RETRIEVING CURATED STACK OVERFLOW POSTS OF SIMILAR PROJECT TASKS

Gláucia Melo dos Santos

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Toacy Cavalcante de Oliveira

Rio de Janeiro
Dezembro de 2018

RETRIEVING CURATED STACK OVERFLOW POSTS OF SIMILAR PROJECT TASKS

Gláucia Melo dos Santos

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

____

Prof. Toacy Cavalcante de Oliveira, D.Sc.

____

Prof. Cláudia Maria Lima Werner, D.Sc.

____

Prof. Eber Assis Schmitz, PhD.

RIO DE JANEIRO, RJ - BRASIL

DEZEMBRO DE 2018

Santos, Gláucia Melo dos

Retrieving curated Stack Overflow Posts of similar project tasks / Gláucia Melo dos Santos – Rio de Janeiro: UFRJ/COPPE, 2018.

XI,118 p.:il.; 29,7cm.

Orientador: Toacy Cavalcante de Oliveira

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2018.

Referências Bibliográficas: p.92-99

1. Software Engineering. 2. Curated Stack Overflow Posts. 3. Project Task Context Similarity. I. Oliveira, Toacy Cavalcante de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

A única maneira de começar é agradecendo a Deus. Sou imensamente grata. Agradeço aos meus pais Fátima e Cristóvão, minha irmã Gisele, minha madrinha Lila e prima Thaisa pelo apoio ILIMITADO enquanto estive buscando o título de mestre e nas tantas outras fases.

Agradeço ao meu orientador Toacy, por toda a sua paciência e apoio, além de me impulsionar ao máximo como acadêmica, sempre respeitando e tentando tirar o máximo proveito do meu histórico profissional. Também por me dar a oportunidade de visitar a Universidade de Waterloo. Agradeço também à sua família por ter desempenhado um papel muito importante durante minha visita a Waterloo. Estou muito feliz por ter tido essa oportunidade e por ter tido a chance de trabalhar com os incríveis professores Paulo Alencar e Don Cowan.

Agradeço a todos os meus colegas do Grupo Prisma pelo apoio, e aos meus amigos do mestrado Victor e Rachel. Agradeço ao Luiz Oliveira, aos professores Daniel e Jano, com quem publiquei meu primeiro artigo. Agradeço também ao Global Affairs Canada, por financiar parte de minha pesquisa por meio do programa de bolsas ELAP. Obrigada a empresa SpazioDati que nos permitiu utilizar os serviços de sua API.

Alguns amigos me apoiaram incrivelmente durante o caminho à esta conquista: todos os meus amigos da OWSE, que me ajudaram no trabalho enquanto eu estava no campus; Ulisses, que me ouviu durante esses 3 anos; Bruno Brazil, que me ajudou com dados para meus estudos; Paola Fernandes, que foi minha roommate durante o primeiro ano de Mestrado e respeitou minha dedicação, cuidando da casa e cuidando de mim mesma quando a única coisa que eu conseguia fazer era conciliar trabalho e estudos; Paola Kozlowiski, por todas as longas conversas e por me motivar toda vez que eu pensava que não era boa o suficiente, e por me receber em sua casa me ajudando por muitas vezes com a logística para assistir às aulas, assim como minha prima Nathalia. Agradeço também ao meu amigo e mentor Walter Magioli, que me incentivou a realizar esse sonho desde o início.

E por último, mas não menos importante, agradeço à UFRJ e aos professores por me darem a chance de trabalhar e aprender com esse incrível grupo de pesquisadores. Considero como um sonho realizado e, para ser honesta, ainda é difícil acreditar em ter tido a chance de trabalhar com professores de tão alto nível e que tanto me ensinaram.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

