a organização de tais palavras, esse tipo de abstração é proposto por Leite & Franco[18] no modelo de Léxico Ampliado da Linguagem (LAL).

### 1.1.5.
### Metáfora de GitHub

GitHub é um repositório e um ambiente colaborativo, que promove a interação entre desenvolvedores, através das vantagens próprias de um ambiente de rede

PUC-Rio - Certificação Digital Nº 1322105/CA

social. Atualmente, segundo Metz[19], GitHub possui mais de 35 milhões de projetos *open source*. Tal abundancia de projetos, vem sendo utilizada recentemente para projetos de pesquisa[20,21,22,23], os quais, exploram dados relevantes de projetos, como também informações sobre os usuários desenvolvedores, isto devido à peculiaridade de GitHub de conter informação atualizada e transparente. Outro trabalho[24], mais relacionado à ER, utiliza o GitHub como uma ferramenta para criar e manter rastreabilidade das *issues* dos projetos. A Figura 2, mostra a comunicação entre as entidades de GitHub.

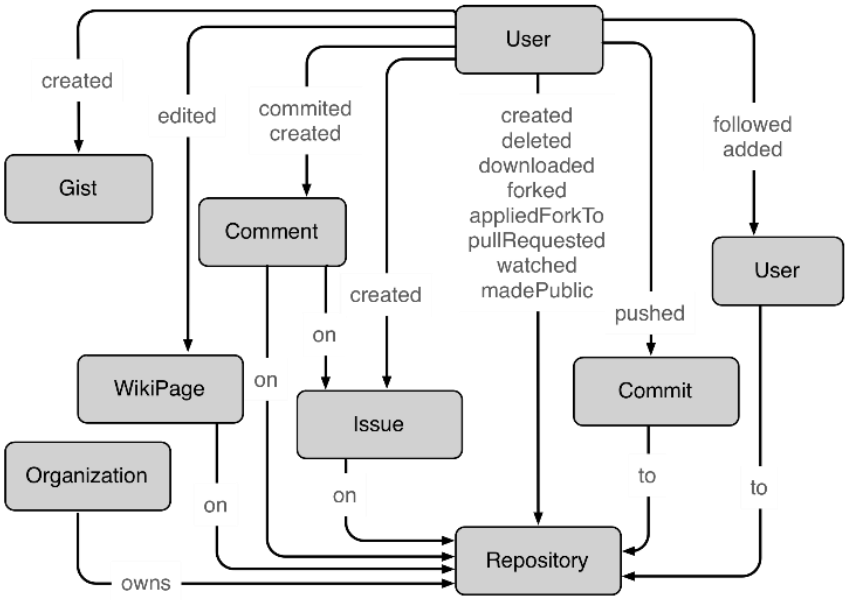



**Figura 2. GitHub *schema*, Rodriguez[25]**

### 1.1.6. Mineração de Textos (TM)

A mineração em textos é definida como o processo de derivar informação de alta qualidade a partir do texto. Hotho et al.[26] define *text-mining* como o descobrimento de conhecimento (KDT) dentre os textos. Isto é possível através do uso de técnicas de *Information Retrieval* (IR), *Information Extraction* (IE), assim como *Natural Language Processing* (NLP), conectando-os com algoritmos de *Data-Mining*, *Machine Learning* e Estatística. É importante notar, que a definição de *text-mining* pode variar de acordo com a perspectiva da área que investiga. A seguir, resumimos algumas definições[26,27] de *text-mining* que são uteis para esta pesquisa.

**Text Mining (Information Extraction),** esta abordagem assume que a mineração de texto corresponde essencialmente a extração de informações em textos não estruturados, ou semiestruturados. Na maioria dos casos, é realizado utilizando técnicas de *natural language processing* (NLP).

**Text Mining (Text Data Mining),** mineração de texto também pode ser definida como a aplicação de algoritmos e métodos de áreas como *machine learning* e *statistics to texts*, com o objetivo de encontrar padrões úteis. Para este efeito, é necessário pré-processar os textos adequadamente. Muitos autores

PUC-Rio - Certificação Digital Nº 1322105/CA

utilizam métodos de *information extraction*, *natural language processing* (NLP) para as etapas de pré-processamento, a fim de extrair dados a partir de textos. Finalmente, técnicas de data-mining podem ser aplicadas aos dados extraídos.

**Text Mining em Engenharia de Requisitos,** no âmbito da ER, temos trabalhos[27,28] que utilizam técnicas de *text-mining*. Massey et al.[27] explica como a técnica de *topic modelling* pode ajudar a filtrar documentos segundo o tópico de interesse.

> *Topic modelling* é uma técnica de text-mining que pode encontrar os tópicos em grandes coleções de documentos. *Topic modelling* permite determinar o interesse particular do indivíduo para depois analisar uma coleção de documentos e determinar o que eles dizem acerca desse interesse. Os tópicos identificados representam tópicos latentes; especificamente, eles são distribuições de probabilidade das palavras no vocabulário definido pela coleção de documentos. Os únicos inputs requeridos são o número de tópicos e os textos dos documentos.

### 1.1.7. Processamento de Linguagem Natural (NLP)

A natureza da documentação de requisitos é de conter descrições em linguagem natural, o qual, traz impactos tais como: ambiguidade nos textos e inconsistências, antes mesmo de que esses fatos se tornarem modelos. Em tal sentido, NLP vem sendo utilizado por abordagens[29,30] de validação de especificações de Requisitos, e análise de documentos de requisitos, entre outros. Para a nossa abordagem, que é tratar artefatos de GitHub para extrair informação (frases) com estereotipo de requisitos, identificamos as seguintes tarefas de NLP que possibilitariam tal objetivo:

***Discourse analysis,*** Mc Kevitt et al.[31] argumentam que frases ou palavras, não podem ser entendidas em termos de valores de verdade senão considerarmos o contexto; isto é, as informações a minerar, devem ser mostradas no contexto que ocorrem, de maneira que o leitor possa ter um maior entendimento de uma possível informação relacionada a requisitos (IRR). Isso é possível com técnicas, como o KWIC[32].

***Part-of-speech tagging* (*POS-tagging*),** chamado também de *gramatical tagging*[33] é o processo de marcar uma palavra em um texto, com a categoria gramatical a qual pertence, tal como: noun, verb, adjective, adverb, proper-noun, preposition, conjunction, interjection, entre outros. Nossa bordagem o utiliza com o intuito de expor as características gramaticais de palavras latentes nos textos.

***Information Retrieval* (IR),** Esta técnica tem a ver com o armazenamento, busca, e a obtenção de informação. Como definido por Manning et al.[34]:

> "Recuperação de informação (IR) é encontrar o material (geralmente documentos) de natureza não estruturada (geralmente texto) que satisfaz uma necessidade de informação dentro de grandes coleções (normalmente armazenado em computadores)"

PUC-Rio - Certificação Digital Nº 1322105/CA

A busca pode ser baseada em metadados ou baseada em índices. O processo inicia quando o usuário introduz uma questão de busca (*query)* que é procurada na fonte de informação (seja um banco de dados ou uma coleção de documentos). No entanto, os resultados retornados podem ou não coincidir com a questão de busca, por isso os resultados são normalmente classificados. Este ranking dos resultados é uma diferença fundamental de busca em IR em comparação com a busca em banco de dados.

***Information Extraction* (IE),** Esta técnica é relacionada, em geral, com a extração de informação semântica de textos não estruturados ou semiestruturados, de maneira que seja possível criar uma vista estruturada das informações presentes nos textos. Piskorski & Yangarber[35] o definem:

> A tarefa de Extração de Informações (IE) é identificar um conjunto predefinido de conceitos em um domínio específico, ignorando outras informações irrelevantes. O domínio consiste de um corpus de textos, juntamente com uma necessidade de informação que seja claramente especificada. Em outras palavras, IE é derivar informação factual estruturada de texto não estruturado.

Entre as tarefas típicas de IE temos: *Name entity recognition*: útil para localizar entidades como pessoas, organizações, lugares, entre outros; *Semi-structured information extraction*: utilizado para extrair informações de textos em linguagem natural, que não necessariamente tem uma estrutura de banco de dados, mas contém informações uteis tais como conceitos ou palavras chaves; *Language and vocabulary analysis:* consiste no processamento de textos para identificar palavras frequentes, similares ou relacionadas; *Audio extraction:* é a procura de caraterísticas relevantes para a extração de trechos importantes, ou para segmentar áudios identificando falantes recorrentes, entre outros.

### 1.1.8. Reuso de Conhecimento

O resultado da nossa proposta é um conjunto de IRR (Informações Relacionadas a Requisitos), extraídas com o intuito de serem reusadas na elicitação de informações para a criação de requisitos. Para tal fim, acreditamos que contribuímos com o tipo de reuso descrito por Goldin & Berry[36]:

> **"**A reutilização é um componente importante de muitos esforços de melhoria em produtividade de software, porque reutilização, quando praticado com cuidado, pode resultar em sistemas de software de melhor qualidade a um custo mais baixo e com um menor tempo de entrega**"**

> **"**Granularidade de reutilização pode variar, desde reutilizar um único artefato, tal como um componente, documento ou caso de teste, até a reutilização de um produto inteiro. Reutilização pode ocorrer durante qualquer fase do desenvolvimento, incluindo durante a análise proposta e análise de marketing**"**

Kupiec et al.[37] definem um processo para o reuso, o qual define as seguintes atividades: (1) capturar ou documentar conhecimento, (2) empacotar

PUC-Rio - Certificação Digital Nº 1322105/CA

conhecimento para reuso, (3) distribuir conhecimento (provendo às pessoas o acesso a esse), e (4) reuso de conhecimento. Tal processo é adotado para a recuperação de informações de GitHub (seção 3.2).

### 1.1.9. R Project

R é uma linguagem de programação, e também um ambiente de desenvolvimento para cálculos estatísticos. O próprio R, e um pacote básico sobre o qual é possível instalar outros pacotes estatísticos para um proposito especifico. No âmbito desta pesquisa, os pacotes que foram utilizados, sem mencionar as suas dependências, são os seguintes:

Pacote de text-mining (tm)
Pacote de análise qualitativa de dados (qdap)
Pacote para processamento de linguagem natural (openNLP)
Pacote para visualização de palavras frequentes (wordcloud)

O principal motivo para utilizar R, é dado pelo tipo de investigação escolhida, que é de tipo exploratória. Nesse tipo de pesquisa, as ideias do que fazer com a informação encontrada precisaram da criação de scripts para a experimentação com os dados. No entanto, foram revisadas ferramentas como o *Weka*, da qual tomamos como modelo os exemplos de *datasets* para tarefas de text-mining. Dado que os artefatos recuperados do GitHub são textos, a representação desses em *datasets* se tornou necessário.

### 1.1.10. Atlas.ti

É uma ferramenta que apoia nas tarefas de análise qualitativa de documentos. Na área de requisitos, vem sendo utilizada para fazer anotações sistemáticas em documentos não estruturados. A ferramenta, apoia ao usuário na análise manual de textos, permitindo anotações e códigos, os quais, podem levar à criação de redes semânticas. O nosso trabalho explorou o Atlas.ti (http:// atlasti.com) no sentido de capturar os usos de tal ferramenta, tanto como as formas visuais para organizar as informações que serão abstraídas.

## 1.2. Desafíos da Pesquisa

Em repositórios *open source* como GitHub, Google Code, e Source Forge, um usuário é livre de carregar um projeto sem precisar de uma configuração predeterminada para os arquivos de um projeto; é necessário apenas caracterizá-lo, com nome, descrição, e linguagem de programação utilizada, entre outros. Nesse sentido, temos nosso **primeiro desafio**, que é a pesquisa exploratória dos artefatos de GitHub, a fim de encontrar características e estruturas comuns, como também mecanismos que permitam automatizar as tarefas de recuperação (IR) e de extração (IE).

PUC-Rio - Certificação Digital Nº 1322105/CA

A escolha de quais artefatos são candidatos a serem minerados para extrair IRR (Informações Relacionadas a Requisitos), é dado pelo tipo de conteúdo que esses artefatos contem, em tal sentido, os artefatos que possuam textos com estereótipos de requisitos (funcionalidades) tem maior relevância. Por exemplo, no GitHub e Google Code, tem-se nas *issues* as diversas tarefas para o desenvolvimento de um sistema, no caso de Source Forge, esses são chamados de *tickets*. No GitHub, a descrição de um projeto está registrada no documento *readme*, para o caso de Source Forge e Google Code, isso é cadastrado no campo *summary*. Aqui, nossa pesquisa exploratória encontra o **segundo desafio**: a recuperação automatizada de artefatos com a mesma função comunicativa, sejam de tipo *readmes* ou *issues*, dado uma busca relacionada a um domínio (*query*).

Neste trabalho, similaridade é definida pelos projetos semelhantes ao software que se deseja realizar. No cenário manual, um usuário realiza uma busca no GitHub e seleciona os projetos similares que sejam relevantes para atender um determinado problema, no entanto, esse processo é inviável quando se tem milhes de projetos disponíveis. Nesse momento acontece o **terceiro desafio**, que é a semi-automatização de leitura de documentos através de abstrações, os quais permitam a filtragem de projetos com IRR relevantes.

Uma vez filtrados os projetos relevantes e extraídas as IRR candidatas para reuso, temos **o quarto desafio,** que é a avaliação da qualidades de tais IRR através de um caso de estudo real, com uma Startup de software.

## 1.3. Estratégia adotada

A investigação de tipo exploratória foi adotada, dado que, como citado por Ventura[13], esse permite estudar um fenômeno a partir da exploração intensa de um único caso. Para tal fim, se escolheu como caso de estudo, a procura de IRR em projetos que pertencem ao domínio de Real Estate, relacionado ao negócio de venda e compra de imóveis.

O documento está estruturado da seguinte forma: O capitulo 2 relata a pesquisa exploratória sob o GitHub e as abordagens que possibilitarão sua mineração. O capitulo 3, detalha os mecanismos para a recuperação de artefatos de GitHub, o qual é apoiado por um processo criado para tal fim. No capítulo 4, são desenvolvidas duas estratégias para a semi-automatização de leitura de documentos, as quais utilizam técnicas de mineração de textos (TM) e tratamento de linguagem natural (NLP). No capítulo 5, é apresentado uma análise qualitativa das IRR por meio de um estudo de caso realizado em uma Startup de software. Finalmente, concluímos essa dissertação comparando nosso trabalho com literatura relacionada, apresentado as conclusões e os trabalhos futuros. A seguir a (Tabela 1) mapeia os desafios para cada capitulo de esta pesquisa.

PUC-Rio - Certificação Digital Nº 1322105/CA

**Tabela 1. Relação entre capítulos e desafios**

| | | |
|---|---|---|
| Capitulo 2 | Desafio 1 | A pesquisa exploratória dos artefatos de GitHub, a fim de encontrar características e estruturas comuns. |
| Capitulo 3 | Desafio 2 | A recuperação automatizada de artefatos com a mesma função comunicativa, sejam de tipo *readmes* ou *issues*. |
| Capítulo 4 | Desafio 3 | A semi-automatização de leitura de documentos através de abstrações, os quais permitam a filtragem de projetos com IRR relevantes. |
| Capítulo 5 | Desafio 4 | A avaliação da qualidades de tais IRR através de um caso de estudo real, com uma Startup de software. |

## 1.4. Resumo

Descrevemos a motivação e os conceitos básicos que norteiam esse texto. Explicamos os desafios ao abordar a mineração de textos sobre documentos em projetos *open source* e como esses desafios serão tratados em cada capítulo.

PUC-Rio - Certificação Digital Nº 1322105/CA

# 2
# Pesquisa Exploratória

Neste capitulo, apresentamos um estudo de caso utilizando o domínio *Real Estate* para obter projetos relacionados no GitHub, os quais serviram como base para a análise de dados abstraindo: conceitos e regularidades. Tudo isto com o objetivo de clarear o problema de pesquisa e ganhar familiaridade com os mecanismos de recuperação de informação em repositórios *open source*.

Dado que a descoberta de IRR de uma forma semi-automatizada é um tema pouco explorado na área de ER, um caso de estudo foi escolhido de modo que possamos confirmar nossas hipóteses iniciais e encontrar outras. Os desafios encontrados para esta pesquisa abrangem áreas como Inteligência Artificial[38], Engenharia de Software[39], Engenharia de Conhecimento[40] e Sistemas de Informação[41]; o estudo de projetos relacionados ao domínio *Real Estate*, nos levou a conhecer mecanismos existentes em tais áreas, que podem ajudar a atingir os desafios. Em particular, o caso de *Real Estate* foi selecionado pelo interesse em conhecer mais sobre o domínio, e com a finalidade de criar aplicativos inovadores reusando o que já existe em projetos *open source* similares. As hipóteses preliminares foram:

1. Reusar conteúdos existentes em projetos *open source* pode resolver o problema de conhecer um domínio rapidamente, e com isso elicitar requisitos de maneira mais eficiente.

2. O Reuso de conhecimento pode ajudar em projetos com restrições de tempo e baseados no momento ideal para lançamento de um produto (*time-to-market*).

3. A escolha de requisitos relevantes para um MVP com base em produtos semelhantes pode criar um produto diferenciado.

4. A identificação de informações relacionadas a funcionalidades implícitas de um domínio, pode antecipar funcionalidades que levaram a criar um MVP com maiores probabilidades de aceitação de usuário.

## 2.1.
## Exploração do GitHub

Os projetos em GitHub como fontes de informação, podem prover fatos com conhecimento sobre um domínio. Para tal, foi realizado uma leitura manual de vários projetos resultado da consulta “Real Estate”, isto nos fez perceber que a obtenção de IRR, não é uma tarefa simples, dado que, nem todos os artefatos de um projeto tem a intenção absoluta de comunicar funcionalidades do software. Além disso, a estrutura particular de cada projeto dificulta a automatização da recuperação extração de IRR utilizando mineração de textos. Assim, a nossa tarefa inicial foi a familiarização com a metáfora deste repositório para identificar o artefatos mais aparente para uma primeira experiência.

PUC-Rio - Certificação Digital Nº 1322105/CA

### 2.1.1. Metáfora do GitHub

Um trabalho preliminar para acompanhar um projeto de software utilizando o GitHub[42, 43], permitiu conhecer melhor o ambiente distribuído e colaborativo deste repositório. Esse ambiente, pode ser sintetizado da seguinte forma: quando criamos um projeto, precisamos cadastrá-lo criando um nome, uma descrição e um documento *readme*. Para acompanhar as tarefas de desenvolvimento, se teve o cadastro das *issues*, e quando uma *issue* precisava ser clarificada, os membros da equipe podiam fazer um *comment* embaixo dele. Além disso, podíamos relacionar (criar rastros) nas *issues* utilizando o caractere #.

Nessa experiência, percebemos que as *issues* eram a seção mais utilizada para descrever os requisitos do nosso software. Por tal, procuramos manualmente outros projetos para conferir se existia esse padrão. Observou-se, que nem sempre uma *issue* descreve requisitos, as vezes, descrevem erros de software, e outras vezes, tarefas de desenvolvimento descritas em nível de código. Com relação a isto, há um trabalho[24] que destaca o valor das issues para gerenciar requisitos. Assim, apesar de ter encontrado a função comunicativa de cada artefato, não é possível afirmar que um determinado artefato, na maioria das vezes, vai conter IRR.

Após esta exploração, se pensou na estratégia de semi-automatizar o procedimento manual que um usuário teria que fazer no GitHub. Esse procedimento consiste em: (1) procurar uma questão de busca, (2) abrir algum dos projetos do resultado da pesquisa, (3) ler a descrição e *readme* do projeto, (4) se for um projeto relevante verificar as *issues* e seus *comments*. Seguindo esse raciocínio, se definiu o escopo desta pesquisa e uma estratégia para explorar IRR em GitHub (Figura 3). A recuperação artefatos e extração de IRR serão realizados sobre o artefato *readme* de GitHub, pois sua função comunicativa é resumir o que cada projeto implementa, ou seja um *readme* pode conter descrições de funcionalidades. Vislumbramos aqui um trabalho futuro, que é o processamento das *issues*, os quais podem ser filtrados utilizando as etiquetas que contem, isso pode levar à terceira tarefa, que é o processamento dos *comments* (evolução de cada *issue*). Dita estratégia visa explorar GitHub seguindo o processo manual de busca. Para o caso de outros repositórios, como por exemplo, Source Forge, existe uma estrutura similar: *summary*, *project*, *ticket*, *discussion*.

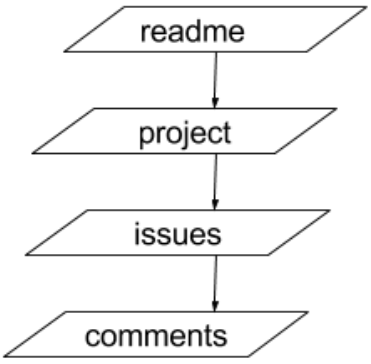



**Figura 3. Ordem de exploração de Artefatos em GitHub**

PUC-Rio - Certificação Digital Nº 1322105/CA

### 2.1.2.
### Artefato Readme

Kupiec et al.[37] afirmam:

> "*Abstracts* são às vezes utilizados como substitutos de um documento completo, por exemplo, como a entrada para sistemas de busca de texto, mas também para acelerar o acesso, fornecendo um ponto intermediário de fácil digestão entre o título de um documento e seu texto completo, que é útil para a avaliação rápida da relevância".

A exploração nos projetos de GitHub evidenciou o *readme* como primeiro artefato a ser processado. Tomando a afirmação de Kupiec et al.[37], julgamos que, os *readmes* tem o papel de *abstracts* sobre os projetos de GitHub. Nos exemplos a seguir, apresentamos alguns KWIC obtidos dos *readmes* por meio de técnicas simples de mineração de texto. O objetivo é encontrar informações que podem fornecer conhecimentos sobre projetos existentes relacionados com o conceito de *Real Estate*. Foram usados diversos termos na composição das consultas de busca, com isto queremos mostrar, como uma variação mínima na configuração da mineração de texto, pode levar a resultados diferentes.

Mineração usando a consulta "Real Estate"

#Miami-**Real-Estate**-Agent-Your-Guiding-Star-to-Buy-Best-Apartment-of-your-Choice If you are looking for the beneficial and stable real-estate agent who can help you in buying <a ref="http://www.luxrealestatemiami.com/">Luxury Apartments in Miami</a> then make sure to select the one that is reliable and trustworthy. You would need the help of **real estate** agent while buying, marketing or rental a **real estate**, even if you are searching for the residential or commercial...you would be able to find numerous websites that would take you to the names of best realtors that are available in Miami.

***Rastro***: Usuário: RichardJames12. Projeto: Miami-**Real-Estate**-Agent-Your-Guiding-Star-to-Buy-Best-Apartment-of-your-Choice

Mineração usando a consulta: "Real Estate" e "Feature"

* List **Feature**d properties
* Display full details of a property
* Search Form with autocomplete for location
Currently there are reserve page name these are:
* Property - display the single or full details of property.
* Search Properties - Display the result of the searched properties

***Rastro***: Usuário: wp-plugins. Projeto: wp-**real-estate**-property-listing-crm.

Mineração usando a consulta "Real Estate" e "Feature" e "issue"

* [PRO] Added RSS **feature** for listings.
* Fixed an **issue** on open house **feature**.
* Fixed some **issues** on rooms **feature**.
* Added mailto **feature** for agent info activity.

**Rastro**: Usuário: wp-plugins. Projeto: **real-estate**-listing-realtyna-wpl

PUC-Rio - Certificação Digital Nº 1322105/CA

No entanto, verificou-se o problema da ambiguidade em descrições em linguagem natural; isto é, o conceito *Real Estate* é também utilizado para descrever um atributo de desenho de telas em dispositivos eletrônicos, resultando na geração de falsos positivos, como é mostrado abaixo:

> Because it's displayed vertically and I want to save screen **real estate**, I want to reduce the maximum number of candidates ido displays:
>
> **Rastro**: Usuário: bradleywright. Projeto: emacs.d.

> An option map of additional drawing parameters. I lowered it to 63dpi so it wouldn't take up so much screen **real estate**.
>
> **Rastro**: Usuario: Engelberg. Projeto: instaparse.

Baseados no trabalho de Brett[44], se realizou a primeira tarefa em direção à extração de informação: a recuperação automática de readmes de projetos de GitHub. Dita tarefa utiliza os conceitos[45] da área de *Information Retrieval* (IR), como é: **A entrada,** um *keyword search query*, que expresse a informação da necessidade de um usuário, **As saídas**, *links or pointers to documents*, isto é, uma coleção de *readmes* contendo informação (rastro ou *pointers*) para as fontes do qual foram extraídos.

Brett[44] recomenda que a busca de tópicos sob uma coleção grande de documentos (*corpus*), deve ser no mínimo com 1000 documentos. Em tal sentido, o GitHub pode fornecer tal quantidade, pois, ao fazer uma busca, é usual obter resultados, que está na ordem de mais de 1000 projetos. Assim, se tornou necessária a criação de uma ferramenta para tal recuperação automatizada. Dita ferramenta, poderia ser reusada quando a criação de um *corpus* de *issues*, ou um *corpus* de seus *comments* seja oportuno.

### 2.1.3. Cenários de uso do Readme

Dado que a linguagem natural tende a ser ambígua, as nossas primeiras experiências em mineração de textos em *readmes*, nos mostraram que a construção de uma ferramenta para apoiar a Elicitação de Requisitos não é possível sem a análise manual ou discriminação do Elicitador. Como indicado por Goldin & Berry[46] ao referir-se à ferramenta *Abstfinder* para encontrar abstrações de conceitos em documentos de requisitos, "A*bstfinder* ajuda a encontrar pistas para o Elicitador, que o ajudam na identificação de abstrações, ao previsto que, o Elicitador esteja preparado para fazer o raciocínio de seleção". A seguir, descrevemos alguns cenários nos quais a mineração de textos em *readmes* pode ajudar na sua semi-automatização de leitura.

**Descartando a leitura de projetos sem a frase "*Real Estate*" incorporado**: O critério de busca predefinido do GitHub, não toma em conta se a consulta é formada por palavras compostas, por exemplo, a consulta "real estate"

PUC-Rio - Certificação Digital Nº 1322105/CA

é traduzida como “real OR estate”. Além disso, se existir resultados com as duas palavras no *readme*, não é garantido na ordem de "real estate".

**Descartando a leitura de readmes sem conteúdo:** GitHub requer um artefato *readme* para cada projeto criado. É comum que os usuários criem *readmes* fictícios, apenas para cumprir esta regra. Portanto, as buscas por esses artefatos podem retornar resultados inúteis.

**Priorização de projetos usando os metadados de GitHub:** Como referenciado nos trabalhos[47,48,49], GitHub tem aspectos sociais que podem ser usados para classificar os projetos. Aspectos como o número de *stars*, *forks*, *issues* ou *issues* com *comments* são metadados que irão melhorar a relevância de um projeto.

**Encontrando Projetos com *live-samples*:** O reuso de software é melhorado quando um projeto insere um exemplo visual, tal como, uma imagem ou um elo que apresente o que o projeto faz. Enquanto isso, com a mineração de *readmes* percebemos que os elos podem levar a visualizar versões em produção de um projeto, isto melhora a compreensão rápida de um projeto, fornecendo o melhor plano ao decidir o que reutilizar.

**Provocando criatividade:** Através da leitura de projetos similares, e pela comparação de projetos, é possível apoiar a criatividade de projetos com ideias iniciais, e na montagem de diferentes partes de projetos, o qual possibilitaria a inovação.

### 2.1.4. Restrições de GitHub

A seguir, listamos algumas das limitações encontradas ao explorar o API de GitHub para recuperar documentos:

1. O limite de requisições é de 5000/hora:
   Essa restrição faria com que o acesso pelo API se limitasse a cinco questões de busca. Cada questão retorna 1000 projetos.
2. A quantidade de resultados máxima é de 1000/questão:
   Usualmente, GitHub mostra mais de 1000 projetos coincidindo com a questão de busca, mas apenas 1000 projetos são disponibilizados. Tal fato, nos fez pensar em uma situação onde, talvez, o projeto de número 1001 seja de interesse para o usuário por possuir funcionalidades relacionadas ao domínio.
3. Não é possível usar caracteres especiais como parte da questão de busca:
   Caracteres do tipo <. , : ; / \ ` ' " = * ! ? # $ & + ^ | ~ < > ( ) { } [ ]> impedem que realizemos uma busca por frase, entre aspas, tal como o "real estate", resultando em documentos com falsos positivos.

## 2.2. Abstração de Informações

Após a exploração de *readmes*, se identificou que tais documentos contêm informações uteis para reuso, as quais estão misturadas com informações não

PUC-Rio - Certificação Digital Nº 1322105/CA

relevantes. A extração das informações uteis (IRR) precisa de mecanismos de filtragem que permitam automatiza-lo. Para tal fim, e baseados nos trabalhos relacionados[10,11,12], se adotaram duas abordagens (Cap. 4), dado que essas foram utilizadas na extração de (conceitos, palavras frequentes, e tópicos) existentes em documentos de elicitação, e em documentos de especificações de requisitos. Para nosso objetivo, é preciso misturar ambas abordagens, dado que as abstrações que procurarmos, as IRR (frases), podem estar compostos de conceitos, palavras frequentes e tópicos relevantes.

### 2.2.1. Informação Relacionada a Requisitos

Definimos como informações relacionadas a requisitos (IRR), às frases que podem ser mineradas para nos dar uma ideia dos objetivos de um projeto, de maneira que o elicitador ganhe rapidamente o máximo conhecimento contextual. As IRR podem servir para reuso imediato, como também para a geração de novas ideias. A seguir, mostramos dois exemplares de IRR extraídos do *readme* (Figura 4):

*"Find all properties near you with iPhone's GPS"*
*"One-click "Contact Agent" or "Ring Agent" for all properties"*

A filtragem manual deste tipo de informações é simples de forma manual, mas ineficiente ao tratar com centenas de documentos. No entanto, a automatização de tal tarefa, pode resultar ineficiente, dado que não é possível assegurar que todos os IRR serão relacionados ao domínio objetivo; esse é o motivo pelo qual, esse trabalho propõe uma abordagem **semi-automatizada**. Assim, é possível reduzir o esforço de leitura ao automatizar a recuperação de artefatos, e traves de eles, a filtragem de projetos relevantes, dos quais é possível abstrair os IRR. Sem embargo, a discriminação de que IRR é útil ou não, só pode ser determinado por o elicitador. É importante notar que a filtragem de projetos relevantes, precisa de mecanismos como é a identificação de padrões que o automatizem.

### 2.2.2. Abordagens para Filtrar Informações

Uma das estratégias encontradas para filtrar de informações está no trabalho[10] que propõe a filtragem através do uso da técnica *POS-tagging*, o qual é usado para anotar (criar um *tag*) com a sintaxe (forma gramatical) de cada palavra em textos com linguagem natural. Tal anotação é útil para identificar automaticamente o tipo de palavras que possuem uma semântica desde a visão da ER.

PUC-Rio - Certificação Digital Nº 1322105/CA

```
# ozEstate

http://skitch.com/anthonymittaz/be5c2/photoshop

## License

Copyright 2009 Anthony Mittaz. All rights reserved.

This sourcecode is released under the Apache License, Version 2.0
http://www.apache.org/licenses/LICENSE-2.0.html

## Missing libraries

The installed iPhoneSimulator is missing the `tidy` library. There
is a rake task to install it:

        rake install_dependencies


## For Sale

ozEstateFS MAKES IT EASY AND FUN TO FIND, MANAGE, AND VIEW
AUSTRALIAN PROPERTY FOR SALE IN ONE INTUITIVE INTERFACE.

Key Features
Find:
- Works with all properties on www.realestate.com.au.
- Find all properties near you with iPhone's GPS.
Manage:
- Use a Favourite's list to track properties.
- One-click "Contact Agent" or "Ring Agent" for all properties.
Interface:
- All this in a beautiful iPhone UI that makes you want to use it.

Note
Requires an Internet Connection.

## For Rent

ozEstate MAKES IT EASY AND FUN TO FIND, MANAGE, AND VIEW
AUSTRALIAN PROPERTY RENTALS IN ONE INTUITIVE INTERFACE.

Key Features
Find:
- Works with all properties on www.realestate.com.au.
- Find all properties near you with iPhone's GPS.
Manage:
- Use a Favourite's list to track properties.
- One-click "Contact Agent" or "Ring Agent" for all properties.
Interface:
```

**Figura 4. Dados brutos de Readme no GitHub**

A (Tabela 2) mostram o *POS-tag* de palavras com semântica para a E.R. Por exemplo, é possível através da identificação automática de um verbo modal (allow) + noun (user), a ancoragem da seguinte IRR:

> *"Main Screen **allows user** to type in an address in the United States to query for Basic Info Screen shows the basic information of the real estate in a tabular format, Price History Screen shows charts of the real estate prices in the past 1, 5, and 10 years."*

**Tabela 2. POS-Tags com semântica para a filtragem de IRR**

| POS-Tag | Semântica em Engenharia de Requisitos | Termos |
|---|---|---|
| Verbos modais | Expressando possíveis/obrigatórios requisitos | must<br>wants<br>allows<br>should |
| Substantivos | Expressando possíveis stakeholders: lugares, pessoas, organizações. | user<br>client<br>developer |
| Substantivos qualitativos | Expressando possíveis requisitos não funcionais | security<br>precision<br>usability |

PUC-Rio - Certificação Digital Nº 1322105/CA

Uma segunda abordagem para filtrar informações, é a utilização de *templates* ou regularidades existentes em documentos de especificação de requisitos, os quais são chamados de *boilerplates*[50]. Por exemplo é comum este tipo de descrições de requisitos:

*"The system should provide"*
*"The system should be able to*"

Ambas abordagens serão utilizadas para a filtragem de IRR (capitulo 4) de maneira automática.

## 2.3. Resumo

Descreveu-se a pesquisa exploratória utilizando o caso "Real Estate". Se descreveu a função de artefatos de GitHub (*readme, issue e comments*). Se identificou uma estratégia para explorá-los e que poderia ser aplicado em outras fontes de informação (repositórios *open source).* Descrevemos brevemente as estratégias para a mineração de textos que podem ser atingidas por duas abordagens: a primeira apojada na sintaxe das palavras, e a segunda baseada na identificação de padrões. Uma contribuição importante é a publicação[51] de dita pesquisa exploratória.

PUC-Rio - Certificação Digital Nº 1322105/CA

# 3
# Coletando Informações

Neste capitulo apresentamos um processo para a mineração de documentos de repositórios *open-source*. Tal processo (3.1) é baseado em abordagens similares[46,52] e na literatura sobre técnicas de *text-mining*[53]. Em sequência, a atividade Collate (3.1.1) é automatizada para a função que permite recuperar os documentos *readme* dado uma questão de busca.

## 3.1.
## Processo para Minerar Informações

Vários autores[54,46,10,55] da área de ER, têm feito esforços para descobrir ou minerar textos a partir de um *corpus of texts* para os seguintes propósitos: desambiguação de especificações, identificação de conceitos relevantes, compreensão do domínio do problema, classificação de requisitos não funcionais, entre outros.

Para sistematizar nossa estratégia, criamos um processo que apoie as tarefas de mineração de textos baseado em dois trabalhos. O primeiro[53], é relacionado a um processo para utilizar técnicas de mineração no ambiente estatístico computacional R. O segundo[52], define três fases para abordar a mineração em textos etnográficos. Assim, se definiram três níveis de atividades utilizando a técnica de modelagem SADT[56] mostrados na Figura 5.

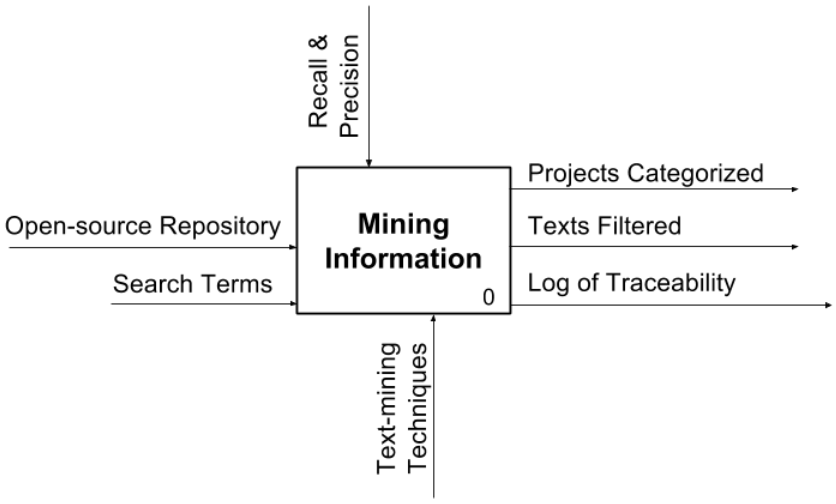



**Figura 5. Modelo SADT do processo, nível 0**

De maneira geral, o **nível zero** de atividades define as entradas e as saídas esperadas do processo para a mineração de textos. A primeira entrada, é composta pelo repositório de projetos open-source GitHub. A segunda entrada, são os termos de busca do usuário, tal como ele o realizaria no buscador do repositório. O processo é controlado pelas medidas *recall* e *precision* utilizadas no desenho de sistemas de *Information Retrieval*[57]. *Recall*, se refere à fração de documentos que são relevantes para uma consulta e que são recuperados com sucesso; e *Precision*, é a fração de documentos recuperados que são relevantes para uma consulta:

PUC-Rio - Certificação Digital Nº 1322105/CA

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

Com tais medidas, o *recall* do processo é medido pela quantidade de projetos relevantes que são recuperados do GitHub, e *precision* do processo, é medido pela qualidade das IRR.

A primeira saída do processo, são os projetos categorizados por similaridade, pudendo ser pelo domínio ao qual pertence um projeto, ou por alguma abstração relevante que caracterize os textos. É importante notar que, mesmo que a consulta feita ao GitHub é relacionada a um domínio, essa não garante que todos os projetos resultado da busca sejam relacionadas à consulta; isto foi evidenciado no caso exploratório de "*real estate*" de ser utilizado em dois domínios. A segunda saída são as IRR (*filtered texts)*. A terceira saída é o *log* de rastreabilidade dos textos, desde sua extração do repositório até a filtragem das seus IRR; isto com a finalidade de que o usuário possa identificar o repositório de interesse para procurar mais informações manualmente. O mecanismo para transformar as entradas em saídas, é dado pelos pacotes para mineração de texto existentes para a ferramenta estatística computacional R[53]. A continuação, o processo é detalhado em três macros atividades (Figura 6), definidas a seguir:

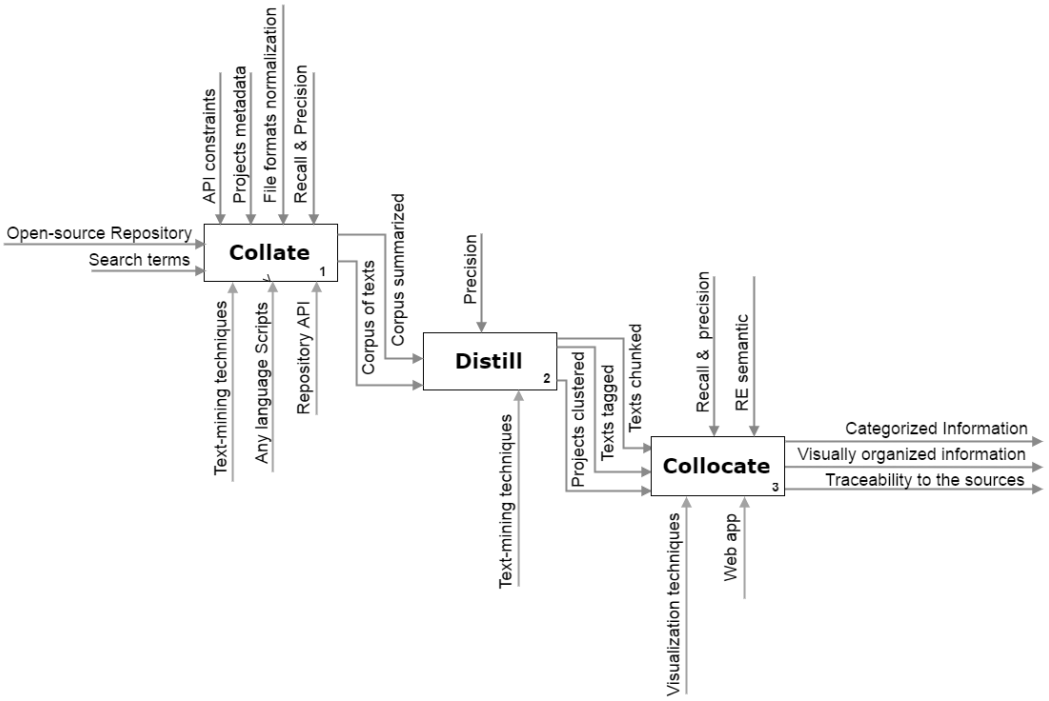



**Figura 6. Modelo SADT de macro atividades do processo, nível 1.**

PUC-Rio - Certificação Digital Nº 1322105/CA

### 3.1.1.
### Collate

O propósito desta atividade é produzir automaticamente uma coleção de textos *corpus of texts*, o qual ao conter textos semiestruturados, precisa ser processado para transforma-lo em um corpus estruturado, de maneira que seja possível produzir um *corpus summarized*, isto permitirá realizar as tarefas de mineração de informações. A atividade é dividida em três subatividades (Figura 7): *Retrieve*, que tem como objetivo recuperar os textos de GitHub, para o escopo de esta pesquisa, apenas *readmes; Transform*, que se encarga de buscar maior precisão no *corpus* textos recuperados de GitHub; *Check*, que revisa se o processo de transformação de textos não elimina os traços originais do *corpus*.

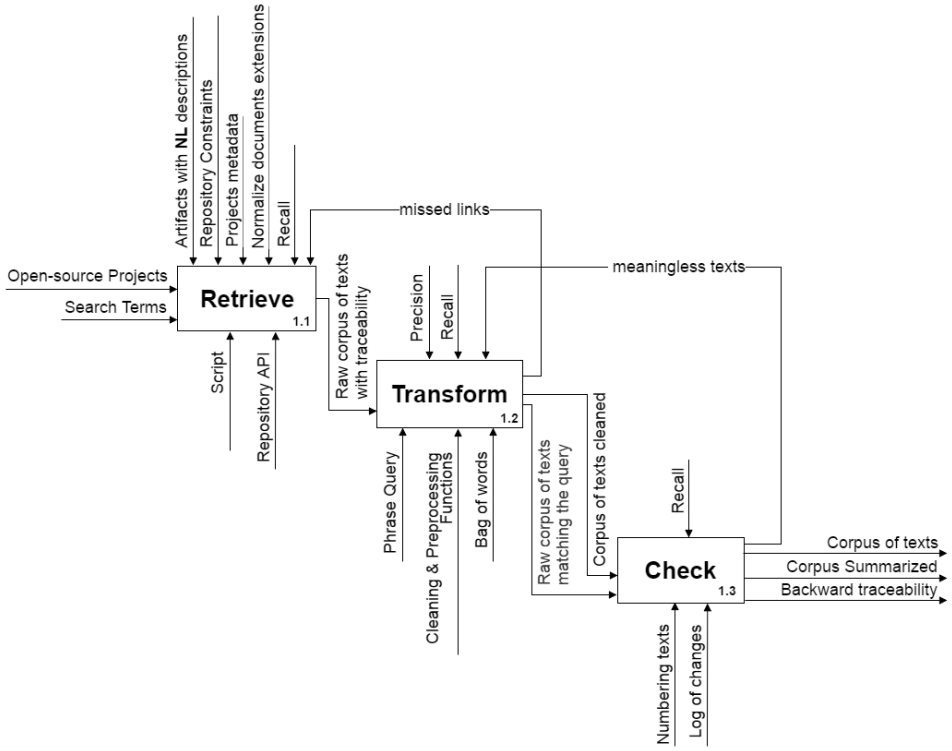



**Figura 7. Modelo SADT de atividades do Collate, nível 2.**

Os controles para a recuperação de projetos são:

a. As restrições do API do repositório, detalhados na seção (2.1.3).

b. Os metadados utilizados em GitHub para dar relevância a um projeto, isto é dado pelas valorações *star* ou *fork*. Uma *star* indica o favoritismo por um projeto, e um *fork* indica que o projeto está sendo utilizado por outro usuário sem afetar o projeto original; aqui, cabe ao proprietário do projeto decidir a incorporação das mudanças propostas.

c. O controle *Recall* é usado para obter mais *readmes* de projetos relevantes, do que o permitido pelo GitHub de maneira convencional (1000 textos).

PUC-Rio - Certificação Digital Nº 1322105/CA

d. O controle *Precision* é usado para retirar readmes não relevantes, ou seja, *readmes* vazios, *readmes* apenas com título, *readmes* que não possuam os termos da questão de busca.

e. O controle *normalize documents extensions* renomeia todos os textos para uma mesma extensão (.txt).

Os mecanismos para a requisição de projetos são:

a. A API v3.0 de GitHub que permitirá realizar as requisições de textos de projetos.

b. Qualquer linguagem de programação que permita a execução das requisições.

c. As técnicas de mineração de textos que ajudem a realizar as tarefas de transformação de um corpus não ou semiestruturado para um estruturado.

A (Figura 7) mostra as subatividades do *Collate*, e que são detalhadas a seguir.

PUC-Rio - Certificação Digital Nº 1322105/CA

#### 3.3.1.1 Retrieve

A necessidade de criar a atividade *Retrieve,* foi inspirada pelos vários exemplos de *datasets* existentes em R para experimentar as diferentes técnicas de *text-mining*. Assim, essa atividade é iniciada por uma questão de busca do usuário, que é processado pelo *script*[58] criado para recuperar *readmes* de projetos. Tal *script* utiliza a técnicas *querying*[45] de *Information Retrieval* (IR).

Os controles para a recuperação de textos são:

a. Textos contendo linguagem natural, pois processo atenderá o processamento de diferentes artefatos de GitHub, como especificado em (seção 2.1.1).

b. As restrições do GitHub (seção 2.1.4), que foram tratadas de maneira que seja possível obter mais resultados do que o limite atual: 1000 projetos como máximo.

c. Os metadados de GitHub: *star* e *fork*.

d. A uniformidade dos textos, pois, nem todo *readme* é encontrado na sua extensão predeterminada (*.md*), podendo ter extensões como (*.rdf, .doc, .html*), entre outros.

e. O controle *recall*, que garante a ordenação dos readmes, preservando a ordenação predeterminada dos projetos como no GitHub.