RECUPERANDO POSTS DO STACK OVERFLOW QUE SOFRERAM CURADORIA DE TAREFAS SIMILARES

Gláucia Melo dos Santos

Dezembro/2018

Orientador: Toacy Cavalcante de Oliveira

Programa: Engenharia de Sistemas e Computação

O desenvolvimento de software depende de diversas tecnologias e métodos e, como resultado, as equipes de desenvolvimento de software geralmente lidam com problemas em que não são especialistas. Para lidar com a falta de conhecimento, desenvolvedores normalmente procuram informações em sites de perguntas e respostas, como o Stack Overflow, um site usado para encontrar soluções para problemas específicos relacionados à tecnologia. O acesso a esses sites não é integrado ao ambiente de desenvolvimento de software e porque as associações entre os projetos de desenvolvimento de software e as fontes de suporte de soluções conhecidas não são explicitamente registradas. Com isso, desenvolvedores de software podem investir um esforço em procurar soluções para problemas semelhantes várias vezes. Essa falta de integração dificulta o reuso do conhecimento obtido, além de não evitar esforços de busca e seleção, a curadoria, repetidas vezes. Esta pesquisa tem como objetivo realizar um estudo sobre a associação explicita entre elementos do projeto (como tarefas de projeto) a publicações do Stack Overflow que já sofreram curadoria por desenvolvedores, e apresenta um estudo sobre sugestões de publicações do Stack Overflow a desenvolvedores com base na similaridade de tarefas de projeto.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

RETRIEVING CURATED STACK OVERFLOW POSTS OF SIMILAR PROJECT TASKS

Gláucia Melo dos Santos

December/2018

Advisor: Toacy Cavalcante de Oliveira

Department: Computer Science Engineering

Software development depends on diverse technologies and methods and as a result, software development teams often handle issues in which team members are not experts. In order to address this lack of expertise, developers typically search for information on web-based Q&A sites such as Stack Overflow, a well-known place to find solutions to specific technology-related problems. Access to these web-based Q&A locations is currently not integrated into the software development environment, and since the associations between software development projects and the supporting sources of known solutions, usually referred to as knowledge, is not explicitly recorded, software developers often need to search for solutions to similar recurring issues multiple times. This lack of integration hinders the reuse of the knowledge obtained, besides not avoiding efforts of search and selection, curation, of this knowledge over and over again. This research aims at proposing a study regarding explicitly associating project elements (such as project tasks) to Stack Overflow posts that have already been curated by developers, and presents a study about Stack Overflow posts suggestions to developers based on similarity of project tasks.

INDEX

INDEX OF FIGURES

INDEX OF TABLES

INDEX OF EQUATIONS

1 Introduction

This chapter introduces the dissertation and presents its context and main objectives. It also presents the methodology and how this document is divided and organized over the next chapters.

1.1 Context and Motivation

Software development is a knowledge-intensive collaborative activity (DI CICCIO *et al.*, 2015) (VASANTHAPRIYAN *et al.*, 2015). Currently, the development environment constantly changes with the variety of technologies in use. Therefore, new knowledge must be constantly gathered and software engineers need to engage in tasks that are related to knowledge management, such as learning, capturing and reusing collaborative knowledge during a software project (VASANTHAPRIYAN *et al.*, 2015). Software development tasks are organized through a software development process and performed by people. During the execution of a software development process (the act of working on tasks related to software development), knowledge and expertise from developers are vital for the project to succeed (VASANTHAPRIYAN *et al.*, 2015).

There are a number of components that support software development, such as code editors and debuggers (ROBERTO MINELLI *et al.*, 2015). The software development environment is usually an integrated system of these components with which developers interact to build a software product. In order to acquire external support (e.g., code snippets), developers frequently switch between a development environment and browsers (MEYER *et al.*, 2017). The support tools where developers search for assistance and the development environment are not closely coupled. Developers must leave the development environment, reason about relevant and accurate terms for searches, open a browser, verify the results of the search, check if the source is reliable and only then transfer the knowledge obtained to the software and test the code or information (PONZANELLI *et al.*, 2014) (MEYER *et al.*, 2017). Such activity usually occurs more than once, as software projects are large-scale organized processes (software projects are iterative).