A saída desta atividade é um corpus "cru" *raw corpus of texts with traceability*, contendo *readmes* tal e como aparecem no GitHub. Os *readmes* são nomeados com a seguinte formatação: número de aparição nos resultados do

repositório + nome de usuário + nome do projeto. Isto permitirá manter o rastro de um texto até sua fonte.

#### 3.3.1.2 Transform

Essa atividade foi desenhada para transformar o corpus de textos criado na atividade *Retrieve*. O objetivo é produzir um *corpus* estruturado que possa ser trabalhado nas tarefas de mineração de textos.

Os mecanismos para a transformação de textos são:

a. *Phrase query*: é o passo que permitirá retirar textos não relevantes de um *corpus* de textos. O uso de frases em consultas, como mencionado por Kumaran & Allan[59], é conhecido de melhorar a precisão em tarefas de *Information retrieval.* Esse mecanismo é necessário dada a restrição de GitHub de não permitir a busca entre aspas, resultando em textos com falsos positivos. A (Figura 8) mostra as funções utilizadas para processar um *corpus* de 1141 textos em R, dando como resultado 162 *readmes* sem o texto "real estate" ou "Real Estate".

```
library(tm)
library(qdap)
corpus<- Corpus(DirSource("/home/nita/r-docs/readmes-realestate"),
readerControl=list(language="lat"))
phrases <- c("real estate", "Real Estate")
word.freq <- apply_as_df(c, termco_d, match.string = phrases)
mcsv_w(word.freq, dir = NULL, open = T, sep = ", ", dataframes = NULL,
pos = 1, envir = as.environment(pos))
```

**Figura 8. Script para reduzir o Corpus de Readmes**

b. O segundo mecanismo *cleaning & preprocessing*, foi adotado do trabalho[53], pois, um processo de limpeza é necessário para que um corpus seja transformado em um *dataset*, eliminando elementos como: pontuações, espaços extras em branco, e *stop-words* (*the*, *and*, *or*, entre outros).

c. O terceiro mecanismo, após a limpeza, é a preparação do *corpus* para ser sumarizado. Isto é obtido com a técnica de *Bag of Words*, que é um modelo utilizado em IR; a representação de tal modelo é uma matriz de documentos por palavras, sem considerar a gramática ou a ordem dela. Um *corpus* de texto sumarizado irá a permitir, por exemplo, a frequência com que cada palavra aparece em cada texto, considerando também, sua frequência em relação a todo o *corpus*. Isto, é logrado com o esquema (*tf-idf*) (Berry. M, 2004).

Os controles desta atividade são:

a. *Recall* e *precision*, detalhado ao definir a macro atividade *Collate*.

PUC-Rio - Certificação Digital Nº 1322105/CA

b. *Missed Links*, que é um controle útil quando o objetivo seja capturar elos existentes em textos. Elos para as versões em produção de projetos, e demonstrações, entre outros, permitem ao usuário agilizar a leitura das funcionalidades dos projetos.

As saídas de esta atividade são:

a. Um *corpus* de textos ainda não estruturado, no entanto, mais preciso.

b. Um *corpus* de textos semiestruturados (*summarized*), contendo uma matriz de documentos por termos, com o qual é possível criar um *dataset*.

#### 3.3.1.3 Check

Este processo foi desenhado para garantir a qualidade do *corpus* de textos, tal como sua versão sumarizada. Ambos, deverão conter um arquivo no qual sejam registrados os traços que permitam identificar a fonte dos readmes após a transformação dos textos.

Os controles de esta atividade são:

a. *Meaningless*, verifica que na atividade anterior *transform*, um *readme* não seja limpado até ao ponto de perder seu real conteúdo. Por exemplo, se pensou que em outro tipo de abordagem, a abstração de elos das versões em produção dos projetos, poderiam melhorar a experiência do usuário; para tal, a limpeza de elos não seria conveniente e se deveria ter cuidado de manter o contexto nos quais são mostrados.

b. O *Recall*, garante que os textos de um corpus mantenham a relevância (ordem) que esses textos tinham no GitHub.

Os mecanismos desta atividade são:

a. *Numbering texts*, é usado para manter a ordem de relevância dos projetos após o processamento feito na atividade anterior.

b. O mecanismo *log of changes*, que salva as mudanças feitas na nomeação dos arquivos.

A saídas da atividade *Check* são:

a. Um *corpus* com maior precisão, que é dado pelo *phrase query*.

b. Um *corpus summarized*, que é um corpus estruturado, pronto para a criação de um *dataset*.

c. O arquivo de *backward traceability* que contém os rastros de cada texto.

PUC-Rio - Certificação Digital Nº 1322105/CA

### 3.1.2.
### Distill

Após a preparação dos *readmes*, realizado no macro atividade *Collate* (Figura 6), esta atividade é encarregada de "destilar" os textos *readme* relevantes, e com isto possibilitar a extração das IRR. Três atividades (Figura 9) são direcionadas para tal objetivo:

A atividade *Clustering*, foi desenhada para atingir o problema da ambiguidade da linguagem natural. Nesse sentido, os projetos podem ser classificados por similaridade, utilizando técnicas como o *cosine measure*[61] ou o *Latent semantic analysis* (LSA) utilizado em abordagens[52,62] relacionadas a esta pesquisa. No entanto, essa atividade não foi desenvolvida neste trabalho, pelo tempo limitado do plano para a exploração de tais técnicas.

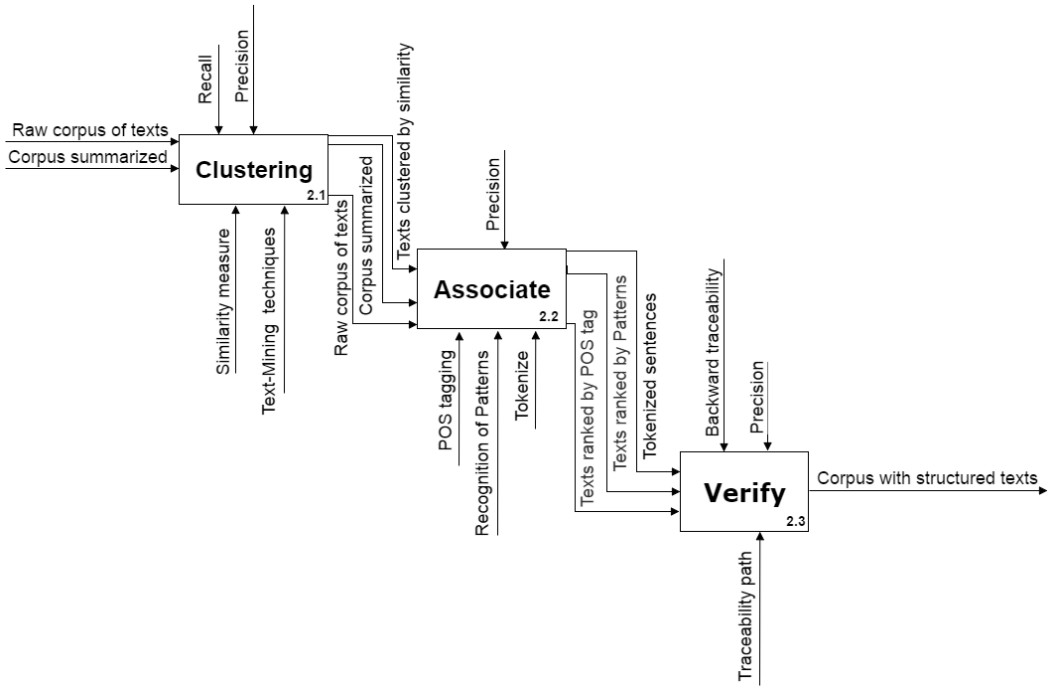



**Figura 9. Modelo SADT de atividades do Distill, nível 2.**

A atividade *Associate* analisa cada texto do *corpus summarized* procurando a semântica dos termos e relacionamentos entre eles.

Para tal fim, definimos três mecanismos:

a. Este mecanismo é alinhado à primeira abordagem[10] descrita em (seção 2.2), utilizando a técnica de *POS-tagging*), na qual, a traves das frequências de palavras de tipo *nouns*, proper-*nouns* e *verbs* é possível abstrair entidades genéricas, entidades especificas e ações em textos do *corpus*.

b. A segunda abordagem[50] é direcionada para o mecanismo *Recognition of patterns*, com o qual é possível identificar a frequência de padrões nos textos, como por exemplo: *"the system should be able*".

PUC-Rio - Certificação Digital Nº 1322105/CA

c. O terceiro mecanismo, *tokenize*, permite a extração automática das IRR (frases) ancoradas pelas duas abordagens, no entanto, essa técnica não foi usada, mas sim a extração manual das IRR para sua avaliação no caso de estudo com a Startup.

As saídas esperadas para esta atividade são:

a. Uma nova organização do *corpus*, baseado na filtragem de textos, segundo a frequência de palavras abstraídas usando o *POS-Tag*ging.

b. Uma nova organização do *corpus*, baseado na filtragem de textos contendo padrões pre identificados. Cabe destacar que isto é realizado sobre o *Raw* C*orpus of texts*, pois o um *Corpus summarized* não possui mais, as palavras de conexão em uma frase (and, in, or, the, is, entre outros).

c. O conjunto de frases extraídas, as quais contém as IRR (seção 2.2.1).

A atividade *Verify*, revisa a traçabilidade, que como em atividades anteriores, após o tratamento de um *corpus*, é necessário verificar que as informações possuam rastros até suas fontes. O controle *precision* é utilizado para gerar um *corpus* mais estruturado, os qual tem a seguinte estrutura:

Nome {número relevância, nome do usuário, nome do projeto}
Relevant Nouns { , , }
Relevant Proper-nouns { , , }
Relevant Verbs { , , }
Phrases {“ ”, “ ”, “ ”, “ ”}
Trace-Path { , , , , }

### 3.1.3. Collocate

No decorrer desta pesquisa, essa atividade se vislumbrou como a mais importante, pois, a mineração de abstrações uteis para a extração de IRR, pode resultar em muitas palavras relevantes, como são os termos frequentes. A visualização de tais dados teria que representar a frequência e relacionamento de tais palavras de maneira que facilite a leitura das informações. Essa facilidade é definida pelo tempo que o leitor terá para discriminar o que é de interesse.

As saídas e mecanismos da atividade são:

a. As IRR apresentadas numa aplicação que utilize técnicas de visualização de dados, de maneira que as informações se mostrem categorizadas por algum aspecto relevante, por exemplo, entidades. A (Figura 10) é uma rede semântica criada utilizando informações extraídas na pesquisa exploratória, para o caso da *Real Estate*. As entidades: API, Zillow, e Rets foram identificadas pelo *POS-tag* como *proper-nouns;* e entidades mais genéricas como: application, web, ou mobile, foram identificadas pelo *POS-tag noun*.

b. As *funcionalidades* colocadas na (Figura 10) podem ser abstraídas pelo uso de padrões ou pelo uso de verbos. A (Figura 11) mostra como a

PUC-Rio - Certificação Digital Nº 1322105/CA

sintaxe das palavras, pode ajudar a criar extrair IRR relacionadas a: requisitos funcionais e requisitos não funcionais.

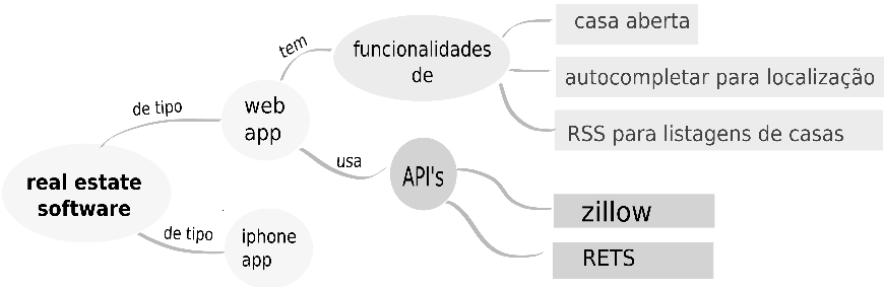



**Figura 10. Rede semântica de informações relacionadas a "*real estate*"**

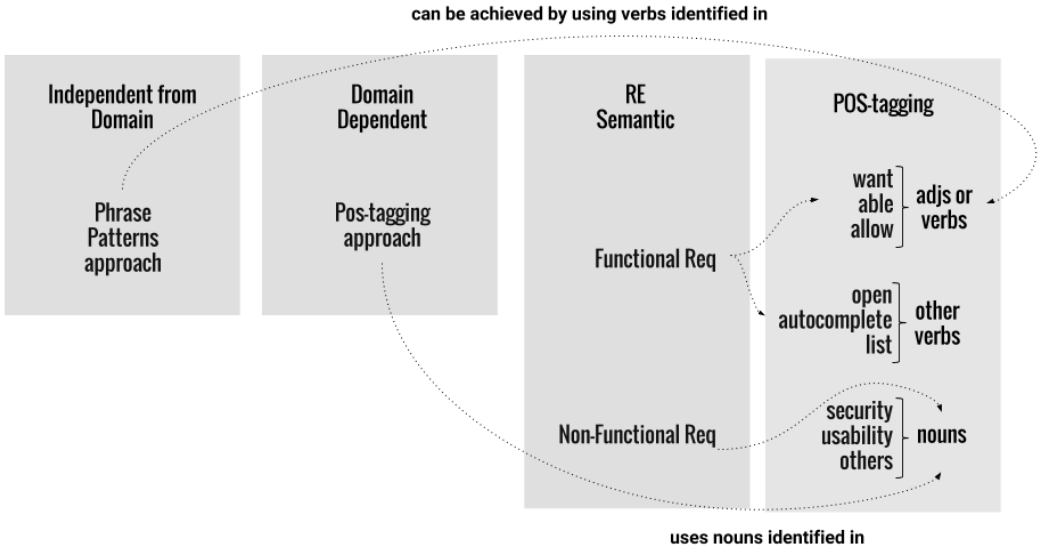



**Figura 11. Combinação de abordagens para ancorar informações**

## 3.2. Recuperando Informações Automaticamente

Nessa seção, utilizamos a atividade *Collate* (Figura 7) para atingir o desafio 2 (seção 1.2). Uma segunda motivação, para a criação uma ferramenta automática, nasceu pela necessidade de sair do caso de estudo exploratório *Real Estate*, de maneira que seja possível recuperar rapidamente um *corpus* de *readmes* para qualquer consulta de busca em GitHub.

### 3.2.1. Desenho

Sinclair[63] afirma este princípio para a construção de um Corpus:

> "O conteúdo de um corpus deve ser selecionado sem levar em conta a linguagem que eles contêm, mas de acordo com sua função comunicativa na comunidade em que elas surgem".

PUC-Rio - Certificação Digital Nº 1322105/CA

Nesse sentido, a recuperação de artefatos com a mesma função comunicativa, no caso, os readmes, cumpre com tal princípio. A visão de essa ferramenta é enquadrada pelas restrições de GitHub (seção 2.1.4). A grosso modo, os critérios de desenho foram: a publicação web da ferramenta, que permita a qualquer pessoa a recuperação de *readmes* de GitHub, sem ter que lidar com detalhes técnicos de sua API ou de desenvolvimento. É importante destacar que existe um trabalho mais amplo, o GitHub Torrent[23] que coleta *dumps* (uma imagem dos dados, em certa data) de repositórios completos dado um horizonte de tempo. No entanto, tem barreiras técnicas como é uma credencial (*personal access token),* que cada usuário deve solicitar no GitHub.

### 3.2.2. Arquitetura

Dado que o tempo de desenvolvimento foi uma das principais preocupações para a realização da ferramenta, utilizamos a linguagem Ruby, especificamente, com bibliotecas que permitiram a construção rápida de um *corpus* apesar das restrições do API de GitHub. As bibliotecas reutilizadas, ajudaram a automatizar a extração de *readmes* e empacotamento de um *corpus* em um navegador web, levando em conta, também, o desempenho do mesmo para extrair tal quantidade de textos, usualmente 1000. Cada *gem* (bibliotecas de Ruby) da (Figura 13) tem um propósito específico que é descrito a seguir:

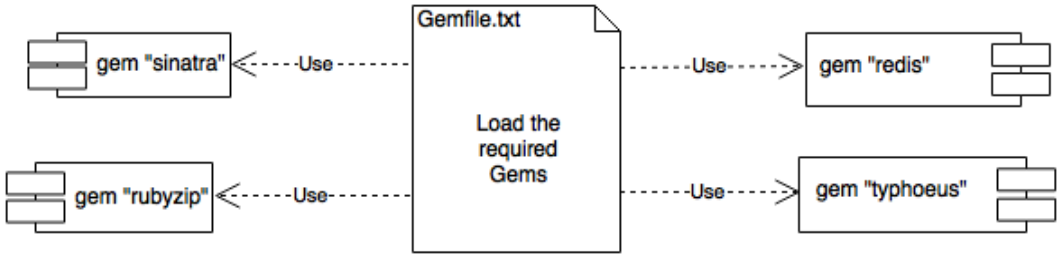



**Figura 12. Bibliotecas "*gems*" de Ruby utilizadas.**

**Sinatra Library**: É um *domain specific language* (DSL) para a criação rápida de aplicações web em Ruby.

**RubyZip Library**: Permite a leitura e escrita dos *readmes* extraídos, colocando-os em uma pasta que é compactada para ser descarregado.

**Typhoeus Library:** Permite um processamento mais rápido das requisições a GitHub. Suporta requisições em paralelo encapsulando a *Curl Library,* que é utilizado para transferir dados através de uma URL

**Redis:** Cria uma estrutura de dados em memória, serve como base de dados para os conjuntos de informação a ser transferidos.

A (Figura 13) descreve os módulos do script criado. O **server.rb** permite a configuração do servidor web que por meio da interface do *ISinatra* carrega o index da aplicação. O **github_consumer.rb** requisita os *Readmes* usando a interface IGitHubCosumer e cria os traços dos *readmes,* por meio da interface

PUC-Rio - Certificação Digital Nº 1322105/CA

IUrlBuilde. Finalmente, o *corpus* é empacotado usando o módulo IZipBinaryCreator.

**Desempenho da ferramenta:** procuramos uma maneira de tornar a ferramenta disponível, apesar da restrição de 5000 requisições por hora para cada endereço IP. Assim, se criou um *proxy* que simula vários endereços IP. O código fonte está disponível em (https://github.com/nitanilla/github-proxy.git).

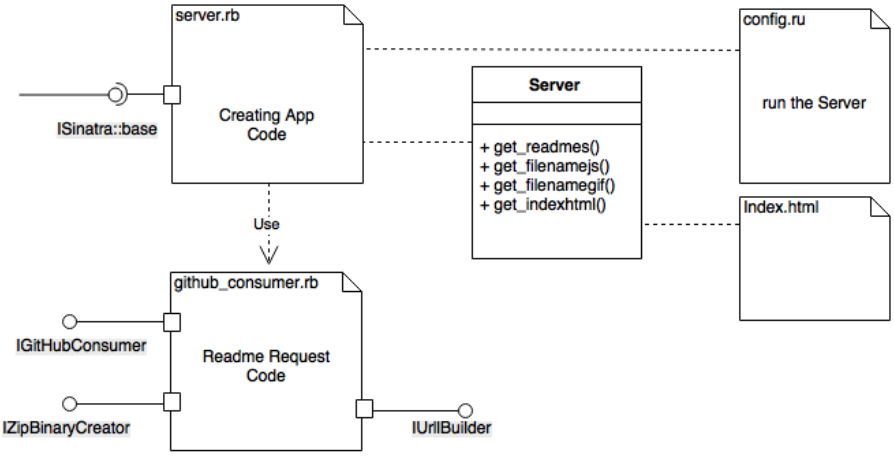



**Figura 13. Principais Componentes da Arquitetura**

### 3.2.3. Estratégia

A fim de trazer mais de 1000 resultados na busca, e com isto, melhorar o *recall* do que é obtido convencionalmente pelo site de GitHub, a ferramenta explora cinco das possíveis ordenações dos projetos que um usuário pode fazer: *bestmatch*, *most starts*, *fewest starts*, *most forks*, e *fewest forks* (Figura 14). Cada ordenação torna-se uma nova consulta. Nos casos de *fewest starts* ou *fewest forks*, os resultados são mostrados na ordem oposta. Com a versão 3.0 da API de GitHub, automatizamos em uma única tarefa, as 5 consultas que tem que fazer-se para as 5 ordenações.

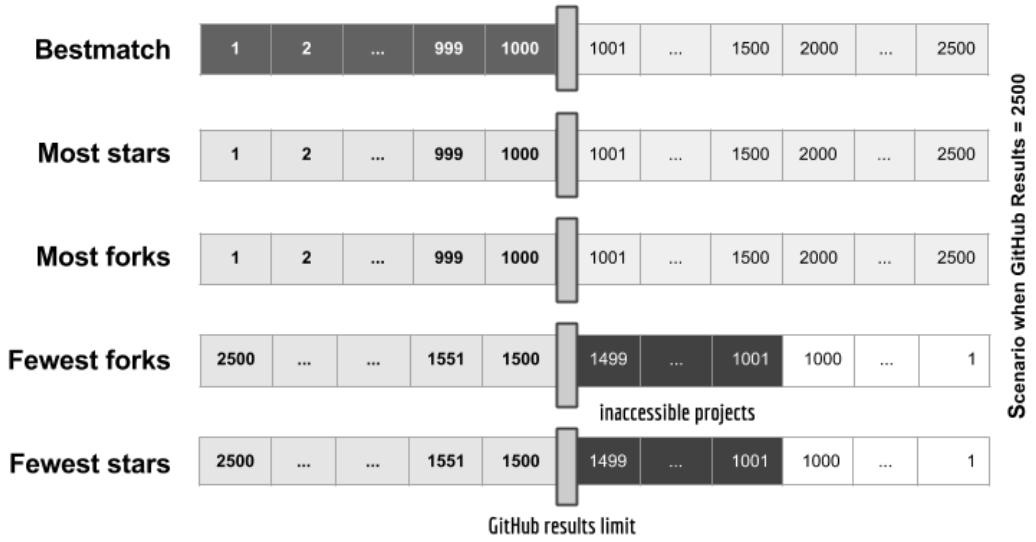



**Figura 14. Recuperação de readmes combinando opções de ordenação**

PUC-Rio - Certificação Digital Nº 1322105/CA

Existe a possibilidade de não recuperar muitos projetos, isto por causa da restrição de GitHub em disponibilizar apenas os primeiros 1000 resultados de qualquer consulta de ordenação, depois disso, não é garantida a ordem de relevância projetos. A perda da ordem de relevância impacta, pois, nas consultas por ordenação, em particular as que são inversas *fewest starts* or *fewest forks*, tem projetos (desde 1499 até 1001) que serão perdidos (Figura 14). Outra preocupação, é que um usuário pode avaliar um projeto dando uma *star* ou um *fork* num mesmo projeto, resultando em projetos com ambas as classificações; portanto, um projeto pode aparecer em duas consultas de ordenação. Nossa estratégia usa uma série de operações de união, a fim de capturar essas interseções para evitar projetos repetidos (Figura 15). Cabe destacar que a situação ideal seria a recuperação de todos os *readmes* para as cinco ordenações (até mesmo os vermelhos). De qualquer forma, consideramos que esta estratégia consegue uma melhora do *recall* dos resultados de GitHub, recuperando mais projetos relevantes quando comparado com o que o usuário obteria se fizesse uma busca manual.

| Bestmatch (1000) | Most Stars | Most Forks | Fewest Stars | Fewest Forks |
|---|---|---|---|---|
| **B** | **MS** | **MF** | **FS** | **FF** |

(((B **U** MS) **U** MF) **U** FS) U FF)

**Figura 15. Ordenação de projetos recuperados de GitHub**

### 3.2.4. Relevância dos documentos

A relevância é definida pelo ranking dos readmes obtidos, que é dado pelo critério de *bestmatch*. Cabe lembrar que o uso das outras ordenações (*stars e forks*) só acrescenta os *readmes* que se colocam depois do resultado número 1000. Finalmente, a ferramenta cria os traços dos *readmes* para suas fontes (Figura 16), nomeando cada documento por seu número de ordenação, seguido do nome de usuário, nome do projeto, e a extensão de arquivo conforme encontrado no GitHub.

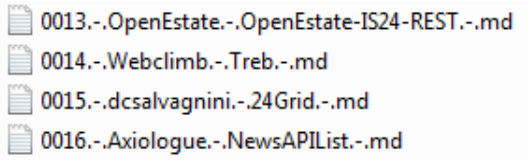



**Figura 16.Traçabilidade dos Readmes para os seus repositórios.**

### 3.2.5. A Ferramenta

A apresentação é simples e voltada para o propósito de *download* de um corpus de *readmes* dada uma consulta (Figura 17). A ferramenta está disponível na seguinte URL (http://corpus-retrieval.herokuapp.com/), e a divulgação do código no

PUC-Rio - Certificação Digital Nº 1322105/CA

GitHub, segue o espírito do GitHub, que visa atrair investigadores para uma maior colaboração na melhoria da ferramenta.

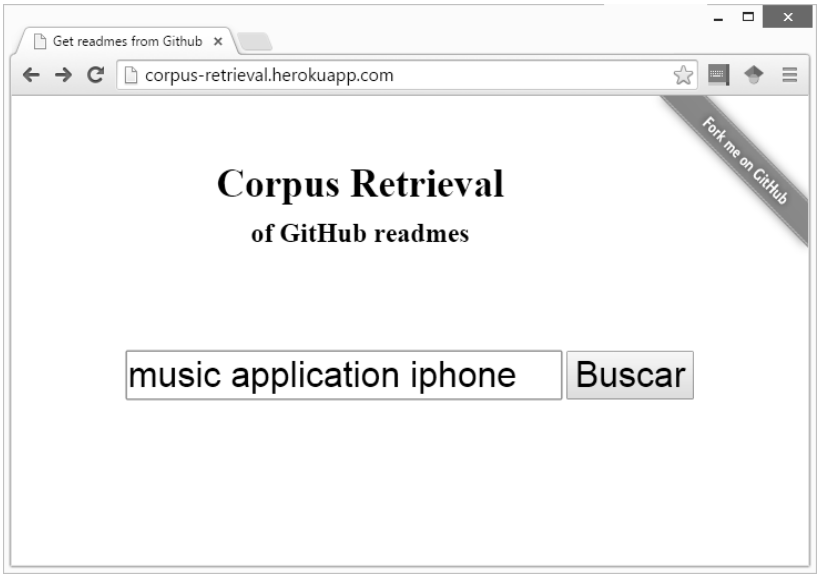



**Figura 17. Aplicação Web para a construção de Corpus de Readmes**

PUC-Rio - Certificação Digital Nº 1322105/CA

### 3.2.6. Uso de um Corpus de documentos

Após a criação de um *corpus* de *readmes*, os textos podem ser utilizados para importação em ferramentas de análise qualitativa, tais como o Atlas.ti, na qual o usuário é auxiliado para realizar anotações (manualmente) de abstrações importantes em cada documento.

As imagens a seguir (Figuras 18 e 19) mostram a importação dos textos de um corpus, as anotações e codes criados para o exemplo, assim como a organização automática de tais anotações em forma de rede semântica.

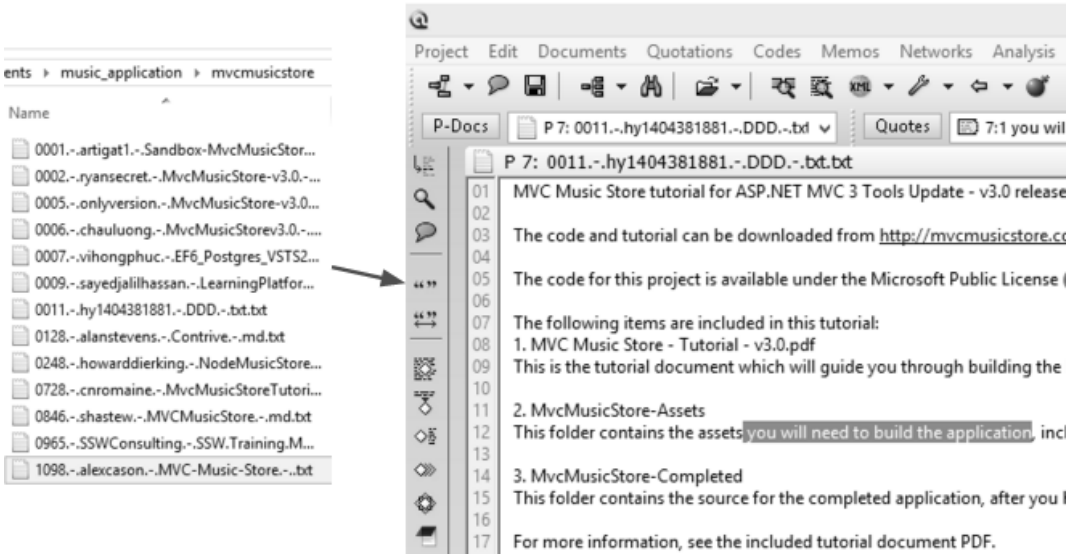



**Figura 18. Importação de documentos no Atlas.ti**

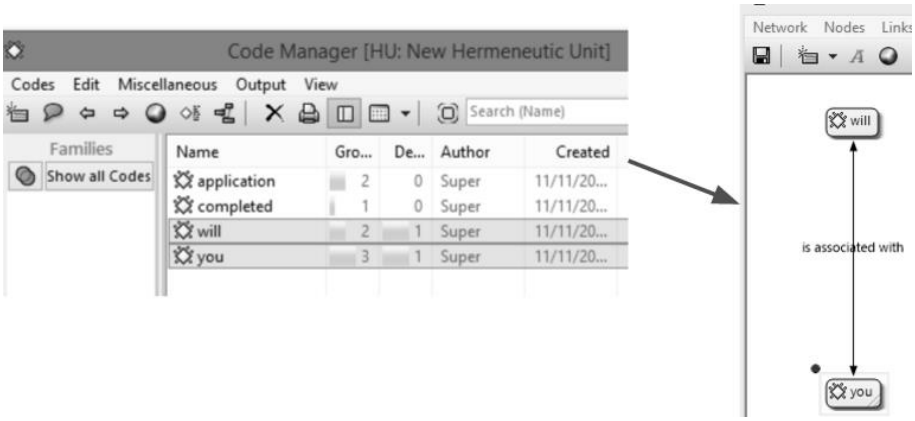



**Figura 19. Criação de *codes* e redes semânticas.**

Nossa pesquisa explorou também o R Project, uma vez que tal ambiente permite análise qualitativa e quantitativa dos dados, assim com o reuso dos pacotes para a criação de script que permitam o tratamento de um *corpus* e a extração das IRR. Em R, um *corpus* pode ser processado para diferentes tipos de analise, assim, é possível criar *datasets*, como o exemplar da (Figura 19) que é um *dataset* conhecido de vinhos, que contem 3 classes, 13 atributos, e 63 instancias. Esse *dataset* é usualmente utilizado em testes de algoritmos de *machine learning*[64].

```
Class, Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Total
phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Color
intensity, Hue, OD280/OD315 of diluted wines, Proline

1,14.23,1.71,2.43,15.6,127,2.8,3.06,.28,2.29,5.64,1.04,3.92,1065

1,13.16,2.36,2.67,18.6,101,2.8,3.24,.3,2.81,5.68,1.03,3.17,1185

2,12.69,1.53,2.26,20.7,80,1.38,1.46,.58,1.62,3.05,.96,2.06,495

2,12.29,2.83,2.22,18,88,2.45,2.25,.25,1.99,2.15,1.15,3.3,290

2,11.62,1.99,2.28,18,98,3.02,2.26,.17,1.35,3.25,1.16,2.96,345

3,12.36,3.83,2.38,21,88,2.3,.92,.5,1.04,7.65,.56,1.58,520

3,13.69,3.26,2.54,20,107,1.83,.56,.5,.8,5.88,.96,1.82,680

3,12.85,3.27,2.58,22,106,1.65,.6,.6,.96,5.58,.87,2.11,570
```

**Figura 20. Exemplo de *dataset* no R Project**

No caso dos *readmes*, para alcançar tal formatação é preciso a transformação de um *corpus* em matrizes de documentos por termos (DTM), ou termos por documentos (TDM), criando assim um *bag of words*. A (Figura 21) por questões de espaçamento, mostram apenas 3 termos dos 6857 existentes.

| Docs | Terms wp-property | Terms www.fivecornersrealestate.com | Terms would |
|---|---|---|---|
| 0001.-.PaulCrickard.-.Turf.js-Book.-.md.txt | 0 | 0 | 0 |
| 0002.-.pariser.-.goog-shrink.-..txt | 0 | 0 | 0 |
| 0004.-.drewcollinsme.-.tv-vision.-.md.txt | 0 | 0 | 0 |
| 0005.-.Jaspworld.-.treb2.-.md.txt | 0 | 0 | 0 |
| 0006.-.deviodigital.-.landingpage-realestate.-.md.txt | 0 | 0 | 0 |
| 0007.-.rjtedge.-.listedprices.-.md.txt | 0 | 0 | 0 |
| 0008.-.TheConnMan.-.N1-Disable-Important-Indicators.-.md.txt | 0 | 0 | 0 |
| 0009.-.amyhua.-.bananatext.-.md.txt | 0 | 0 | 0 |
| 0010.-.ssinfinitus.-.Vishal-Khandelwal.-.md.txt | 0 | 0 | 0 |
| 0012.-.JoshAdamous.-.alante.-.md.txt | 0 | 0 | 3 |
| 0013.-.OpenEstate.-.OpenEstate-IS24-REST.-.md.txt | 0 | 0 | 0 |
| 0014.-.Webclimb.-.Treb.-.md.txt | 0 | 0 | 0 |
| 0015.-.dcsalvagnini.-.24Grid.-.md.txt | 0 | 0 | 0 |
| 0016.-.Axiologue.-.NewsAPIList.-.md.txt | 0 | 0 | 0 |

**Figura 21. Readmes tratados no R Project**

PUC-Rio - Certificação Digital Nº 1322105/CA

E importante mencionar que a quantidade de dados a ser processada pelo R é abundante (Figura 22). Por exemplo, para um corpo de 308 *readmes*, a dimensão é de 6857 termos. O tratamento desse tipo de vetores não é simples, por tal, o processo criado para dar suporta à mineração, visa ajudar na redução de tal vector com atributos (termos) que o caracterizem, de maneira que as dimensões sejam menos esparsas do que no início (Figura 22).

```
> dtm
<<DocumentTermMatrix (documents: 308, terms: 6857)>>
Non-/sparse entries: 14830/2097126
Sparsity           : 99%
Maximal term length: 233
Weighting          : term frequency (tf)
```

**Figura 22. Sumario da matriz DTM para 308 Readmes**

Uma vez que um *dataset* é criado, é possível fazer um analise quantitativo dos dados, tal como encontrar as palavras que ocorrem pelo menos cinco vezes (Figura 23). Essa função, como outras quantitativas para *text-mining*[53] (TM).

```
> findFreqTerms(dtm,5)
  [1] "add"           "agency"          "agent"           "agents"
  [5] "align"         "alt"             "amount"          "analysis"
  [9] "another"       "api"             "app"             "application"
 [13] "apps"          "available"       "base"            "based"
 [17] "basic"         "bootstrap"       "build"           "building"
 [21] "business"      "can"             "center"          "cert"
 [25] "check"         "classified"      "classifieds"     "clients"
 [29] "clone"         "cms"             "code"            "com"
 [33] "commercial"    "community"       "companies"       "company"
 [37] "connection"    "contact"         "content"         "contribute"
 [41] "copy"          "course"          "coverage"        "create"
 [45] "data"          "date"            "developer"       "developers"
 [49] "development"   "documentation"   "don"             "easy"
 [53] "element"       "elements"        "estate"          "estates"
 [57] "estimation"    "example"         "exceptional"     "features"
 [61] "figcaption"    "figure"          "file"            "files"
```

**Figura 23. Termos frequentes no corpus de readmes de *Real Estate***

## 3.3. Resumo

Definimos um processo para a coleta de informações e sua mineração. Dito processo foi baseado na literatura de técnicas de *text-mining* para o entorno R. e abordagens para minerar informações em documentação de Requisitos. A automatização da primeira macro atividade *Collate* é detalhada na construção de um *script*, que cria um corpus de *readmes*, dada qualquer consulta de busca. Finalmente, mostramos brevemente como um *corpus* de textos semiestruturado pode ser utilizado tanto por ferramentas de análise qualitativa de dados, como o *Atlas.ti*, como também por ferramentas de análise quantitativa, como o *R Project*.

PUC-Rio - Certificação Digital Nº 1322105/CA

# 4
# Filtragem das informações

Neste capitulo apresentamos duas abordagens para abstrair as informações relacionadas a requisitos (IRR). A primeira, utiliza técnicas de processamento de linguagem natural (4.1) e a segunda utiliza regularidades independente do domínio, chamado também de padrões (4.2). Apresentamos a extração das informações e uma organização preliminar delas. Esta atividade está alinhada com o segundo macroprocesso *Distill* (Figura 9).

## 4.1.
## Filtragem por *POS-tagging*

No trabalho de Sawyer et al.[10], são utilizadas uma combinação de técnicas para processamento de linguagem natural, para que, dado um grupo de documentos diversos como: entrevistas, atas, manuais, documentação de sistemas existentes entre outros, seja possível a abstração de as propriedades importantes, que permitam um maior entendimento de um domínio. Esta pesquisa, mistura várias abordagens de NLP, assim como o *frequency profiling* utilizado no trabalho de Goldin & Berry[46], que ajudaram na precisão nas das IRR abstraídas.

### 4.1.1.
### Identificação de termos frequentes

Identificação de termos frequentes, como visto no trabalho de Sawyer, et al. [10], tem a desvantagem de que as palavras importantes para um domínio não ocorrem frequentemente em textos não estruturados. Por tratar o assunto, se adotou o esquema tf-idf[60], que é uma estatística numérica que permite refletir o quão importante uma palavra de um texto é em um *corpus*. A seguir, se descreve os passos para identificar os termos frequentes para o *corpus* de readmes do *Real Estate*, assim como a comparação de resultados de ambos esquemas *tf* e *tf-idf*:

Passo 1: utilizamos a atividade *Transform* do processo (3.3.1.2), no qual o corpus é reduzido até mais de 50% em número de palavras:

```
Corpus sem preprocessing: <<TermDocumentMatrix (terms: 54762, documents: 1141)>>
Corpus com preprocessing: <<TermDocumentMatrix (terms: 20854, documents: 1141)>>
```

Passo 2: Sumarização (3.1.2) do *corpus* usando os esquemas tf e tf-idf em R

```
Frequência com o Esquema TF:
 real   estate   can    use    data   will
 1887   1772     1297   847    820    819

Frequência com o Esquema TF-IDF:
 app        website    project    realestate   web        application
 28.74506   28.51921   25.47830   22.40085     21.90195   21.78595
```

PUC-Rio - Certificação Digital Nº 1322105/CA

Passo 3: A criação de uma nuvem de termos (top 40) com ambos esquemas.

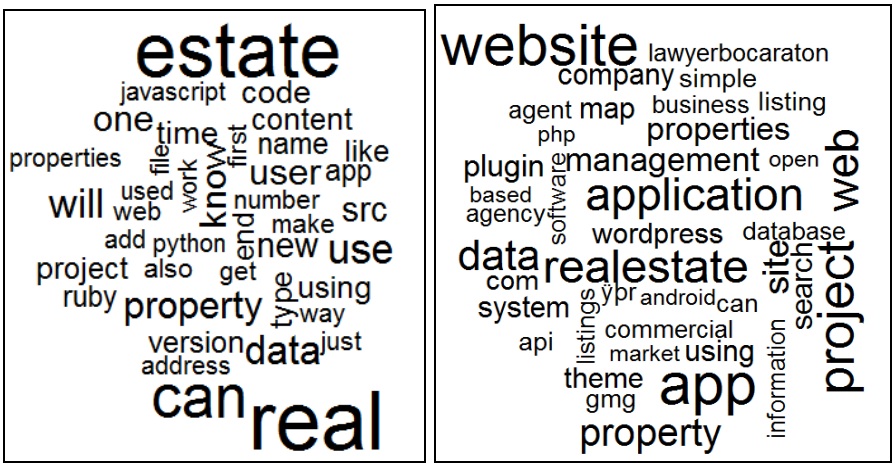



**Figura 24. Termos frequentes. TF (esquerda) e TF-IDF (direita)**

No passo 2, é possível identificar uma melhora nos resultados com o esquema *tf-idf*, pois ele apresenta, como termos frequentes, mais entidades (substantivos). Na (Figura 24) é possível visualizar mais palavras frequentes, e a representação da frequência pelo tamanho de letra. Nesse sentido, o esquema *Tf-idf* é aquela que nos mostra mais termos relevantes, pois vários desses termos, são parte da terminologia do domínio Real Estate; palavras como: Agent, Listing, Properties, Agency, nos dão uma ideia do tipo de projetos que existem no domínio de *Real Estate*. Em contrapartida, a esquema TF mostra apenas o termo Property. Devemos recordar que a escolha das 40 palavras foi feita sobre um corpus de 20.840 palavras, por tal, existem mais palavras relevantes a ser mostrados, sem embargo, a representação visual seria sobrecarregada, o qual impacta na leitura do elicitador.

### 4.1.2. Identificação de substantivos

Esta abordagem surgiu ao analisar a forma gramatical de palavras como (Agent, Listing, Properties, Agency) os quais sendo do tipo *noun* (NN) podem ajudar ao leitor a identificar rapidamente as entidades existentes em um corpus. Assim, utilizando a técnica *POS tagging*, se extraíram as palavras desse tipo. Para verificar como a abstração deste tipo de palavras pode ajudar na filtragem de projetos relevantes, criamos um protótipo de nuvens de palavras (Figura 25) (https://nitanilla.shinyapps.io/munaysipas/), o qual pode ajudar a localizar projetos de interesse pela frequência de uma palavra no texto *readme*.

PUC-Rio - Certificação Digital Nº 1322105/CA


Inserir algum termo da nuvem... listing

| readme | termosFreq |
|---|---|
| 0912.-.17.-.wp-plugins.-.easy-property-listings.-.txt.txt | 106 |
| 1029.-.2.-.arcticleo.-.reso_data_dictionary.-.md.txt | 91 |
| 1477.-.8.-.wp-plugins.-.real-estate-listing-realtyna-wpl.-.txt.txt | 32 |
| 0637.-.2.-.agentevolution.-.wp-listings.-.txt.txt | 31 |
| 1251.-.2.-.wp-plugins.-.wp-listings.-.txt.txt | 31 |
| 0354.-.9.-.Placester.-.Community-App.-.txt.txt | 29 |
| 1366.-.9.-.wp-plugins.-.placester.-.txt.txt | 29 |
| 0831.-.8.-.wp-plugins.-.wprealestate.-.txt.txt | 28 |
| 1501.-.7.-.iplaydu.-.real-estate-listing-realtyna-wpl.-.txt.txt | 23 |

**Figura 25. Nuvem de substantivos mais frequentes dos textos Readme**

### 4.1.3. Organização de termos frequentes

Tendo que a identificação IRR pode ser alcançada se filtramos *nouns*, *verbs*, *adverbs* ou frases (*boilerplates*), foi desenhada uma estratégia para extraí-los de maneira ordenada, e esteja de acordo com a atividade do *Collocate* (3.1.3).

Como primeiro passo (Figura 26), teríamos a criação de várias nuvens, segundo a *tag* da palavra. Nesse ponto, se calculam os termos frequentes para cada nuvem. Isto pode levar a filtrar os *readmes* segundo a *tag* selecionada. Uma vez que identificado o grupo de interesse, é possível filtrar um subconjunto de projetos. Um segundo passo vem a ser a identificação das IRR no subconjunto de *readmes*

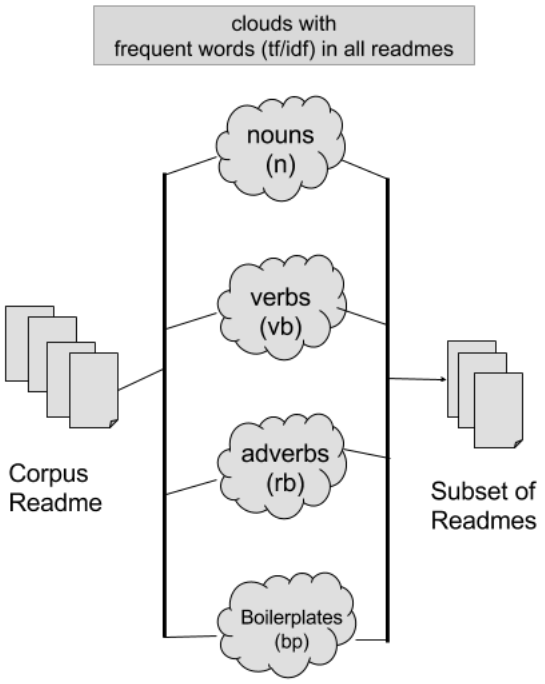



**Figura 26. Proposta para organizar os termos frequentes**

PUC-Rio - Certificação Digital Nº 1322105/CA

## 4.2.
## Filtragem por Boilerplates

Regularidades, padrões ou *templates* de requisitos são também chamados de *Boilerplates*. Segundo Arora et al.[50]:

> "Boilerplates fornecem padrões para a estrutura gramatical de frases de requisitos. Boilerplates facilitam ainda mais a análise automatizada sobre os requisitos em linguagem natural".

Nesse sentido, mesmo que este trabalho não esteja tratando com documentos de requisitos propriamente ditos, os artefatos *readmes*, *issues*, or *issues comments*, possuem textos com estereotipo de requisitos. Exemplar de *boilerplates*, tipicamente utilizado em especificações de requisitos são:

*"the system will be able to process…"*
*"the system should provide..."*

Tais frases, podem ser construídos segundo a estrutura proposta por Arora et al.[50], (Figura 21). A filtragem por *boilerplate* pode resultar em uma maior precisão das IRR em *readmes*, dado que uma *noun* pode estar ancorada em uma frase não relacionada a requisitos. Sem embargo, as chances de uma frase frequente para descrever funcionalidades, podem alcançar uma maior precisão.