Important sources of knowledge - which is the information software developers use to support them while working - are Question and Answer (Q&A) websites. These websites represent communities where people ask and/or answer questions to get support for whatever problem or doubt they encounter. Many Q&A websites have voting and reputation systems implemented to ensure users are likely to get valuable content (ANDERSON *et al.*, 2012). A famous Q&A site for software engineers is Stack Overflow (MAMYKINA et al., 2011; FUMIN et al., 2016; LIU et al., 2016; SAHU et al., 2016), a community with millions of users that has become famous as a main source of knowledge for developers. Stack Overflow is focused on getting detailed answers about technical problems commonly found during software development. Stack Overflow is mostly, but not uniquely, used during the development phase that comprises coding and testing. Stack Overflow allows developers to place questions regarding problems they have encountered so anyone can attempt to answer the questions. Even though Stack Overflow participants might not be aware of specific details of a project or company, as long as the developer provides enough information for the problem to be understood or reproduced, one or more persons will likely work on an answer.

Although Stack Overflow is widely used during software development as a source of knowledge for developers (MAMYKINA et al., 2011), there are still issues about explicitly associating the tasks performed during development and the knowledge obtained from a Stack Overflow Post. Researchers have pointed the lack of integration of the support often needed by software developers with the project as an open issue (WANG *et al.*, 2014) (PONZANELLI *et al.*, 2014) (PONZANELLI *et al.*, 2013); Researches also point the lack of integration between IDEs and software development team's workflow as a problem (PONZANELLI *et al.*, 2014). Some papers have proposed solutions to integrate Stack Overflow to the software project, but mainly focusing in issues (bugs, exceptions) and not the project plan. There is research stating that developer's expertise results in successful projects (VASANTHAPRIYAN *et al.*, 2015). This expertise is not considered when the selection of Stack Overflow Posts is totally automated. Also, for projects that are not managed in English language, this can be challenging, considering Stack Overflow content is mainly in English language. And research has already pointed the text overlap between issues and Stack Overflow can be as low as 16% (CORREA & SUREKA, 2013).

Researchers have worked on investigating the daily routine of software developers, by observing how they perform their tasks, in order to perceive how developers organize their work. Studies found that developers can spend most of their time reading documentation (SINGER *et al.*, 2010), and they also discovered search tools are massively used (MEYER *et al.*, 2017). More specifically, 57% of the daily routine of developers is invested in fixing bugs; making enhancements on the system consumes 35% of their day. Searching for external sources of knowledge is not only a daily and usual activity performed by software developers, but also part of most of their workday.

The effort of tapping into sources of support, reasoning about the help needed and choosing among the vast available content will be referred in this dissertation as *curation*. This term, inspired by the arts fields, has the following meaning in Oxford dictionary:

> *“Select, organize, and look after the items in (a collection or exhibition) … or*
>
> *Select, organize, and present (online content, merchandise, information, etc.), typically using professional or expert knowledge”*

This effort of tapping into documentations and sources of knowledge is a great part of developer’s routine, and is not coupled to the development project.

In summary, the problems identified are:

- Developers employ tacit knowledge or knowledge that was discovered during development when looking in Stack Overflow, and these Posts are not **explicitly captured** in the software project;
- Search results are **detached** (PONZANELLI et al., 2014) from project, meaning there is no explicit association between curated Stack Overflow Posts and software development;
- The curation effort, a time-consuming part of the daily life of software developers, could be reused due to the iterative characteristic of software development, but is **hardly reused**;

The research reported in this dissertation aims at capturing and reusing curation effort by proposing to use project task information to identify and integrate the curated Stack Overflow Posts with the software development project. This integration allows Stack Overflow Posts suggestions and reuse once similar project tasks are identified. We believe there is a possibility to suggest Stack Overflow Posts by integrating the curated Stack Overflow Post to the project and using project task text similarity to aid this suggestion, and this dissertation provides indications that this is possible.

Integrating curated Stack Overflow posts to software development projects could help developers avoid performing curation of information multiple times, redundantly. This work proposes to address the problems by (1) associating curated Stack Overflow posts used to complete a project task with that task; (2) identifying a project task context, (3) proposing a study to investigate the possibility to relate project tasks by their similarity; (4) building an implementation that is able to retrieve project task similarities, and (5) evaluating the study. The goal is to investigate the possibility to reuse Stack Overflow Posts based on the similarity of the project tasks they are working on or have worked on, and understand how project task context can influence the effectiveness of Stack Overflow Posts suggestion.