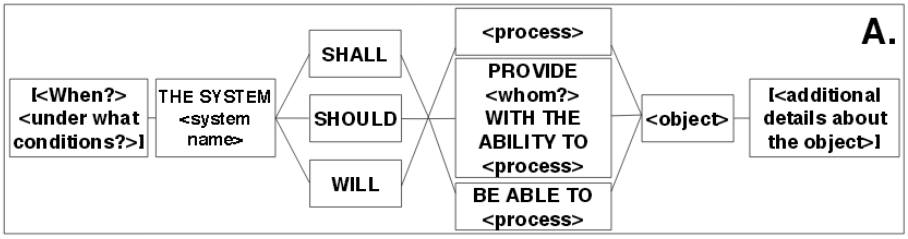



**Figura 27. Rupp's Boilerplates, Arora et al. [50]**

No caso desta pesquisa, a exploração dos readmes permitiu encontrar estruturas similares aos *boilerplates*. Isto seria, como denominado por Ridao & Leite[65] *"regularidades independentes do domínio"*, os quais permitiram ancorar as IRR. Por exemplo, uma das estruturas encontradas nos *readmes*, é:

**As a user** [subject] **I want** [verb] **so that** [goal/estate]

Em tal estrutura, identificamos que verbos frequentes como (want, can, might, shall, should, would, allows, permit), de tipo gramatical *modal verb*, podem ser identificadas utilizando a técnica de *POS-Tagging.* Palavras frequente, é também, uma estratégia contemplada por Leite & Franco[18] na Elicitação de Requisitos. Nessa estratégia, é construído um vocabulário, o qual captura o jargão usado pelos especialistas de um domínio. Eles propõem o seguinte processo para elicitar o léxico de uma aplicação:

PUC-Rio - Certificação Digital Nº 1322105/CA

Primeira atividade: a identificação dos símbolos da linguagem em (entrevistas, observação, ou leitura de documentos). As tarefas desta primeira atividade são:

a. Identificação de palavras-chaves (símbolos) utilizadas com frequência (léxico)
b. Não procurar identificar funções da aplicação observada, mas apenas os seus símbolos.

Segunda atividade: procurar o que cada símbolo significa, através da entrevista de atores da aplicação. É importante notar, que nossa abordagem não considera atores que validem o significado dos símbolos minerados. Nesse sentido, vislumbramos a correlação de símbolos com ontologias existentes de um domínio, o qual poderia ajudar na precisão da identificação de *readmes* mais relevantes, bem como na composição de melhores consultas.

As abordagens mencionadas influenciaram na estratégia descrita a seguir: Dado que as estruturas de *boilerplates* proposta por Arora et al.[50], foram extraídas pelo analise em documentos de especificações de requisitos, se pensou na criação de 4 boilerplates baseados nessa estrutura (Figura 29), com isso, e para evitar o viés no processo, se escolheu um domínio diferente do Real Estate. Se construiu um *corpus* para a consulta "*digital library*", os 1219 *readmes* recuperados foram minerados com o mecanismo *phrase query* definido a atividade *Distill* (3.1.2) do processo, para finalmente obter as seguintes frequências:

**Tabela 3. Frequência de boilerplates no *corpus* "digital library"**

| Should provide | Should be able to | Will provide | Will be able to |
|---|---|---|---|
| 1 | 35 | 11 | 7 |

Tais números (menos de 5% de documentos coincidindo algum *boilerplate*, indicavam que, de fato, esses *boilerplates* são eficazes quando a mineração é feita sobre documentos de especificações de requisitos. Assim, resgatando-se a ideia dos *boilerplates*, se pensou na identificação manual de regularidades em uma mostra de textos *readmes*. Partindo das regularidades já observadas na pesquisa exploratória, "*As a user" e "I want",* se escolheram aleatoriamente 291 readmes para a consulta "*music application*", e se identificaram as seguintes regularidades (Tabela 7). É importante destacar que conforme as regularidades se mostravam, elas foram anotadas, ou seja, nem sempre a intenção foi a procura de frases exatas, ou frases predefinidas.

### 4.2.1. O processo

A identificação de boilerplates seguiu uma abordagem estatística. Para tal, a partir da consulta "music application" com um total 1206 *readmes*, se selecionou uma mostra de 291 readmes. A seleção, com uma margem de erro de 0.05, foi aleatória, para o qual se criou script que foi publicado no GitHub para futuros testes (https://github.com/nitanilla/Random-Readme).

PUC-Rio - Certificação Digital Nº 1322105/CA

Começou-se agrupando os readmes pelo seu tamanho (Tabela 5): Grupo A. 0kb <= 1kb; Grupo B. 1kb <= 2kb; Grupo C. de 2kb <= 6kb; Grupo D. de 6kb <= 200kb. Dita agrupação por tamanhos foi com a intenção de balancear o número de *readmes* em cada grupo.

**Tabela 4. Número de mostras para cada grupo de readmes**

| | Grupo A | Grupo B | Grupo C | Grupo D | Total |
|---|---|---|---|---|---|
| **Total readmes** | 865 | 127 | 107 | 107 | 1206 |
| **Porcentagem** | 72% | 11% | 9% | 9% | 100% |
| **Número de mostras** | 209 | 31 | 26 | 26 | 291 |

A leitura de readmes tomou em média 25 horas distribuídas em 5 dias. O Grupo A com 209 readmes levou 10 horas distribuídas em 3 dias. O grupo B, 8 horas distribuídas em 2 dias. O grupo C, 5 horas em um dia. Finalmente o grupo D, levou 2 horas em um dia só. A medida que a leitura de readmes ocorria, se pensou que uma classificação dos readmes poderia ser feita. Assim, as possíveis *tags* (Tabela 5) para classificar um *readme* são: readmes que contem IRR, tem a *tag requirement; readmes* que explicam instruções de instalação tem a *tag development; readmes* que não possuem descrições além do título, ou não fornecem a suficiente descrição informativa, tem a tag *Not IRR.*

**Tabela 5. Classificação dos readmes segundo o seu conteúdo**

| Critérios | Exemplo | Tag |
|---|---|---|
| O conteúdo é mesma query | Music Application | Not IIR |
| O conteúdo não fornece compreensão do que o repositório faz. | Music lab Application | Not IRR |
| Conteúdo não é relacionado ao domínio objetivo | Library for node applications | Not IRR |
| Pelo menos, o texto explica que o projeto faz. | Social music application | Requirement |
| O texto é mais entendível para um leitor desenvolvedor. | Add ex music sandbox to your list of dependencies in ‘mix.exs‘ | Development |

O resumo (Tabela 6) dos grupos de *readmes* com a quantidade de *tags* identificadas, é descrita a seguir. É importante destacar a linha que indica o número de textos que não pertencem ao domínio objetivo, como já visto para o caso do *Real Estate.*

**Tabela 6. Classificação dos readmes pelo seu discurso**

| Tags | Group A | Group B | Group C | Group D |
|---|---|---|---|---|
| Requirements | 165 | 26 | 23 | 12 |
| Development | 46 | 19 | 16 | 9 |
| Not IRR | 22 | 2 | 0 | 1 |
| Not a music Application (not IRR) | 0 | 0 | 0 | 15 |

PUC-Rio - Certificação Digital Nº 1322105/CA

A seguir os *boilerplates* identificados:

**Tabela 7. Boilerplates identificados e sua frequência no *corpus***

| Boilerplate | # aparecimentos nos readmes | Boilerplate | # aparecimentos nos readmes |
|---|---|---|---|
| “can be used to” | 41 | “used to” | 164 |
| “I want to” | 36 | “allow you” | 16 |
| “allows for” | 20 | “allow user” | 14 |
| “that allows you to” | 31 | “to allow” | 54 |
| “allows users” | 27 | “to provide” | 77 |
| “you can” | 926 | “that can” | 120 |
| “intended for” | 23 | “one can” | 27 |
| “users can” | 29 | “which can” | 57 |
| “can do” | 60 | “that lets you” | 17 |
| “lets you” | 44 | “to create” | 279 |
| “enable” | 312 | “features” | 299 |
| “project is a” | 19 | “project to” | 32 |
| “designed for” | 34 | “designed to” | 66 |
| “that shows” | 22 | “goal” | 92 |
| “to manage” | 61 | “you should” | 97 |
| “it should be” | 23 | “should be able to” | 38 |
| “should be” | 222 | “in order to” | 76 |
| “can be” | 655 | | |

Temos as seguintes observações acerca da leitura de *readmes*:

1. Grupo A: vários textos continham imagens e elos para as versões em produção dos projetos. Isso, se for minerado, melhoraria a experiência do usuário ao localizar o repositório de interesse a ser reusado.

2. Grupo B: Os textos foram mais detalhados do que no grupo A, incluindo os propósitos, as características (*features),* e também as instruções de instalação.

3. Grupo C: similares ao grupo B, contendo a mais informações sobre os erros do projeto.

4. Grupo D: a maioria dos textos continham instruções de instalação, textos longos, com muito detalhamento de instruções para desenvolvimento. A maioria dos textos não estava referido ao domínio “music application”.

Uma observação relevante da experiência foi que, cada *corpus* de documentos poderia ter seus próprios *boilerplates*; a tarefa de identificá-los, foi definido na atividade *Distill* do processo (2.1.7) como *Recognition of patterns*. Por outro lado, existem palavras latentes relacionadas ao domínio, como por exemplo: Spotify, Last.fm, Soundcloud, Google play music, Facebook, Twitter, entre outros, que, sem ser palavras diretamente relacionadas a o vocabulário do domínio, podem ser uteis para agrupar (*clusterizar*) projetos. Vislumbramos aqui

PUC-Rio - Certificação Digital Nº 1322105/CA

uma mistura entre: a identificação de documentos com ditas palavras, de tipo substantivo próprio, como também a identificação de entidades (substantivos), os quais ao ser mapeados com ontologias do domínio, existentes em bases de conhecimento externas como Wikipédia, melhoraria as oportunidades de encontrar palavras do domínio, e com isto dar maior precisão à extração de IRR.

### 4.2.2. Organização de projetos usando ambas filtragens

Após a identificação de *boilerplates*, e utilizando a primeira filtragem por *POS tagging*, é possível combiná-las, seguindo os fluxos da (Figura 28).

1. Obter um subgrupo de readmes (Figura 26).

2. Identificar 4 tipos de frases nos *readmes*, cada frase pode ser ancorada pelas filtragens: (a) *boilerplates*, (b) substantivos frequentes, (3) verbos frequentes, (4) advérbios frequentes.

3. Identificar termos frequentes sobre as frases, contudo, utilizando o esquema *tf*, pois nesse momento já teríamos frases num documento só, por isso o esquema *tf-idf* não seria aplicável. Alternativamente, é possível criar nós, reconhecendo automaticamente os substantivos: "*system", ou* pronomes pessoais como *"you"* nas frases. Finalmente, o passo final é organizar as frases ancoradas por tais nodos.

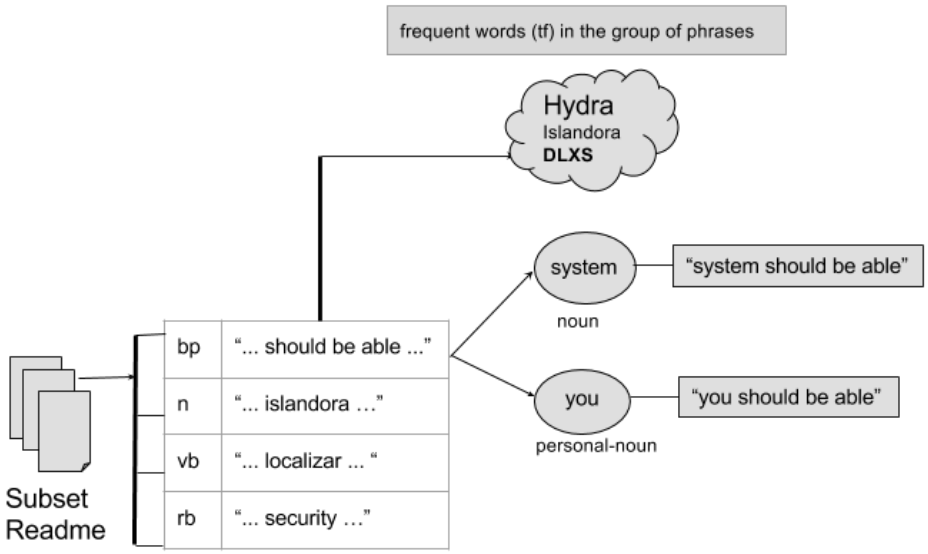



**Figura 28. Organização de abstrações utilizando ambas filtragens**

É importante mencionar que as organizações propostas devem ser testadas antes da criação de uma ferramenta completa para exposição das IRR, dado que, a forma da visualização de informação pode impactar na adoção de um produto deste tipo.

PUC-Rio - Certificação Digital Nº 1322105/CA

## 4.3. Resumo

Descrevemos as estratégias para ancorar informações relacionadas a requisitos, seja pelo *POS-tagging*, que é dado segundo a estrutura gramatical das palavras, ou pelos *boilerplates* (padrões). Se descreveu como ambas estratégias podem ser misturadas para organizar os readmes, isto é, primeiro etiquetando cada palavra do *readme* com o POS-Tag, agrupa-los pelo POS-tag, achar a frequência das palavras em cada grupo, para depois filtrar os readmes (Figura 27). Outra organização, é primeiro filtrando os *readmes* que possuam *boilerplates***,** filtrar a frase ancorada pelo *boilerplate*, para depois encontrar os termos frequentes nos *readme*s filtrados (Figura 28).

PUC-Rio - Certificação Digital Nº 1322105/CA

# 5
# Analise das Informações

Neste capítulo apresentamos um estudo de caso para experimentar as IRR extraídos com os *boilerplates* identificados. Se detalha o desenho do experimento, e o teste realizado com participantes de uma Startup de um aplicativo de música. Se mostram as avaliações de tais IRR, e finalmente, as lições aprendidas.

## 5.1.
## Motivação

Imagine a seguinte situação: um grupo de músicos está querendo produzir uma aplicação de música; eles acreditam que poderia ser um sucesso. Entraram em contato com investidores-anjo que estão dispostos a investir, mas eles precisam de mais detalhes sobre a ideia. Para tal, eles decidiram contratar uma empresa de ER para organizar os requisitos, antes de contratar uma empresa desenvolvedora de software para construir o aplicativo. A ideia geral dos músicos é que o usuário possa ter noção de uma cidade, um bairro ou de um lugar especifico como uma universidade através da música que está sendo escutada ao redor. Ocorre que a empresa de engenharia de requisitos contratada para fazer o trabalho, não está familiarizada com o domínio e teria que adquirir rapidamente conhecimento contextual para colaborar melhor com os músicos. Além disso, teria também que construir requisitos adequados para os futuros desenvolvedores. Este conhecimento contextual deve ser relacionado tanto com o lado do cliente, mas também para a possível ecologia de software onde o aplicativo irá funcionar.

Com o intuito de provar os *boilerplates* e as frases no contexto ancoradas por esses, procurou-se uma Startup relacionada ao domínio de música para que se avalie a utilidade de tais frases. Cabe destacar que a identificação de *readmes* contendo o *boilerplate* foi automática, e a extração das IRR no contexto de alguma *boilerplates,* foi realizada manualmente.

*Hear* (https://www.facebook.com/apphear/?fref=ts), é uma Startup de software que começou em julho de 2015 como parte do programa de desenvolvedores criativos de aplicações para a Apple (BEPID -PUC-Rio). A equipe, é formada por 4 estudantes de graduação das áreas de design e arte, sendo que um deles possui conhecimentos sobre desenvolvimento de software. O propósito de *Hear* é criar uma experiência musical baseada na geo–localização, assim, um usuário pode ser capaz de entender um lugar (cidade, bairro ou universidade) a partir da música que é escutada ao redor. Atualmente, eles estão trabalhando na versão 0.3 na fase de teste da *Apple Store*.

PUC-Rio - Certificação Digital Nº 1322105/CA

## 5.2.
## Desenho do caso de estudo

O experimento foi pensado em três fases: 1) piloto, 2) primeiro teste com três membros da equipe, 3) segundo teste com o membro desenvolvedor. O piloto foi planejado para verificar se as *tags* criadas para avaliação (Tabela 8), atendiam o que o usuário precisava dizer sobre uma frase. De outro lado, precisava-se conhecer a experiência de leitura de tais frases, em vista de que a apresentação das informações é um aspecto importante da atividade *Collocate* do processo. Nos testes, foram utilizados planilhas e formulários de Google para facilitar o acesso, e pela estatística dos resultados.

O piloto utilizou o boilerplate ***"can be used to"*** com 41 aparições. Com isto, nos mineramos as 41 frases (anexo 1), e solicitamos a um dos membros a avaliação de esses segundo os seguintes *tags*.

**Tabela 8. Tags para avaliação de frases**

| Tags |
|---|
| Already developed |
| 1.1. Which version was developed? |
| Being built for the current release |
| A good idea to take into consideration |
| Not interesting for now |

O teste foi realizado virtualmente, e levou em média, 20 minutos para a leitura das 41 frases. Como resultado temos que:

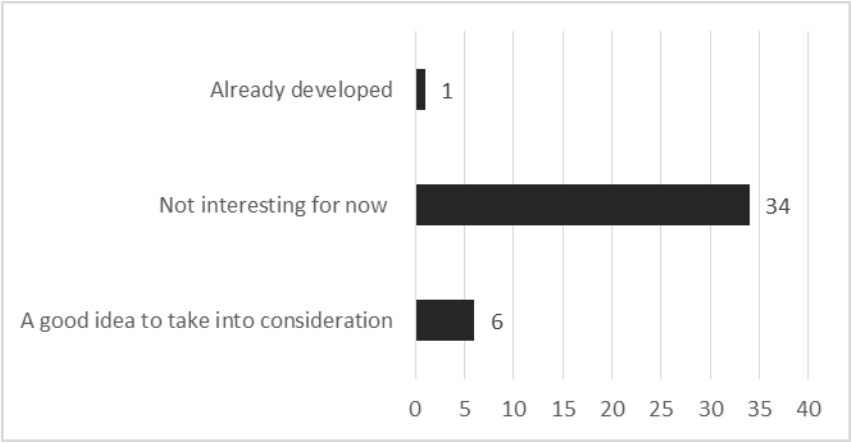



**Figura 29. Avaliação das frases no piloto de teste**

Foi perguntado o quão fácil foi a leitura das frases, para o qual o usuário respondeu: *"no começo parecia simples, mas, perto da frase número 30 os textos foram muito vagos e fora do contexto".* A seguir, foi perguntado se necessário a criação de uma *tag* extra, ele recomendou acrescentar as seguintes: *out of the context* e *not undestandable.* Ademais, nos foi comentado acerca da necessidade de demos, e de elos diretos das IRR para suas fontes no GitHub. Finalmente, foi expressado que, mesmo que esses sejam apenas frases, sua leitura se tornou uma tarefa tediosa.

Um dos comentários que chamou nossa atenção, foi respeito à única frase categorizada como *Already developed*:

PUC-Rio - Certificação Digital Nº 1322105/CA

*"Megastream streaming download of a file (can be used to preview videos or music)"*

Ele comentou que essa informação houvesse sido útil na versão 0.1 da aplicação, dado que, a pesar de ser obvio que a aplicação teve que reproduzir música, o fato de conhecer que o *streaming* é mais eficiente que o *downloading, foi* percebido apenas na versão 0.2.

## 5.3. Primeiro Teste

De acordo com o piloto, se fizeram as melhorias referente ao número de frases e a apresentação delas. Tomou-se o *boilerplate* ***"allow user"*** com 14 aparições, e se mineraram as IRR. Se escolheram aleatoriamente 8 IRR (Anexo 1.2) e foram disponibilizados num formulário de Google (http://bit.ly/music-app-1g-1test-allow). O link foi dado a três membros da equipe com os seguintes perfis: 1) analista de sistemas, 2) estudante de cinema, e 3) desenvolvedor de desenho. Sobre os resultados podemos julgar que as frases têm 16% de imprecisão (2 valorações são do tipo *I don't understand*, e outras 2 *not interesting for now*).

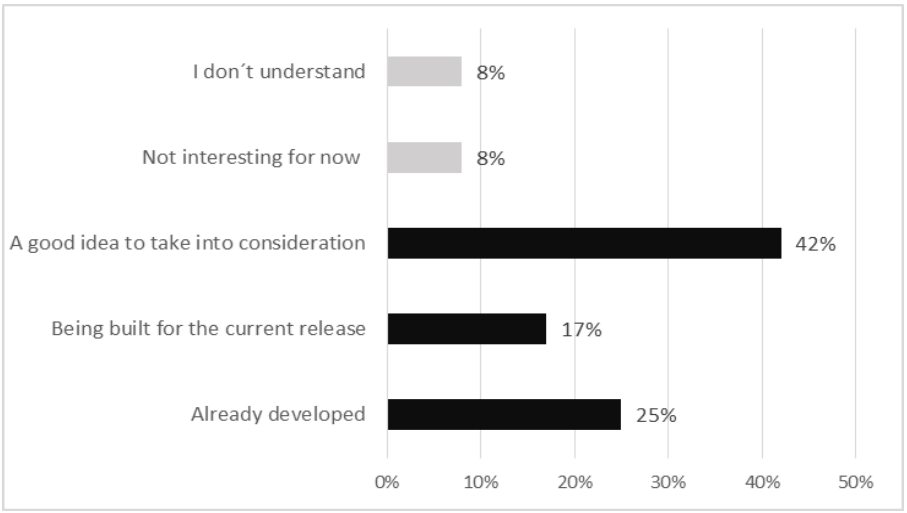



**Figura 30. Resumo de respostas do primeiro teste**

Apesar de ter majoritariamente repostas positivas, os resultados mostraram que os membros da equipe têm percepções diferentes do seu projeto, dado que em várias questões, cada um deles marcou uma opção diferente.

**Tabela 9. Valorações feitas pelos 3 membros da equipe a 10 frases**

| Valoração | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 |
|---|---|---|---|---|---|---|---|---|
| Already developed | 1 | 1 | 1 | 1 | 2 | | | |
| Being built for the current release | 1 | 1 | | | | | | 2 |
| A good idea to take into consideration | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 |
| Not interesting for now | | | | 1 | | 1 | | |
| I don't understand | | | | | | 1 | 1 | |
| Out of context | | | | | | | | |

PUC-Rio - Certificação Digital Nº 1322105/CA

De outro lado, mesmo que não existam repostas do tipo *out of the conte*xt, o que poderia indicar-nos que o *boilerplate **"allow user"*** tem uma boa precisão, não podemos afirmar isso, pois, ao testar tal *boilerplate* em outros domínios, resultou em informações vagas. Tal problema foi identificado também por Sawyer et al.[10] ao experimentar uma abordagem[66] de comparação de *corpus* de textos. Assim, vislumbramos que uma abordagem para extrair automaticamente os *boilerplates* que pertencem a um *corpus*, melhoraria a precisão das frases.

## 5.4. Segundo Teste

Para o último teste, utilizamos dois *boilerplates* **"*allows user*"**, **"*allow the user to*"**, e um verbo **"*allows*",** com a intenção de trazer mais IRR. No total, 11 frases foram avaliadas pelo membro com perfil desenvolvedor (Anexo 1.3). Das respostas, nenhuma das IRR ancoradas por verbos foi valorada positivamente. Das IRR ancoradas pelos *boilerplates* tem-se 1 identificada como fora de contexto.

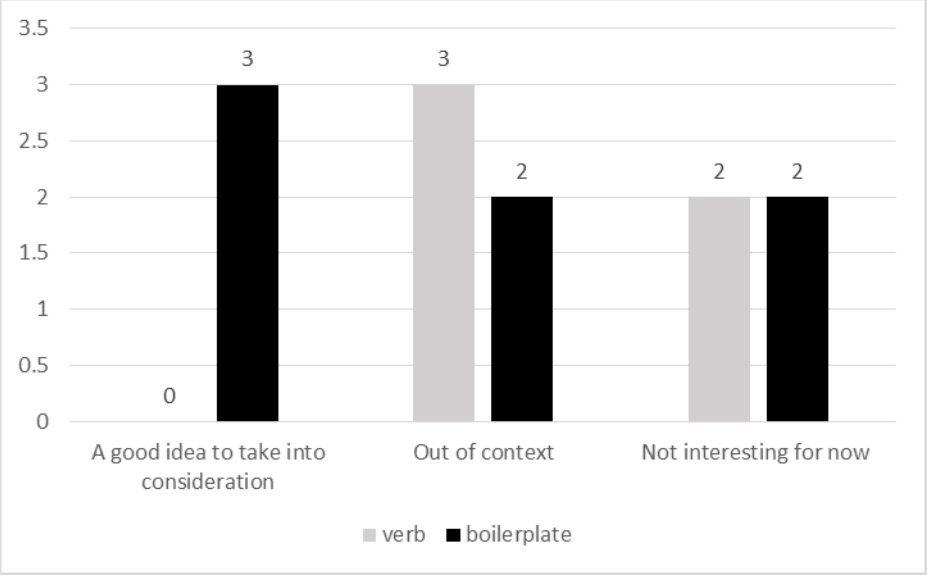



**Figura 31. Segundo teste do caso "music application"**

## 5.5. Teste de precisão

Com a finalidade de avaliar a qualidade das IRR ancorados pelos *boilerplates*, se escolheu o **"allow user"** para ser testado em 3 *corpus* diferentes para as seguintes consultas: Music Application, Real Estate, e Digital Library. Se mineraram os *readmes* contento aquele *boilerplate*, obtendo os seguintes valores:

**Tabela 10. Teste de precisão de boilerplates em vários domínios**

| Domínio | Nro de Readmes |
|---|---|
| Music application | 14 |
| Real estate | 5 |
| Digital library | 5 |

PUC-Rio - Certificação Digital Nº 1322105/CA

Revisamos cada *readme* para avaliar quantos deles estão relacionados a cada domínio. Temos como resultado que, o caso de Music Application tem mais readmes relacionados (21% de imprecisão), os casos de Real Estate e Digital Library tem menos *readme* relacionados (40% de imprecisão).

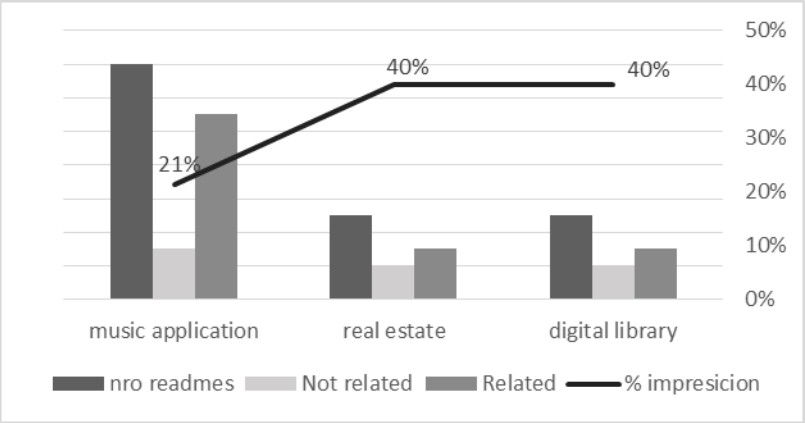



**Figura 32. R Teste de precisão de boilerplates em vários domínios**

## 5.6. Notas e comentários

Quando realizado uma busca convencional pelo site do GitHub com a query "music application" ou "music application in: readme", nenhum dos repositórios, dos quais as IRR foram extraídas, aparecem na primeira página de resultados. Assim, acreditamos que nossa abordagem de *boilerplates* melhora a precisão dos resultados.

Dos testes realizados, encontramos que a maioria das valorações são do tipo *A good idea to take into consideration*, em tal sentido consideramos que as IRR que podem ajudar para outro tipo de reuso, que é para ajudar na criatividade de projetos inovadores, como as Startups.

Do piloto realizado, nos conferimos que a apresentação das informações é um dos desafios para uma futura ferramenta de reuso de conhecimento para apoiar à Elicitação de Requisitos.

## 5.7. Ameaças

O experimento foi realizado utilizando a estratégia de triangulação[67], que consiste em considerar várias fontes. A triangulação *observer triangulation*, foi praticado com o intuito de verificar se existe concordância na avalição das frases presentadas. Como visto na avaliação da *tag* de tipo *Already developed*, foi possível capturar as discrepâncias (Tabela 8), isto é, os membros da equipe da Startup têm diversas percepções do próprio projeto.

Outra estratégia utilizada é a *Negative case analysis* abordada na seção (5.5), realizando o processo de mineração de *boilerplates* sobre outros *corpus* de textos. Isto como um controle para refutar a teoria, dado que, os *boilerplates* foram identificados no *corpus* da consulta "Music Application". Como visto na (Figura 32) a imprecisão incrementa quando o domino é distinto da fonte dos *boilerplates*.

PUC-Rio - Certificação Digital Nº 1322105/CA

Nos dividimos o caso de estudo em 3 fases: O piloto, o primeiro teste com três participantes, e o segundo teste com o participante mais involucrado no projeto. O piloto permitiu refinar os testes posteriores, e além disso, o piloto realizado com 41 frases deixou ver que existe uma necessidade para organizar melhor as informações (atividade *Collocate*) pois isto impacta no leitor.

Finalmente, Robson, C., & McCartan[67] indicam que a principal ameaça a fornecer uma descrição válida do que foi visto ou ouvido, reside na imprecisão ou imperfeição dos dados. Em Anexo 1, o detalhamento de cada fase é disponibilizado para análises posteriores.

## 5.8. Resumo

Descreveu-se os testes realizados para o caso da *Music Application*. Se descreveu em detalhe o desenho do caso de estudo e as diferentes estratégias para os testes. A análise dos resultados é feita destacando a precisão dos textos minerados.

PUC-Rio - Certificação Digital Nº 1322105/CA

# 6
# Conclusões e trabalhos futuros

Este capítulo apresenta as conclusões do trabalho. Uma breve comparação com trabalhos relacionados, as contribuições da pesquisa e sugere trabalhos futuros.

## 6.1.
## Conclusões

Berry et al.[68] no trabalho "*The case for dumb requirements engineering tools*" aponta que a questão de como separar uma tarefa em partes (o trabalho manual do trabalho de análise) é, naturalmente, uma das muitas questões que desafiam os desenvolvedores que utilizam NLP para a ER. Neste trabalho, ao considerar como fonte de informação o GitHub, o principal objetivo se tornou a procura de mecanismos que possibilitem o reuso dos seus conteúdos desde a visão de requisitos. Nesse sentido, a automatização da parte manual, como é a obtenção de projetos relevantes para abstrair IRR (Informações Relacionadas a Requisitos) de forma automatizada, pode ser alcançada com as heurísticas propostas nesta pesquisa. Com tais informações contribuímos para que as tarefas de elicitação (entrevistas, questionários, entre outros) sejam tratadas de melhor maneira em domínios desconhecidos pelo elicitador.

Acredita-se que a pesquisa exploratória levou a descoberta de propriedades em documentos de GitHub, o qual nos levou a definir a questão de pesquisa e a definição de um processo para explorá-lo. Em tal sentido, acreditamos que a mineração de informações em repositórios abertos, se torna em um forte aliado no processo de elicitação de requisitos, uma vez que mais informações estão sendo armazenadas com acesso aberto na web.

Com os resultados alcançados até agora estamos mais perto de construir uma maneira de usar informações não estruturadas ligadas ao código que apoiem na elicitação de requisitos, especialmente em projetos com restrições de tempo.

## 6.2.
## Trabalhos relacionados

As abordagens que têm sido estudadas mostraram os desafios e riscos ao lidar com documentos em linguagem natural. O trabalho *AbstFinder*[46] difere da nossa abordagem no conjunto de dados que eles mineraram, isto é, um *corpus* de documentos de especificações de requisitos. Nesse sentido, o desafio desta pesquisa é tratar com a imprecisão dos documentos recuperados, que não são requisitos propriamente ditos, mas explicam o que um projeto faz na maioria dos casos.

Um segundo trabalho[10] é mais parecido com o nosso, no sentido que o *corpus* deles é composto por documentos de elicitação como reportes e atas. Assim, foram utilizadas algumas das suas estratégias, considerando também o

PUC-Rio - Certificação Digital Nº 1322105/CA

ruído presente em alguns textos, o que podem sobrepor abstrações importantes quando se realiza a mineração. Neste trabalho[10], é utilizado um *corpus* de documentos para um domínio especifico (*Air Traffic Control*), em contrapartida, nossa proposta tem o desafio de identificar os domínios ao qual pertencem os documentos de GitHub, isso foi revelado no caso do "*Real Estate*" utilizado em dois domínios. Nesse sentido, outro trabalho de Stone & Sawyer[52] aborda a desambiguação usando técnicas de LSA (*Latent Semantic Analysis*) para o tratamento de palavras polissêmicas, poderia ajudar na tentativa de desambiguar documentos.

## 6.3. Contribuições

A principal contribuição desta pesquisa é o reuso de abordagens para abstrair informações relevantes, existentes na literatura da ER; como também a utilização de técnicas de NLP, que pertencem a outras áreas de pesquisa. A mistura de técnicas e abordagens, podem ajuda a tornar o GitHub, em uma fonte alvo para reuso de informações que apojem as tarefas de elicitação de requisitos. O processo proposto nesta pesquisa, pode também servir de guia para a mineração de outros artefatos de GitHub os quais possuam linguagem natural, como são as *issues* e *comments* entre outros.

Na primeira atividade *Collate* do processo proposto, teve como objetivo coletar documentos segundo a sua função comunicativa, no caso deste trabalho, os readmes. A automatização de esta tarefa é importante pois permite começar rapidamente o analise qualitativo ou quantitativo sobre os dados, a ferramenta criada[58] permite acessibilidade e disponibilidade de tais dados para qualquer tipo de pessoa que deseje fazer análise de *readmes* de projetos de GitHub dado uma query.

Na segunda atividade *Distill* do processo proposto, o objetivo é filtrar os textos *readmes* de maneira que a extração de IRR (Informações Relacionadas a Requisitos) seja sobre um *corpus* mais preciso. Se estabeleceram mecanismos baseados em duas abordagens, Filtragem por *POS-tagging* útil para filtrar informações latentes, e que podem estar altamente relacionadas ao domínio, e Filtragem por *Boilerplates* que é uma abordagem independente do domínio. Esses mecanismos são importantes, pois permitem automatizar a anotação manual de padrões relevantes nos readmes, o qual é inviável para centenas de readmes.

Na terceira atividade *Collocate* do processo proposto, o objetivo e organizar, e categorizar as informações filtradas que deem maior precisão na acessibilidade das informações. Isto significa que, a apresentação de termos latentes em projetos, sejam de tipo substantivo, substantivos próprios ou verbos, impacta na forma em que os projetos serão agrupados e ranqueados, como também na extração das IRR ancoradas pelas duas abordagens do *Distill*. É importante refinar esta atividade do processo, dado que a pletora de dados de projetos de GitHub, podem ser expostos desde diversos aspectos. Assim, ao decorrer da pesquisa, e com o estudo de caso para avaliar os textos, se criaram mapas conceituais para uma futura pesquisa na área de visualização de dados.

PUC-Rio - Certificação Digital Nº 1322105/CA

## 6.4. Trabalhos futuros

No decorrer de esta pesquisa, desde a definição de desafios (Capitulo 1) até o caso de estudo para validar as informações mineradas (Capitulo 5) vislumbramos a automatização de algumas tarefas do processo proposto, as quais ainda não foram abordadas pela limitação do tempo, e pelas técnicas fortemente relacionadas a outras áreas.

Dentre as possibilidades de reuso de informação existentes no GitHub, temos a exploração de outros artefatos importantes que possuem IRR tais como as *issues* e seus *comments*. Outra possibilidade, é a exploração de textos estruturados como o código fonte de projetos, para filtrar seus comentários, e seus *commits*.

Quanto a ferramenta para coletar *readmes* automaticamente, pensamos em reusa-lo para a coleta automatizada de *issues*. Outra evolução possível para o caso dos *readmes,* é dar a opção a dois tipos de consultas: a convencional ("real estate"), e outra que é focada apenas nos *readmes* (in:readme "real estate"). A primeira recupera projetos segundo a relevância do GitHub, a segunda recupera qualquer *readme* que possua as palavras "real" e "estate", tal consulta apesar de trazer projetos com menor relevância, pode levar a encontrar outro tipo de projetos, que pertencem a domínios relacionados ao domínio objetivo.

Em relação ao processo, e dentre as possibilidades de automatização, temos a atividade para *clusterizar* projetos de GitHub segundo ao domínio que pertencem (Distill). Com isto, a mineração sobre cada cluster pode resultar em IRR com maior precisão. Outra evolução importante é o mecanismo *tokenize* (Figura 9) para a extração das IRR em contexto de maneira automatizada.

Outra possibilidade de trabalho futuro é a evolução de abordagens para filtrar IRR, a literatura na área de ER vem utilizando técnicas como o LDA para encontrar tópicos num corpus de documentos, como também a técnica LSA que é utilizado para o tratamento de palavras polissêmicas. Esse tipo de palavras foi percebido no caso de estudo do *Real Estate*, a qual uma palavra composta que é utilizada em dois domínios. O LSA seria útil para desambiguar os domínios.

Em relação à avaliação das IRR (Informações Relacionadas a Requisitos), vislumbramos a evolução do trabalho com o estudo de novos casos, especialmente em projetos de tipo Startup para explorar como a disponibilização de tais IRR pode melhorar a criatividade, e, consequentemente a inovação de tais projetos.

Finalmente, relacionado a atividade *Collocate*, surge como trabalho futuro a evolução da apresentação visual das informações, de maneira que o leitor consiga filtrar efetivamente os projetos de interesse.

PUC-Rio - Certificação Digital Nº 1322105/CA

# Referências Bibliográficas


1 LEITE, J. C. S. P. **Livro Vivo: Engenharia de Requisitos**. 1994. Disponível em:http://livrodeengenhariaderequisitos.googlepages.com/ERNOTASDEAULA.pdf. Último acesso: 04-04-2016.

2 PATERNOSTER, N., GIARDINO, C., UNTERKALMSTEINER, M., GORSCHEK, T., & ABRAHAMSSON, P. **Software development in startup companies**: A systematic mapping study. Information and Software Technology, 2014. 56(10), pp. 1200-1218.

3 POTTS, C. **Invented requirements and imagined customers**: requirements engineering for off-the-shelf software. In Requirements Engineering, Proceedings of the Second IEEE International Symposium, 1995. pp. 128-130.

4 HO NGUYEN AND MCGUIRE, EUGENE. **A Case Study on the Software Culture at a Rank Startup**: An Argument for Adopting a Mature, Corporate, and Process-Driven Mentality. AMCIS Proceedings, 1998. Paper 292.

5 RIES, E. **The Lean Startup Methodology,** 2012. Disponível em: http://theleanstartup. com/principles (1.11. 2012). Último acesso: 04-04-2016.

6 SANTALA, JAAKKO ALEKSI. **The Scalable Startup: Customer, Business and Software**. Dissertação de Mestrado. Universidade de Helsinki, Finlândia. 2013.

7 METH, H. **A Design Theory for Requirements Mining Systems**. Tese de Doutorado. Universität Mannheim. Faculty of Business Administration (Business Computer Science). Mannheim, Germany, 2013.

8 VLAS RE, ROBINSON WN. **Two rule-based natural language strategies for requirements discovery and classification in open source software development projects**. Journal of Management Information Systems, 2012. 28(4):11-38.

9 LEITE, J. C. S., DE MORAES, E. A., & DE CASTRO, C. E. P. S. **A Strategy for Information Source Identification**. In WER, 2007. pp. 25-34.

10 SAWYER, P., RAYSON, P., & COSH, K. **Shallow knowledge as an aid to deep understanding in early phase requirements engineering**. Software Engineering, IEEE Transactions, 2005. 31(11), pp.969-981.

11 MAAREK YS, BERRY DM, KAISER GE. **An information retrieval approach for automatically constructing software libraries**. IEEE Transactions on software Engineering, 1991. 17(8):800-13.

12 MAHMOUD A, CARVER D. **Exploiting online human knowledge in Requirements Engineering.** IEEE 23rd International Requirements Engineering Conference (RE), 2015. pp. 262-267).

13 VENTURA, M. M. **O estudo de caso como modalidade de pesquisa**. Rev Socerj, 2007. 20(5), 383-386.

14 GOGUEN JA, LINDE C. **Techniques for requirements elicitation**. RE, 1993. 93:152-64.

15 ZAVE, P., & JACKSON, M. **Four dark corners of requirements engineering**. ACM transactions on Software Engineering and Methodology TOSEM, 1997. 6(1), pp.1-30.

16 ZOWGHI, D., & COULIN, C. **Requirements elicitation**: A survey of techniques, approaches, and tools. In Engineering and managing software requirements. Springer Berlin Heidelberg, 2005. pp. 19-46.


PUC-Rio - Certificação Digital Nº 1322105/CA

17 LEITE, J. C. S. P. **Baú de Fatos**. 04/2011. Em Livro Vivo: Engenharia de Requisitos. http://livrodeengenhariaderequisitos.blogspot.com/, 2007. Último acesso: 04-04-2016.

18 LEITE, J.C.S.P., FRANCO, A**. 0 Uso de Hipertexto na Elicitação de Linguagens da Aplicação**. Anais de IV Simpósio Brasileiro de Engenharia de Software, SBC, 1990. pp. 134-149.

19 METZ, C. **Triple Play:** GitHub code now lives in three places at once. https://www.wired.com/2016/04/github-now-three-places-keep-code-connected/ Wired, 2016. Último acesso: 05-05-2016.

20 CABOT, J., CÁNOVAS IZQUIERDO, J. L., COSENTINO, V., & ROLANDI, B. **Exploring the Use of Labels to Categorize Issues in Open-Source Software Projects**. In Proceedings of the 22nd International Conference on Software Analysis, Evolution, and Reengineering. SANER, 2015. pp. 479–483.

21 CÁNOVAS IZQUIERDO, J. L., COSENTINO, V., ROLANDI, B., BERGEL, A., & CABOT, J. **GiLA**: GitHub Label Analyzer. In Proceedings of the 22nd International Conference on Software Analysis, Evolution, and Reengineering SANER, 2015. pp. 550–554.

22 YU, Y., WANG, H., YIN, G., & LING, C. X. **Reviewer Recommender of Pull-Requests in GitHub**. In Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution, 2014. pp. 609–612.

23 GOUSIOS, G. **The GHTorent dataset and tool suite**. In Proceedings of the 10th Working Conference on Mining Software Repositories, IEEE, 2013. pp. 233-236.

24 SALO, R. **A guideline for requirements management in GitHub with lean approach.** Dissertação de Mestrado. University of Tampere. School of Information Sciences (Computer Science). Tampere, Finland, 2014.

25 RODRIGUEZ, MARKO. **Big Graph Data on Hortonworks Data Platform**, 2012 Disponível em: http://br.hortonworks.com/blog/big-graph-data-on-hortonworks-data-platform/. Último acesso: 4-11-2016.

26 HOTHO, A., NÜRNBERGER, A., & PAAß, G. **A Brief Survey of Text Mining**. In Ldv Forum, 2005. Vol. 20, No. 1, pp. 19-62.

27 MASSEY, A. K., EISENSTEIN, J., ANTÓN, A. I., & SWIRE, P. P. **Automated text mining for requirements analysis of policy documents**. In Requirements Engineering Conference (RE), 21st IEEE International, 2013. pp. 4-13.

28 KAIYA, H., & SAEKI, M. **Using domain ontology as domain knowledge for requirements elicitation**. In Requirements Engineering, 14th IEEE International Conference, 2006. pp. 189-198.

29 GERVASI, V., & NUSEIBEH, B. **Lightweight validation of natural language requirements: a case study**. In Requirements Engineering. Proceedings 4th IEEE International Conference, 2000. pp. 140-148.

PUC-Rio - Certificação Digital Nº 1322105/CA

30 KOF, L. **Natural language processing: mature enough for requirements documents analysis?**. In Natural language processing and information systems. Springer Berlin Heidelberg, 2005. pp. 91-102.

31 MC KEVITT, P., PARTRIDGE, D., & WILKS, Y. **Approaches to natural language discourse processing**. Artificial Intelligence Review, 1992. 6(4), pp. 333-364.

32 LUHN, H. P. **Key word-in-context index for technical literature (kwic index)**. American Documentation, 1960. 11(4), 288-295.

33 VOUTILAINEN, A. **Part-of-speech tagging**. The Oxford handbook of computational linguistics, 2003. pp. 219-232.

34 MANNING, C. D., RAGHAVAN, P & SCHÜTZE, H. **Introduction to Information Retrieval,** Cambridge University Press, 2008. Disponível em: http://www-nlp.stanford.edu/IR-book/ Último acesso: 04-04-2016.

35 PISKORSKI, J., & YANGARBER, R. **Information extraction: Past, present and future.** In Multi-source, multilingual information extraction and summarization. Springer Berlin Heidelberg, 2013. pp. 23-49.

36 GOLDIN, L., & BERRY, D. M. **Reuse of requirements reduced time to market at one industrial shop: a case study**. Requirements Engineering, 2015. 20(1), pp. 23-44.

37 KUPIEC, J., PEDERSEN, J., & CHEN, F. **A trainable document summarizer**. In Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, 1995. pp. 68-73

38 NECHES R., FIKES R., FININ T., GRUBER T, PATIL R., SENATOR T, SWARTOU W. **Enabling Technology for Knowledge Sharing AI** MAGAZINE, 1991. Vol 12, No 3.

39 FREEMAN, P. **Reusable software engineering:** Concepts and research directions. ITT Proceedings of the Workshop on Reusability in Programming, 1983. pp .129-137.

40 EKAPUTRA, F. J., SERRAL, E., & BIFFL, S. **Building an empirical software engineering research knowledge base from heterogeneous data sources**. In Proceedings of the 14th International Conference on Knowledge Technologies and Data-driven Business, ACM, 2014. p. 13.

41 MARKUS, L. **Toward a Theory of Knowledge Reuse:** Types of Knowledge Reuse Situations and Factors in Reuse Success. Journal of Management Information Systems, 2001. Volume 18, Issue 1,

42 PORTUGAL, R., KINDER, E., & LEITE, J. C. S. P. **Visualizer:** A tool for disclosure of XML scenarios content, 2014. Disponível em: https://github.com/nitanilla/XML-visualization. Acessado em 11/3/2015.