1.2 Objectives

This dissertation aims at investigating the possibility to reuse curated Stack Overflow Posts through the identification of similar project tasks. To investigate the problems researched and to address the investigations, an objective for this work was established. To define a goal for this study the GQM (Goal, Question, Metric) approach (VAN SOLINGEN *et al.*, 2002) was used. According to the GQM approach, the main objective of this work is to

analyze *project task context and similarity*
with the purpose of *reusing curated Stack Overflow Posts*
from the point of *software developers*
in the context of *software project development.*

The specific goals towards this main goal are:

Obtaining the state-of-the-art of existing approaches that associate software development projects with Stack Overflow. This objective aims to solve the lack of vision of the state of the art in the area of strategies that aim to associate the knowledge available in Stack Overflow to software development. To do this, it is important (1) to investigate the current proposed strategies, and (2) to identify what the proposals have in common, what are the most common metrics and to map the results in each selected paper. To accomplish the objectives, this dissertation aims at (1) presenting a systematic mapping study of the current state of the literature and (2) identifying the characteristics of the retrieved articles, in order to support the elaboration of the proposal in this work.

Identification of project task contexts. It is important to identify a set of project task contexts that effectively allow curated Stack Overflow Post reuse through project task similarity retrieval.

Implementation of a process to retrieve project task similarities. In order to assess the possibility to reuse curated Stack Overflow Posts when project tasks are similar, we believe it's important to build as part of the study an implementation of a process that retrieves project task similarities.

Evaluation. Obtain metrics for the provided implementation in order to quantitatively assess the possibility to reuse curated Stack Overflow Posts, as well as to investigate possible outcomes when project context elements vary.

1.3 Methodology

The methodology presented conducted the research, in order to achieve the objectives proposed in Section 1.2. Figure 1.1 presents this dissertation's methodology timeline and activities, which will be detailed further in this work.

1. **Selection of research topic:** this dissertation's subject arose from one master course "Knowledge Intensive Processes" and from the participant's industry experience in software engineering and use of Stack Overflow support. This brought the interest in this subject among the research participants and these discussions supported the selection of this topic.
2. ***Ad-Hoc* Literature Review:** after the topic selection, an *ad-hoc* literature review was conducted in order to gather detailed information regarding associating Stack Overflow Posts to software development. During this review, we observed there are a number of researches aiming at solving this problem, indicating there is a concern in decreasing the separation between Stack Overflow content and software development.
3. **Preliminary Assessment:** after gathering existing information in literature of existing solutions for the problem, we conducted a preliminary assessment, a Java implementation with industry practitioners, in order to get indications if (1) the problem identified existed in an industry scenario, and (2) the idea of the proposal could possibly allow Stack Overflow Post reuse.

4. **Systematic Mapping Study:** considering the positive outcome of the preliminary assessment, a Systematic Mapping Study (SMS) was performed in order to obtain information regarding current researches on software development and Stack Overflow Post association. The articles found were important for getting insights about the problem we've identified, as well as current proposed tools and studies.
5. **Data Collection and Implementation:** after the SMS was concluded and a broader view of research was settled, an implementation of the proposal was built in R language and using Dandelion similarity API, which proved to be powerful tools. Although results were good, this phase ended up serving as idea improvement, rather than for the final proposal. Dandelion belongs to a private company and they don't share the code, hampering the proposal's reproduction. For this reason, this implementation is not described in this dissertation. The dataset collected to test this implementation was kept for further implementations.
6. **Building Process and Evaluating:** with the idea of the proposal matured by the experience obtained in the previous implementation, we developed a process using *RapidMiner*, a powerful data mining tool. Both proposal and evaluation were implemented in this phase of the research. The sample used to evaluate the process was the same sample gathered in the previous step of this methodology.
7. **Systematic Mapping Study Update:** an update of the previous SMS was performed in this step, in order to assess recent papers and test if results and conclusions to include or exclude papers would be different from the first SMS performed.
8. **Final report development:** all performed steps for this research were organized in this dissertation format, providing extensive details regarding findings.

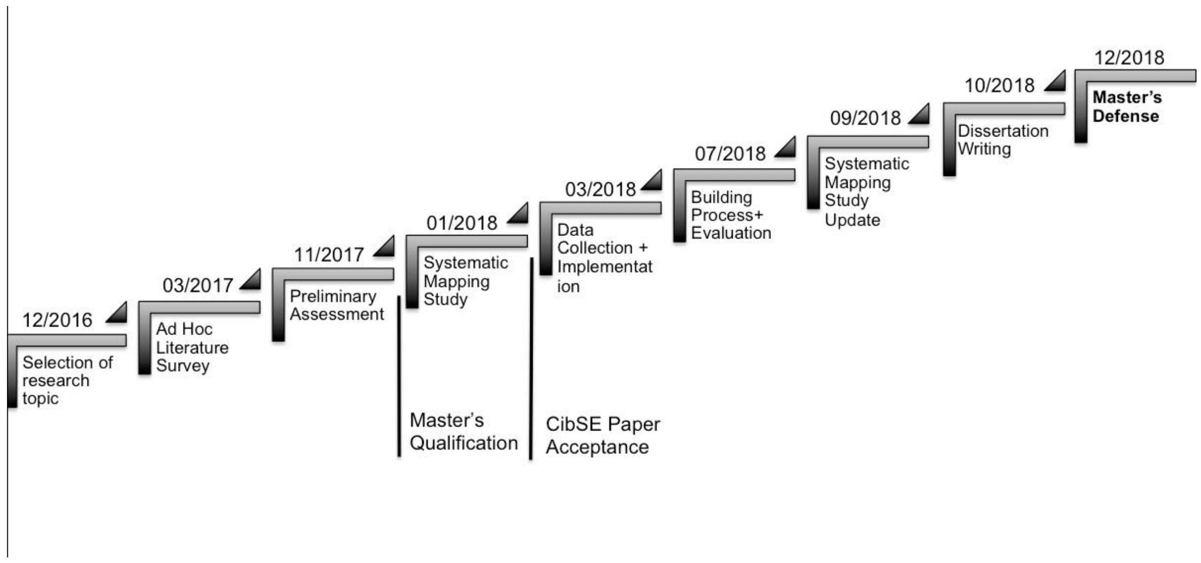


Figure 1.1: Methodology timeline.

The master's qualification was presented in November 23th, 2017. The qualification included the initial literature reviews and the preliminary assessment results. The results of the preliminary assessment were published in XXI Iberoamerican Conference on Software Engineering – CiBSE 2018 (Gláucia Melo, Ulisses Telemaco, Toacy Oliveira, Paulo Alencar, Don Cowan. "**Towards using task similarity to recommend Stack Overflow posts**". Proceedings of CiBSE, 2018.)

1.4 Organization

The overall structure of this dissertation takes the form of six chapters and six appendices. Besides the introduction, which explores the problem, motivations, objectives and methodology of this research, the following chapters are presented:

Chapter 2 – Theoretical Foundation: Chapter two begins by laying out the theoretical dimensions of the research, and looks at important concepts on Software Development. It also presents Question and Answer websites, and Stack Overflow, laying out its importance to the development community worldwide. Data Mining, Recommendation Systems and Similarity strategies for recommendation systems are also presented, since concepts from these theories are used in this dissertation development.

Chapter 3 – Systematic Mapping Study: the Systematic Mapping Study plan, execution, results and discussion are presented in this chapter. It features the study planning, articles selection criteria and presents main findings in literature regarding approaches that associate Stack Overflow and software development. Chapter 3 comprises methodology steps #4 and #7.

Chapter 4 – Proposal development and implementation: the study of curated Stack Overflow Posts reuse is described in this chapter. It presents the project task context identification and details the implementation of a process that retrieves project task similarities. This chapter presents step #6 of the methodology.

This Chapter also presents a preliminary study performed in the early stages of this research, step #3 of the methodology.

Chapter 5 – Evaluation: Chapter five presents the execution of the process implemented. The results are presented for both proposed evaluations, reporting precision and accuracy metrics. This chapter describes the evaluation step #6 and uses the data collected in step #5 of this dissertation's methodology.

Chapter 6 – Conclusions: presents a final discussion, future work possibilities and also discusses limitations of this work.