43 ENGIEL P, PIVATELLI J, PORTUGAL R., & LEITE, J. C. S. P **Representando o Conceito de Transparência Através de um Léxico**

PUC-Rio - Certificação Digital Nº 1322105/CA

**Ampliado da Linguagem. Conference:** 18th Workshop on Requirements Engineering WER, 2015.

44 BRETT MR. **Topic modeling**: a basic introduction. Journal of Digital Humanities, 2012. 12; 2(1).

45 THADA, V., & JAGLAN, V. **Web Information Retrieval,** International Journal of Computer Applications, 2013. 76(1).

46 GOLDIN, L., & BERRY, D. M. **AbstFinder, a prototype natural language text abstraction finder for use in requirements elicitation**. Automated Software Engineering, 1997. 4(4), pp. 375-412.

47 GOUSIOS, G., PINZGER, M., & DEURSEN, A. V. **An exploratory study of the pull-based software development model.** In Proceedings of the 36th International Conference on Software Engineering, ACM, 2014. pp. 345-355.

48 DUCHENEAUT, N. **Socialization in an open source software community: A socio-technical analysis**. Computer Supported Cooperative Work (CSCW), 2005. 14(4), pp. 323-368.

49 MARLOW, J., DABBISH, L., & HERBSLEB, J. **Impression formation in online peer production: activity traces and personal profiles in github**. In Proceedings of the 2013 conference on Computer supported cooperative work. ACM, 2013. pp. 117-128.

50 ARORA, C., SABETZADEH, M., BRIAND, L. C., & ZIMMER, F. **Requirement boilerplates:** Transition from manually-enforced to automatically-verifiable natural language patterns. In Requirements Patterns (RePa), IEEE 4th International Workshop, 2014. pp. 1-8.

51 PORTUGAL, R. L. Q., DO PRADO LEITE, J. C. S., & ALMENTERO, E. **Time-constrained requirements elicitation:** reusing GitHub content. In Just-In-Time Requirements Engineering (JITRE), IEEE Workshop, 2015. pp. 5-8.

52 STONE, A., & SAWYER, P. **Identifying tacit knowledge-based requirements**. In Software, IEE Proceedings, 2006. Vol. 153, No. 6, pp. 211-218.

53 FEINERER, K. HORNIK, & D. MEYER. **Text mining infrastructure in R. Journal of Statistical Software**, 2008. 25(5): pp. 1–54.

54 OSBORNE, M., & MACNISH, C. K. **Processing natural language software requirement specifications. In Requirements Engineering**. Proceedings of the Second International Conference, 1996. pp. 229-236.

55 CLELAND-HUANG, J., SETTIMI, R., ZOU, X., & SOLC, P. **Automated classification of non-functional requirements**. Requirements Engineering, (2007)12(2), 103-120.

56 ROSS, D. T. **Structured analysis (SA):** A language for communicating ideas. Software Engineering, IEEE Transactions, 1977. (1) pp. 16-34.

57 KENT, A., BERRY, M. M., LUEHRS, F. U. & PERRY, J. W. **Machine literature searching VIII. Operational criteria for designing information retrieval systems**. Amer. Doc., 1955. 6: pp. 93–101.

PUC-Rio - Certificação Digital Nº 1322105/CA

58 PORTUGAL, R. L. Q., HUGO ROQUE AND DO PRADO LEITE, J. C. S. **A Corpus Builder:** Retrieving Raw Data from GitHub for Knowledge Reuse In Requirements Elicitation." 3rd Annual International Symposium on Information Management and Big Data, 2016.

59 KUMARAN, G., & ALLAN, J. **Information retrieval techniques for templated queries**. In Large Scale Semantic Access to Content (Text, Image, Video, and Sound), Le Centre De Hautes Etudes Internationales D'informatique Documentaire, 2007, May. pp. 671-686.

60 BERRY, M. W. Survey of text mining. Computing Reviews, 2004. 45(9), 548.

61 STEINBACH, M., KARYPIS, G., & KUMAR, V. **A comparison of document clustering techniques**. In KDD workshop on text mining, 2000. Vol. 400, No. 1, pp. 525-526.

62 VENKATARAMANI, R., ASADULLAH, A., BHAT, V., & MUDDU, B.. **Latent Co-development Analysis Based Semantic Search for Large Code Repositories.** In IEEE International Conference on Software Maintenance, 2013. pp. 372-375.

63 SINCLAIR, J. **Developing linguistic corpora**: a guide to good practice. Corpus and text–basic principles, 2004.

64 LICHMAN, M. **UCI Machine Learning Repository**. [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2013.

65 RIDAO, M., DOORN, J., & LEITE, J. C. S. D. P. Domain independent regularities in scenarios. In Requirements Engineering. Proceedings. Fifth IEEE International Symposium, 2001. pp. 120-127.

66 LECOEUCHE, R. **Finding comparatively important concepts between texts.** In Automated Software Engineering. Proceedings ASE. The Fifteenth IEEE International Conference, 2000. pp. 55-60.

67 ROBSON, C., & MCCARTAN, K. Real world research. John Wiley & Sons. Chicago, 2016.

68 BERRY, D., GACITUA, R., SAWYER, P., & TJONG, S. F. **The case for dumb requirements engineering tools.** In Requirements Engineering: Foundation for Software Quality. Springer Berlin Heidelberg, 2012. pp. 211-217.

69 PORTUGAL, R. L. Q., & DO PRADO LEITE, J. C. S. **Extracting Requirements Patterns from Software Repositories**. Em Requirements Patterns (RePa), IEEE Workshop in Requirements Engineering Conference, 2016.

PUC-Rio - Certificação Digital Nº 1322105/CA

# Glossário

**A**

**API**: Interface de Programação de Aplicações, é um conjunto de definições de sub-rotinas, protocolos e ferramentas para a construção de software e aplicações.

**B**

***Bag of Words:*** é uma representação simplificada utilizada para o processamento de linguagem natural e recuperação de informação (IR). Nesse modelo, um texto (como uma frase ou um documento), é representado como o saco (multiset) das suas palavras, ignorando a gramática e até mesmo a ordem das palavras, mas mantendo multiplicidade.

**C**

**Consumidor Final:** Pessoa ou organização com uma variedade de necessidades, que exige bens ou serviços em troca de dinheiro.

***Corpus of texts***: Uma coleção de textos escritos, especialmente obras de um determinado autor ou um corpo de escritos sobre um determinado assunto.

***Curl Library:*** Uma biblioteca (usado desde várias linguagens de programação) para transferências URL do lado do cliente.

**D**

***Dataset***: É um conjunto de dados, geralmente apresentados em forma tabular. Cada coluna representa uma variável em particular. Cada linha corresponde a um determinado membro do conjunto de dados em questão.

***Dumps***: Uma cópia dos dados armazenados para um local diferente, realizada tipicamente como uma proteção contra a perda.

PUC-Rio - Certificação Digital Nº 1322105/CA

**F**

***Forks*:** É uma cópia de um repositório em GitHub. *Forking* um repositório permite que se experimente livremente com alterações sem afetar o projeto original.

**G**

**Gem:** programas de Ruby e bibliotecas em um formato de autocontido chamado de "gem".

***Git:*** *Sistema livre e de código aberto distribuído de controle de versão de projetos com rapidez e eficiência.*

***GitHub:*** *Serviço Web de hospedagem de projetos open source. Oferece as funcionalidades do Git.*

**I**

***Issues***: São uma maneira de manter o controle de tarefas, melhoras, e erros de projetos de software.

***Issues comments***: Permite fornecer feedback sobre um issue a qualquer pessoa com acesso ao projeto.

**Investidor anjo**: É uma pessoa física que faz investimentos com seu próprio capital em empresas que estão nascendo e que apresentam alto potencial de crescimento, como as Startups.

**IRR**: Informação relacionada a requisito. Fatos com estereotipo de funcionalidades.

**L**

***Logfile*:** É um arquivo que registra os eventos que ocorrem em um sistema operacional, ou mensagens entre diferentes usuários de um software de comunicação.

PUC-Rio - Certificação Digital Nº 1322105/CA

**LDA:** é um modelo estatístico generativo que Permite a conjuntos de observações a ser explicado por grupos não observadas que explicam por que algumas partes dos dados são semelhantes.

**LSA**: é uma técnica de processamento de linguagem natural, em particular, a semântica de distribuição, de analisar um conjunto de relações entre os documentos e os termos que eles contêm, produzindo um conjunto de conceitos relacionados com os documentos e termos.

**M**

***Markdown***: é uma linguagem simples de marcação desenhado de modo para facilitar sua conversão em HTML.

***Marketable feature*:** Funcionalidade que tem um “valor comercial” intrínseco. É aplicado a uma unidade funcional ao invés de um produto como um todo.

**P**

**Perfil de consumidor:** Conjunto de características que descrevem ao cliente objetivo.

***Personal Access Token*:** Atua como um intermediário, oferecendo o serviço com um *token* de acesso que autoriza informações de conta específica para ser compartilhada.

**Produto Mínimo Viável**: A versão mais simples de um produto, lançado com uma quantidade mínima de tempo e esforço de desenvolvimento, a fim de testar a hipótese de um produto.

**Q**

***Query***: uma pergunta, ou um pedido de informação sobre alguma coisa, a um motor de busca ou banco de dados.

PUC-Rio - Certificação Digital Nº 1322105/CA

**R**

***Readme***: Documento que contem informação útil sobre um programa de software, é usualmente faz parte de sua documentação.

***Real Estate***: O negócio de imóveis; a profissão de comprar, vender ou alugar terrenos, edifícios ou habitações.

**S**

***Stars*:** Em GitHub, expressa o favoritismo por um projeto.

***Startup****: Um empreendimento empresarial, projetado para procurar um modelo de negócios repetível e escalável.*

***Summarization****: Sumarização de conceitos importantes em um texto. Normalmente, estes são termos de alta frequência*

**T**

***TF-IDF:*** Na recuperação de informação, tf-idf é uma estatística numérica que pretende refletir o quão importante uma palavra é um documento em uma coleção ou corpus.

***Time to market***: É a quantidade de tempo que toma a um produto desde sua concepção até a disponibilidade para a venda.

***Topic Modelling:*** É um tipo de modelo estatístico para descobrir os "tópicos" que ocorrem em uma coleção de documentos.

**U**

***URL:*** é uma sigla para *Uniform Resource Locator* e é uma referência (um endereço) para um recurso na Internet.

PUC-Rio - Certificação Digital Nº 1322105/CA

**W**

***Weka***: é uma ferramenta com um conjunto de algoritmos de aprendizado de máquina para tarefas de mineração de dados. http://www.cs.waikato.ac.nz/ml/weka

PUC-Rio - Certificação Digital Nº 1322105/CA

PUC-Rio - Certificação Digital Nº 1322105/CA

# Anexo 1. Caso de estudo “music application”

## A1.1. IRR para o piloto do caso de estudo

| Number.-.User.-.Projeto.-.ReadmeExtension | Requirement Related Information (IRR) | Already developed | Which version was developed | Being built for the current release | A good idea to take into consideration | Not interesting for now |
|---|---|---|---|---|---|---|
| 0026.-.lateralblast.-.iclean.-.md | This is a ruby script that can be used to clean up iTunes files/directories. | | | | | 1 |
| 0104.-.dheeratp.-.iolabproject4.-.md | In this panel we will look at how visualizations can be used to help people explore the music space and discover new, interesting music that they will like. | | | | 1 | |
| 0226.-.ferguson.-.otto.-.md | The script can be used to scan a different folder than was initially scanned when you started Otto. | | | | | 1 |
| 0338.-.joegoodall1.-.Spotify-Streamer.-.md | Can be used to search for artists which are displayed in a list. | | | | 1 | |

PUC-Rio - Certificação Digital Nº 1322105/CA

| Number.-.User.-.Projeto.-.ReadmeExtension | Requirement Related Information (IRR) | Already developed | Which version was developed | Being built for the current release | A good idea to take into consideration | Not interesting for now |
|---|---|---|---|---|---|---|
| 0570.-.alseambusher.-.android-manager.-.md | 1. This is a desktop application which can be used to sync your android device with your computer. | | | | | 1 |
| 0619.-.twillis.-.spazzer.-.txt | There’s also a paster command that can be used to periodically scan your collection and scrape the metadata. currently only looks for mp3 and flac, but can easily be extended to support whatever mutagen supports. | | | | 1 | |
| 0992.-.petewms1.-.raven.-.rdoc | Helpers can be used to wrap functionality for your views into methods. | | | | | 1 |
| 1019.-.ellisonleao.-.magictools.-.md | GameMaker accommodates the creation of cross-platform video games using drag and drop or a scripting language known as Game Maker Language, which can be used to develop more advanced games that could not be created just by using the drag and drop features. | | | | | 1 |
| 1045.-.nicklockwood.-.SoundManager.-.md | he left/right stereo pan of the file. Value ranges from -1.0 to 1.0 and can be used to shift the location of the sound in space. Has no effect on Mac OS 10.6. | | | | 1 | |

PUC-Rio - Certificação Digital Nº 1322105/CA

| Number.-.User.-.Projeto.-.ReadmeExtension | Requirement Related Information (IRR) | Already developed | Which version was developed | Being built for the current release | A good idea to take into consideration | Not interesting for now |
|---|---|---|---|---|---|---|
| 1123.-.scottfrederick.-.spring-music.-.md | Depending on the Cloud Foundry service provider, persistence services might be offered and managed by the platform. These steps can be used to create and bind a service that is managed by the platform: | | | | | 1 |
| 1141.-.drduh.-.OS-X-Security-and-Privacy-Guide.-.md | [facebook/osquery](https://github.com/facebook/osquery) can be used to retrieve low level system information. Users can write SQL queries to retrieve system information. | | | | | 1 |
| 1143.-.samsquire.-.ideas.-.md | Sometimes we want to use X as if it is a Y. Principles in one area can be used to tackle something in a completely different area using a non-traditional style. | | | | | 1 |
| 1146.-.Kickball.-.awesome-selfhosted.-.md | * [phpBB](https://www.phpbb.com/) - flat-forum bulletin board software solution that can be used to stay in touch with a group of people or can power your entire website. `GPLv2` `PHP` | | | | | 1 |
| 1150.-.migueldeicaza.-.MonoTouch.Dialog.-.markdown | The message element can be used to render rows that render message-like information, that includes a sender, a subject, a time, a greyed out snippet and a couple of status features like read/unread or the message count: | | | | | 1 |

PUC-Rio - Certificação Digital Nº 1322105/CA

| Number.-.User.-.Projeto.-.ReadmeExtension | Requirement Related Information (IRR) | Already developed | Which version was developed | Being built for the current release | A good idea to take into consideration | Not interesting for now |
|---|---|---|---|---|---|---|
| 1150.-.migueldeicaza.-.MonoTouch.Dialog.-.markdown | MonoTouch.Dialog now incorporates TweetStation's image loader. This image loader can be used to load images in the background, supports caching and can notify your code when the image has been loaded. | | | | | 1 |
| 1150.-.migueldeicaza.-.MonoTouch.Dialog.-.markdown | The "group" property can be used to tag a boolean element as belonging to a particular group. This is useful if the containing root also has a "group" property as the root will summarize the results with a count of all the booleans (or checkboxes) that belong to the same group. | | | | | 1 |
| 1155.-.redecentralize.-.alternative-internet.-.md | \|Netsukuku\|[Netsukuku](http://netsukuku.freaknet.org) is an ad-hoc network system designed to handle massive numbers of nodes with minimal consumption of CPU and memory resources. It can be used to build a world-wide distributed, fault-tolerant, anonymous, and censorship-immune network, fully independent from the Internet.\|Python\|6 K\|169 K\|14 \|11 year(s)\| | | | | | 1 |
| 1155.-.redecentralize.-.alternative-internet.-.md | \|PPNet\|[PPNet](https://github.com/pixelpark/ppnet) is a middleware that can be used to create a social network, either temporarily or permanently for a group of users. Includes mobile client for Android.\|-\|-\|-\|-\|-\| | | | | | 1 |

PUC-Rio - Certificação Digital Nº 1322105/CA

| Number.-.User.-.Projeto.-.ReadmeExtension | Requirement Related Information (IRR) | Already developed | Which version was developed | Being built for the current release | A good idea to take into consideration | Not interesting for now |
|---|---|---|---|---|---|---|
| 1156.-.scummvm.-.scummvm.-. | There is no in-game option to toggle speech and text. Only ScummVM's audio options can be used to toggle this feature. Note that some spoken dialog is missing from the game texts. | | | | | 1 |
| 1156.-.scummvm.-.scummvm.-. | Note for Windows NT4/2000/XP/Vista/7 users: The default saved games location changed in ScummVM 1.5.0. The migration batch file can be used to copy saved games from the old default location, to the new default location. | | | | | 1 |
| 1161.-.alebcay.-.awesome-shell.-.md | [Dropbox-Uploader](https://github.com/andreafabrizi/Dropbox-Uploader) - Dropbox Uploader is a Bash script which can be used to upload, download, list or delete files from Dropbox | | | | | 1 |
| 1164.-.karan.-.Projects-Solutions.-.md | **Web Browser with Tabs** - Create a small web browser that allows you to navigate the web and contains tabs which can be used to navigate to multiple web pages at once. For simplicity don’t worry about executing Javascript or other client side code. | | | | | 1 |
| 1167.-.leereilly.-.games.-.md | * [ioquake3](https://github.com/ioquake/ioq3) - The free software FPS engine that can be used to play Quake 3, or make your own game. | | | | | 1 |

PUC-Rio - Certificação Digital Nº 1322105/CA

| Number.-.User.-.Projeto.-.ReadmeExtension | Requirement Related Information (IRR) | Already developed | Which version was developed | Being built for the current release | A good idea to take into consideration | Not interesting for now |
|---|---|---|---|---|---|---|
| 1170.-.marcel.-.aws-s3.-.rdoc | Astute readers, as they say, may have noticed that we used the third parameter to pass in the content type, rather than the fourth parameter as we had the last time we created an object. If the bucket can be inferred, or is explicitly set, as we've done in the JukeBoxSong class, then the third argument can be used to pass in options. | | | | | 1 |
| 1173.-.vsouza.-.awesome-ios.-.md | *[WobbleView](https://github.com/inFullMobile/WobbleView) - WobbleView is an implementation of a recently popular wobble effect for any view in your app. It can be used to easily add dynamics to user interactions and transitions. :large_orange_diamond: | | | | | 1 |
| 1173.-.vsouza.-.awesome-ios.-.md | * [Gradle](http://openbakery.org/gradle.html) - The gradle xcode plugin can be used to build iOS or Mac OS X Projects using gradle. | | | | | 1 |
| 1174.-.fffaraz.-.awesome-cpp.-.md | [Boost.PropertyTree](http://www.boost.org/doc/libs/1_55_0/doc/html/property_tree.html) - A property tree parser/generator that can be used to parse XML/JSON/INI/Info files. [Boost] | | | | | 1 |

PUC-Rio - Certificação Digital Nº 1322105/CA

| Number.-.User.-.Projeto.-.ReadmeExtension | Requirement Related Information (IRR) | Already developed | Which version was developed | Being built for the current release | A good idea to take into consideration | Not interesting for now |
|---|---|---|---|---|---|---|
| 1174.-.fffaraz.-.awesome-cpp.-.md | [Boost.PropertyTree](http://www.boost.org/doc/libs/1_55_0/doc/html/property_tree.html) - A property tree parser/generator that can be used to parse XML/JSON/INI/Info files. [Boost]1177.-.prakhar1989.-.awesome-courses.-.md | | | | | 1 |
| 1177.-.prakhar1989.-.awesome-courses.-.md | Computation tools can be used to answer seemingly straightforward questions about a single patient's test results (“Does this patient have a normal heart rhythm?”), or to address vital questions about large populations (“Is there any clinical condition that affects the risks of Alzheimer”). | | | | | 1 |
| 1179.-.avelino.-.awesome-go.-.md | * [gosms](https://github.com/haxpax/gosms) - Your own local SMS gateway in Go that can be used to send SMS | | | | | 1 |
| 1184.-.jgm.-.pandoc.-. | Include contents of *FILE*, verbatim, at the beginning of the document body (e.g. after the `<body>` tag in HTML, or the `\begin{document}` command in LaTeX). This can be used to include navigation bars or banners in HTML documents. This option can be used repeatedly to include multiple files. They will be included in the order specified. Implies `--standalone`. | | | | | 1 |

PUC-Rio - Certificação Digital Nº 1322105/CA

| Number.-.User.-.Projeto.-.ReadmeExtension | Requirement Related Information (IRR) | Already developed | Which version was developed | Being built for the current release | A good idea to take into consideration | Not interesting for now |
|---|---|---|---|---|---|---|
| 1184.-.jgm.-.pandoc.-. | Native pandoc `span`s and `div`s with the lang attribute (value in BCP 47) can be used to switch the language in that range. | | | | | 1 |
| 1184.-.jgm.-.pandoc.-. | For bidirectional documents, native pandoc `span`s and `div`s with the `dir` attribute (value `rtl` or `ltr`) can be used to override the base direction in some output formats.<br>This may not always be necessary if the final renderer (e.g. the browser, when generating HTML) supports the [Unicode Bidirectional Algorithm]. | | | | | 1 |
| 1184.-.jgm.-.pandoc.-. | A dot can be used to select a field of a variable that takes an object as its value. So, for example: | | | | | 1 |
| 1184.-.jgm.-.pandoc.-. | Note that YAML escaping rules must be followed. Thus, for example, if a title contains a colon, it must be quoted. The pipe character (`\|`) can be used to begin an indented block that will be interpreted literally, without need for escaping. | | | | | 1 |

PUC-Rio - Certificação Digital Nº 1322105/CA

| Number.-.User.-.Projeto.-.ReadmeExtension | Requirement Related Information (IRR) | Already developed | Which version was developed | Being built for the current release | A good idea to take into consideration | Not interesting for now |
|---|---|---|---|---|---|---|
| 1184.-.jgm.-.pandoc.-. | With all HTML slide formats, the `--self-contained` option can be used to produce a single file that contains all of the data necessary to display the slide show, including linked scripts, stylesheets, images, and videos. | | | | | 1 |
| 1195.-.eway2012.-.SampleCode.-.md | #StitchedStreamPlayer#<br>A simple AVFoundation demonstration of how timed metadata can be used to identify different content in a stream, supporting a custom seek UI.<br>[URL](https://developer.apple.com/library/ios/#samplecode/StitchedStreamPlayer/Introduction/Intro.html#//apple_ref/doc/uid/DTS40010092) | | | | 1 | |
| 1196.-.megous.-.megatools.-. | megastream Streaming download of a file (can be used to preview videos or music) | 1 | .1 | | | |
| 1199.-.robovm.-.apple-ios-samples.-.md | <sup>Audio & Video</sup> \| <sup>AVFoundation</sup> \| <sup>A simple AVFoundation demonstration of how timed metadata can be used to identify different content in a stream, supporting a custom seek UI.</sup> | | | | 1 | |
| 1199.-.robovm.-.apple-ios-samples.-.md | This sample demonstrates two methods that can be used to implement a custom accessory view in your UITableViewCell's. In both examples, a custom control that implements a toggle-able checkbox is used.</sup> \| | | | | | 1 |

## A1.2. IRR para o Primeiro Teste

Formatação do formulário apresentado para avaliação de frases.

Looking for Similar Projects on GitHub.

Hi, this form wants to show you some projects from GitHub that could be useful in your music application project.

Without compromise feel free to judge them, there are 10 phrases and the link from where they were extracted

In average this form takes 25 minutes. Thank you.


Looking for Similar Projects on GitHub.

*Obrigatório

Moment Music App

https://github.com/shaumik/MusicMachine

**Moment is a web application that stores all your special memories with the music you love. By tapping into Spotify's Web API, Moment allows users to bookmark music they enjoy in a journal format and navigate all their previous memories with music.** *

If comments, please write in the box "otro"

- ☐ Already developed
- ☐ Being built for the current release
- ☐ A good idea to take into consideration
- ☐ Not interesting for now
- ☐ I dont understand
- ☐ Out of Context
- ☐ Outro:

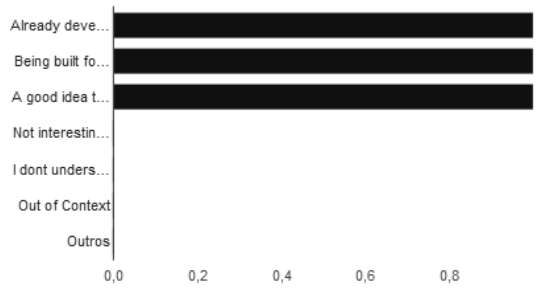
« Voltar Continuar »

PUC-Rio - Certificação Digital Nº 1322105/CA

**Moment Music App**

https://github.com/pstrum/Moment-Music-App

“Moment is a web application that stores all your special memories with the music you love. By tapping into Spotify's Web API, Moment **allows users** to bookmark music they enjoy in a journal format and navigate all their previous memories with music.”

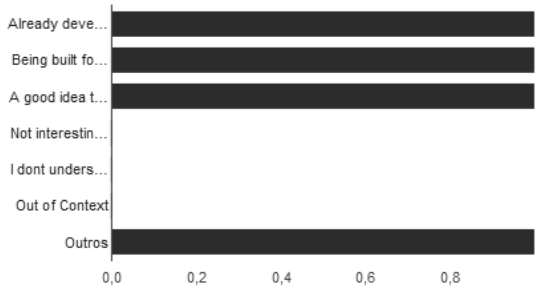



| | | |
|---|---|---|
| Already developed | 1 | 33.3% |
| Being built for the current release | 1 | 33.3% |
| A good idea to take into consideration | 1 | 33.3% |
| Not interesting for now | 0 | 0% |
| I dont understand | 0 | 0% |
| Out of Context | 0 | 0% |
| Outros | 0 | 0% |

**Cuore SoundShack vAndroid**

https://github.com/cuoretech/soundshack_android

“SoundShack Leverages the Power of Broadcom's Latest Wiced chip to Stream HD Audio to Wifi enabled android devices. This Android application **allows users** to synchronously stream music with no lag. User can also asynchronously manage speaker groupings, giving users the power to listen to the football game on their living room speakers and rock out to their patio speakers at the sametime”



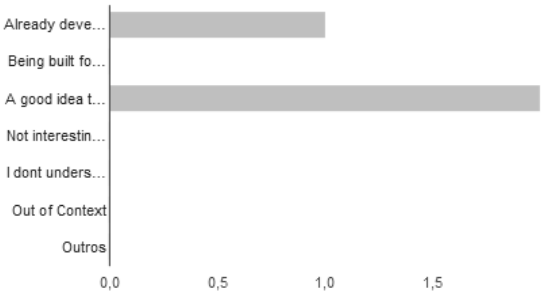

| | | |
|---|---|---|
| Already developed | 1 | 33.3% |
| Being built for the current release | 1 | 33.3% |
| A good idea to take into consideration | 1 | 33.3% |
| Not interesting for now | 0 | 0% |
| I dont understand | 0 | 0% |
| Out of Context | 0 | 0% |
| Outros | 1 | 33.3% |

Comentário da pessoa que respondeu “*Being built for the current release*”: *Tem alguma semelhança com o Hear Speaker*

**MusicMachine**

https://github.com/shaumik/MusicMachine

“Powerful web application that **allows users** to query music and instantly download that music. The project was done using nodejs and expressjs”



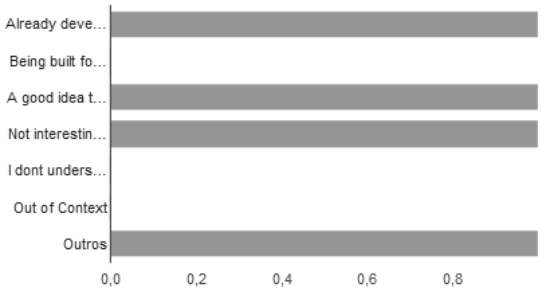

| | | |
|---|---|---|
| Already developed | 1 | 33.3% |
| Being built for the current release | 0 | 0% |
| A good idea to take into consideration | 2 | 66.7% |
| Not interesting for now | 0 | 0% |
| I dont understand | 0 | 0% |
| Out of Context | 0 | 0% |
| Outros | 0 | 0% |

PUC-Rio - Certificação Digital Nº 1322105/CA

### Av-Browser

https://github.com/x0xMaximus/AV-Browser

“AV-Browser is a web application that **allows users** to browse through music visually. I wanted to create a way to abstract the concept of listening to new artists to viewing them. Many people judge a book by its cover, I want to expose this action by allowing viewers to judge a music by its album art and artist's image”.



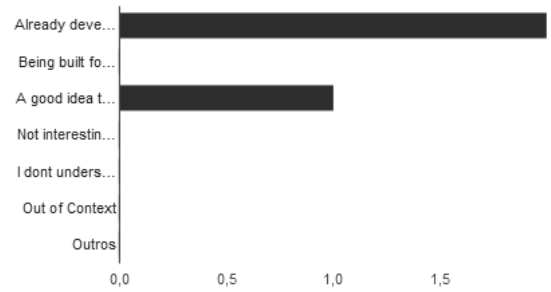

| | | |
|---|---|---|
| Already developed | 1 | 33.3% |
| Being built for the current release | 0 | 0% |
| A good idea to take into consideration | 1 | 33.3% |
| Not interesting for now | 1 | 33.3% |
| I dont understand | 0 | 0% |
| Out of Context | 0 | 0% |
| Outros | 1 | 33.3% |

Comentário da pessoa que respondeu “*A good idea to take into consideration*”: *Parece que o projeto relaciona capas de álbuns com wallpapers e outras imagens. Isso poderia se aplicar ao Hear relacionando os álbuns com o ambiente.*

### Music History (AngularJS)

https://github.com/tombler/music-history-angular

“This is an iteration of an individual project for Nashville Software School, Cohort 10, that **allows users** to search songs and compile a playlist using AngularJS”.

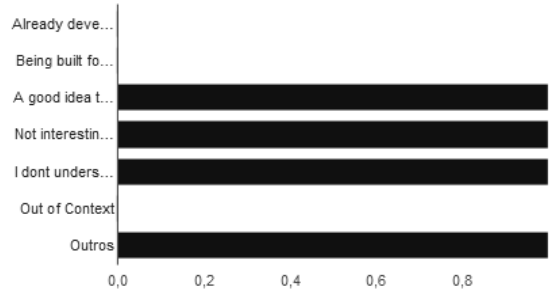



| | | |
|---|---|---|
| Already developed | 2 | 66.7% |
| Being built for the current release | 0 | 0% |
| A good idea to take into consideration | 1 | 33.3% |
| Not interesting for now | 0 | 0% |
| I dont understand | 0 | 0% |
| Out of Context | 0 | 0% |
| Outros | 0 | 0% |

### Mytunes

https://github.com/melindabernrdo/mytunes

Frase 6: “This is a project from the [Hack Reactor] (http://hackreactor.com) Curriculum that I completed as a student at [Makersquare](http://makersquare.com). This front-end application creates a music player in backbone.js **that allows** users to play Aliyah songs in queue.”




| | | |
|---|---|---|
| Already developed | 0 | 0% |
| Being built for the current release | 0 | 0% |
| A good idea to take into consideration | 1 | 33.3% |
| Not interesting for now | 1 | 33.3% |
| I dont understand | 1 | 33.3% |
| Out of Context | 0 | 0% |
| Outros | 1 | 33.3% |

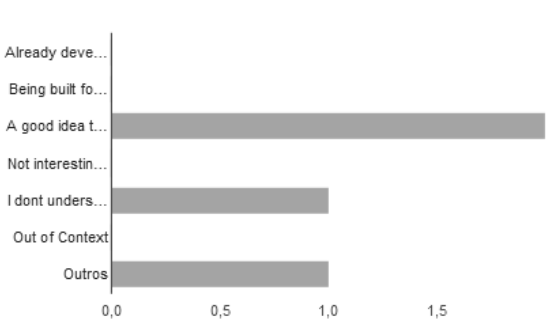
Comentário da pessoa que respondeu “A good idea to take into consideration”: *Pode ser usado como referência para uma versão web do Hear*

PUC-Rio - Certificação Digital Nº 1322105/CA

## Listen

https://github.com/artsmia/listen

**Listen!** is a sound remixing application that **allows users** to remix recorded music and sound interpretations, recorded by Minnesota musicians, of 21 objects in the MIA's collection.



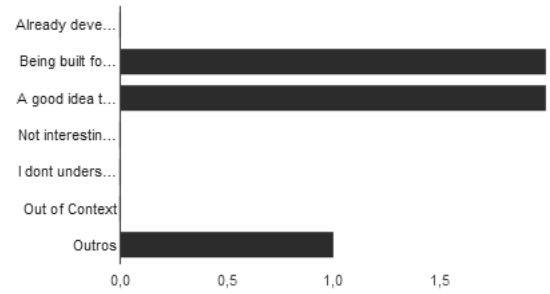

| | | |
|---|---|---|
| Already developed | 0 | 0% |
| Being built for the current release | 0 | 0% |
| A good idea to take into consideration | 2 | 66.7% |
| Not interesting for now | 0 | 0% |
| I dont understand | 1 | 33.3% |
| Out of Context | 0 | 0% |
| Outros | 1 | 33.3% |

Comentário da pessoa que respondeu “A good idea to take into consideration”: *Existem várias limitações das APIs do iTunes, Spotify e Deezer relacionadas a alteração das faixas (basicamente, não podemos fazer mixes), mas também temos planos de incluir faixas de músicos independentes (os mesmos podem fazer upload das suas músicas). E aí, com os devidos direitos, podemos fazer isso. :)*

## Vent [in progress]

https://github.com/bsatterwhite3/Vent

This android application **allows users** to listen to a music playlist tailored to them based on their mood. Their mood is extracted using sentiment analysis on diary entries in the app.




| | | |
|---|---|---|
| Already developed | 0 | 0% |
| Being built for the current release | 2 | 66.7% |
| A good idea to take into consideration | 2 | 66.7% |
| Not interesting for now | 0 | 0% |
| I dont understand | 0 | 0% |
| Out of Context | 0 | 0% |
| Outros | 1 | 33.3% |

Comentário da pessoa que respondeu “Being built for the current release”:
*Já estamos desenvolvendo um algoritmo pra entender o sentimento de cada pessoa com base nas últimas músicas que ela escutou. A Echonest API nos dá a informação dos moods de cada música. Uma outra forma de extrair essa informação seria analisar o que a pessoa escreve nas stories. Excelente referência, embora o projeto pareça ter sido descontinuado.*

PUC-Rio - Certificação Digital Nº 1322105/CA

## A1.3. IRR para o Segundo Teste

### SoundTrek
https://github.com/SiddharthManoj/SoundTrek
Application **allows users to** select a country in the world and listen to music from that country.
A good idea to take into consideration. Comentário:
Pelo que eu entendi, você pode ouvir músicas de outro país. O Hear terá a opção de você selecionar outros lugares para ouvir músicas que as pessoas desses lugares ouviram.

### ![listen interface animation]
https://github.com/artsmia/listen
**Listen!** is a sound remixing application that **allows users to** remix recorded music and sound interpretations, recorded by Minnesota musicians, of 21 objects in the MIA's collection.
Not interesting for now

### Crux
https://github.com/gmacster/crux
Crux is a web application that **allows users to** view an aggregate collection of sheet music for bagpipes.
Not interesting for now

### Cloud-Jam
https://github.com/ankurdh/Cloud-Jam
A mobile application which **allows users to** create a band and lets them share a piece of music they create.
A good idea to take into consideration. Comentário: Estamos considerando a possibilidade de apoiar artistas independentes, permitindo o envio de suas músicas para o app, mesmo que estas não estejam no iTunes.

### # Banana Music Station
https://github.com/shaohaolin/BananaMusicStation
A web application that allows users to listen the hottest music of the week. Banana Music Station **allows users to** create their own music lists, add and remove music to their lists, share comments of the music.
A good idea to take into consideration. Comentário: O projeto parece estar incompleto e tem muito tempo que não tem atualização, mas seria uma boa referência.

### Cuore SoundShack vAndroid
https://github.com/cuoretech/soundshack_android
SoundShack Leverages the Power of Broadcom's Latest Wiced chip to Stream HD Audio to Wifi enabled android devices. This Android application **allows users to** synchronously stream music with no lag. User can also asynchronously manage speaker groupings, giving users the power to listen to the football game on their living room speakers and rock out to their patio speakers at the sametime.
Out of Context

### Foobar2000 web UI
https://github.com/klemola/foobar2000-web-ui
A web application that **allows the user to** send basic Foobar2000 playback commands and adjust application volume level. Information about the track that is currently playing is also displayed and automatically updated when the track or playback status changes.
*Not interesting for now*

PUC-Rio - Certificação Digital Nº 1322105/CA

**# Windows 10 development for absolute beginners**
https://github.com/Windows-Readiness/AbsoluteBeginnersWin10
Hero Explorer (Video 71) The final example **allows the user to** explore Marvel Comic's universe of characters via their programmatic API. The application makes calls by create an MD5 hash of a public and private key as well as a timestamp to authenticate the call.
*Out of Context*

**Pandoc User's Guide**
https://github.com/jgm/pandoc
Specify a default extension to use when image paths/URLs have no extension. This **allows** you to use the same source for formats that require different kinds of images. Currently this option only affects the Markdown and LaTeX readers.
*Out of Context*

**#Sample Code#**
https://github.com/eway2012/SampleCode
QuickContacts demonstrates how to use the Address Book UI controllers and various properties such as displayedProperties, **allows** AddingToAddressBook, and displayPerson. It shows how to browse a list of Address Book contacts, display and edit a contact record, create a new contact record, and update a partial contact record.
*Out of Context*

**# Awesome Android**
https://github.com/JStumpp/awesome-android
[Android-Transition](https://github.com/kaichunlin/android-transition) - **Allows** the easy creation of view transitions that react to user inputs.
*Not interesting for now*. Comentário: Pode ser útil quando for desenvolver o app para Android.

PUC-Rio - Certificação Digital Nº 1322105/CA

## A1.4. IRR para o teste de precisão

### Domínio: Music Application

**0359.-.githubosu.-.iMusic.-.md.txt**
iMusic is a social-oriented music application for iOS devices. Along with acting as a media player, iMusic uses Facebook integration to show users what their friends have been listening to in a convenient feed. In addition, iMusic provides basic Youtube access to **allow users** to stream music found through Youtube’s service.

**0258.-.cheshire137.-.chrome_youtube2spotify.-.md.txt**
## Feature To-do List
* **Allow user** to enter subreddit and pull tracks from the API for that subreddit.
* Incorporate other sites, like Soundcloud or Vimeo, and try to find their Spotify tracks.

**0801.-.nzempsky.-.musicplayer.-.md.txt**
This is a desktop application that will **allow users** to search and play songs from spotify and/or soundcloud, and will display a visualizer while music plays.

**0860.-.Gabrielg1976.-.Dynamic-URL-Music-Generator.-..txt**
This needs to have a preset Audio sample like piano on shoots that can be mapped over keyboard interface..(future version will **allow user** to upload there own sample and map them across the musical Range)

**0897.-.solechang.-.iLList.-.md.txt**
A social media application that focuses on music. Application to **allow users** to collaborate with each other to create an ultimate playlist on realtime. Update: 3/1/16 Very first version of MusicLounge, back then called iLList. It is now open source!

**0789.-.bchambs.-.audiosearch.-.md.txt**
audiosearch is an open source web application designed to **allow users** to find new music. This website is an interactive implementation of The Echo Nest's web services.

**0787.-.figalex.-.rockolapp.-.md.txt**
School project of a cloud based application that will **allow users** to sort and manage music on cloud storage services like dropbox to be easier to listen to them.

**0595.-.orthogonal.-.Lyrics-Commander.-.txt.txt**
The service will **allow users** to play a game in which they tag song lyrics with emotive keywords. Then, that data can be used for interesting things, including suggesting songs to users. As data on tagging is collected, users that answer with the most popular tags will be rewarded in some way. Initially, the system will be fed a number of seed artists, albums, songs, and lyrics from Last.fm and song lyrics services.

**0653.-.Liurunex.-.iOS_RadioMusicPlayer.-.md.txt**
Developed the music radio application based on free API and music json data on internet by using Swift in iOS platform. Designed all elements and layouts of the music radio application.Developed radio view controllers including: the radio selection, radio playlist, music playing, music detail.Developed the http protocol to connect the radio application with free API and json data on internet.
Developed user personal view controller and functions which **allow user** adding, playing, and deleting music in the personal favorite playlist.

**0012.-.ClarSco.-.sheet-music-organiser.-.md.txt**

PUC-Rio - Certificação Digital Nº 1322105/CA

This will become an application that will **allow users** to catalogue their collection of sheet music

### 1147.-.samsquire.-.ideas.-.md

**Allow users** to provide corrections such as spelling (wiki functionality)
* Split up the document into reusable components (wiki like transclusion)
* Rendering data in multiple ways

Ideally the blogger wants to display product attributes consistently and update them from one place.

### 1203.-.migueldeicaza.-.MonoTouch.Dialog.-.markdown.txt

Use this element to **allow users** to load more items in your list. You can customize the normal and loading captions, as well as thefont and text color. The UIActivity indicator starts animating,and the loading caption is displayed when a user taps the cell,and then the NSAction passed into the constructor is executed.Once your code in the NSAction is finished, the UIActivity indicator stops animating and the normal caption is displayed again

### 1152.-.umano.-.AndroidSlidingUpPanel.-.md.txt

.2.1 * Add support for `umanoScrollInterpolator`
* Add support for percentage-based sliding panel height using `layout_weight` attribute * Add `ScrollableViewHelper` to **allow users** extend support for new types of scrollable views.

PUC-Rio - Certificação Digital Nº 1322105/CA

Domínio: Real Estate

**0745.-.wp-plugins.-.bepro-listings-real-estate.-.txt.tx**
BePro Listings Real Estate
There is an upgrade for this plugin on our website. With the upgrade, you are able to:
1. Search & filter listings by Real Estate criteria
2. Visible banner for sold listings
3. **Allow users** and agents to submit and manage their own listings from the front end
4. Details on listings (#beds, Sq Ft & #bath)
5. Form Builder integration providing searchable real estate details

**1004.-.uhub.-.awesome-lua.-.md.txt**
#awesome-lua A curated list of awesome Lua frameworks,
*LuaDocumentor **allow users** to generate HTML and API files from code documented using Lua documentation language.
* [JakobOvrum/LuaWeb](https://github.com/JakobOvrum/LuaWeb) - Small HTTP server in Lua
*[Hevienz/nginx-lua-ds-hotlink](https://github.com/Hevienz/nginx-lua-ds-hotlink) - 基于openresty的防盗链系统
* [ConnorVG/Lua-Slack]

**1014.-.agentevolution.-.wp-listings.-.md.txt**
# IMPress Listings
### Default Taxonomies and Terms ###
* Status (Active, Sold, Pending, For Rent, Reduced, New)
* Property Types (Residential, Condo, Townhome, Commercial)
* Location
* Features
Use the taxonomy creation tool to create your own way of classifying listings (i.e. bedrooms, bathrooms, locations, price ranges) and use those taxonomies to **allow users** to search for listings. Then, reorder the taxonomies as needed.
Find listings by taxonomy using filters in the WordPress admin

**1031.-.hadley.-.data-housing-crisis.-.md.txt**
Project Overview
* PUMA (Public Use Microdata Area): PUMA consists of 5% of the population.
We would like to develop a website that will **allow users** to easily access the data they are interested in, which would otherwise be a daunting task for those who wish to use a data set of this size. Because our analysis and findings also involve large amounts of information, (such as construction price time series for each US metropolitan area) we are exploring interactive graphical methods for displaying this information.

**1094.-.north.-.north.-.md.txt**
Consistency and Predictability
Users really like consistency in their design. Consistency and predictability in design **allow users** to feel safe and confident as they navigate. It reduces cognitive load for the user, allowing them to focus on the content of a site. This means that, when designing, instead of designing pages, component based systems should be designed and pieces of those systems reused to build pages throughout the site. Titles, button names, menu items, and other interaction related copy should be filled with trigger words, words that inspire users to act as the outcome is obvious.

PUC-Rio - Certificação Digital Nº 1322105/CA

Domínio: Digital Library

**0439.-.AmitSupugade.-.library_system.-.md.txt**
# library_system
Python- Digital library system to keep track of books in the library and users of the library Problem statement- A library of books and users. This is a (simplistic) library of books and users where we can add books and users, and **allow users** to checkout those books.

**0479.-.bigbank-as.-.digidoc.-.md.txt**
# PHP Library for Estonian Digital ID
7.1 MobileAuthenticate`
#### Usage
See examples/mobile/authentication.php](examples/mobile/authentication.php).
### Digital Signature With Mobile ID **Allow users** to digitally sign files using mobile ID. Interface Name: `FileSignerInterface` - DigiDocService: `8. Queries and Responses for Digital Signature

**1157.-.samsquire.-.ideas.-.md.txt**
One Hundred Ideas for Computing
Information can be collected *within* the article such as small call to action button buttons that when clicked add form fields for the user to contribute to.
* **Allow users** to provide corrections such as spelling (wiki functionality)
* Split up the document into reusable components (wiki like transclusion)
* Rendering data in multiple ways
Examples: ``` A blogger is reviewing a number of products on an ongoing basis. The blogger creates a post for a given product category and headings for
each offering. Attributes and ratings are placed in a bulleted list.

**1159.-.chiraggude.-.awesome-laravel.-.md.txt**
Awesome Laravel [![Awesome]
Laravel package to generate a UUID according to the RFC 4122 standard
* [Laravel Installer](https://github.com/RachidLaasri/LaravelInstaller) - Laravel package to **allow users** to install your application just by following the setup wizard, like WordPress.

**1222.-.Pi4J.-.pi4j.-.md.txt**
Pi4J :: Java I/O Library for Raspberry Pi
Added GPIO based 4/8 bit LCD WiringPi example program
* Added support for a GpioController.shutdown() method to cleanup terminate all Pi4J threads and executors. https://github.com/Pi4J/pi4j/issues/9
* Added support for a user-definable ExecutorServiceFactory to **allow user** program to provide the implementation for executor service instances and managed thread pools.

PUC-Rio - Certificação Digital Nº 1322105/CA