Appendix A – Dandelion+R Implementation Code: it presents the source code for the R implementation, using Dandelion similarity API. This was step #5 of this dissertation.

Appendix B – Preliminary Assessment Implementation Code: here we present the souce code for the preliminary assessment. It was a Java implementation using a text lists of data.

Appendix C – Dataset Sample: it includes the information of the dataset used in the evaluation step.

Appendix D – SMS exclusion form: we present here a form structured with information from the excluded papers of the Related Work section. It presents each excluded work and the reason for exclusion.

Appendix E – *RapidMiner* process XML: here we present the XML source code of the process implemented in *RapidMiner*.

Appendix F – More combinations of project task context elements: finally, here we present results for a wider list of project task context combinations, other than the ones presented in the Evaluation Chapter 5, performed using *RapidMiner*.

2 Theoretical Foundation

This chapter contains details of the concepts involved in this dissertation. It starts with an introduction of the further connecting and describing each topic.

2.1 Introduction

This dissertation proposes a study to relate similar project tasks in order to reuse Stack Overflow Posts, according to the project task's similarity. Project task context elements are used to discover the similarity. The similarity comparison is conducted for pairs or project tasks. Each pair of tasks is submitted to similarity retrieval between pairs of project tasks. A high similarity can indicate that the tasks could share the same Stack Overflow Post associated to one of them. These main concepts mentioned are described in this section. This section presents an overview of Software Processes, given project tasks (elements' which similarities will provide the source information for similarity retrieval) can be derived from Software Processes. Q&A websites and Stack Overflow, more specifically, are also discussed in this section; hence the knowledge support source for developers discussed in this work is Stack Overflow. This chapter also presents concepts of Data Mining and Recommendation Systems for Software Engineering, as well as Similarity metrics, respectively, since those concepts are also related to the study product of this dissertation. Finally, considerations over the theoretical foundation are made; more specifically how these concepts are aligned and together conceive this work's background.

2.2 From Software Processes to Software Projects

Processes are an important concept to be used during software development, in order to work towards the goal of software quality (FUGGETTA, 1996), or the steps towards achieving a goal (FEILER & HUMPHREY, 1993). The dynamic work of developing software is constantly evolving according to the technology needs of the company and also of the software practices employed (AURUM *et al.*, 2008). Software processes can be used in different contexts, and are heavily context-dependent, which means their outcomes vary according to the development environment, and comes as a way to support the process improvement, management and also provide guidance in performing the process (MÜNCH *et al.*, 2012). There are definitions, such as Rocha's (DA ROCHA, *et al.*, 2001), which suggests that software development involves a transformation phase between Process Modeling and Process Execution, each with characteristics of its own. Process modeling defines essentially a process methodology and process execution provides the steps to be executed as part of that process (MÜNCH *et al.*, 2012). Part of this transformation – a conceptual piece of the process lifecycle - is represented in Figure 2.1. Quality standards, maturity models and software development methodologies, as well as other technological or software architectural related concepts outline the *Concept Cloud* presented in Figure 2.1. Then, this *Concept Cloud* (1) is synthesized into Process Models (2), which contains information about process activities and their relations, considering what are the specialized needs from the organization, work groups and projects (DA ROCHA, *et al.*, 2001). After instantiation, processes can be executed, and project tasks are produced as part of the *Project Work Plan* (3) creation. At this stage, it is possible to create and work on project tasks. Other authors have found this approach to Software Process transformation useful for Process Representation (CAMPOS & OLIVEIRA, 2013), Process Tailoring (PILLAT *et al.*, 2015) and Process Mining (M. VALLE *et al.*, 2017) (SANTOS *et al.*, 2015). While in the Project Work Plan (3) stage, project tasks are created and developers are able to work on these tasks. In this step of software construction, knowledge from software developers, as well as knowledge from a variety of sources is massively needed. The Project Work Plan (3) is the process model with specific characteristics of the organization and deadlines of the project, for example, resources for projects, which developers work on the project and which are responsible for each task and also deadlines for tasks. This is the stage where the project can be started and in which tasks can be executed by developers or automatically (REIS, 2003). In the context of this work, the Project Work Plan (3) can be associated to Stack Overflow (4), which is also represented in Figure 2.1.

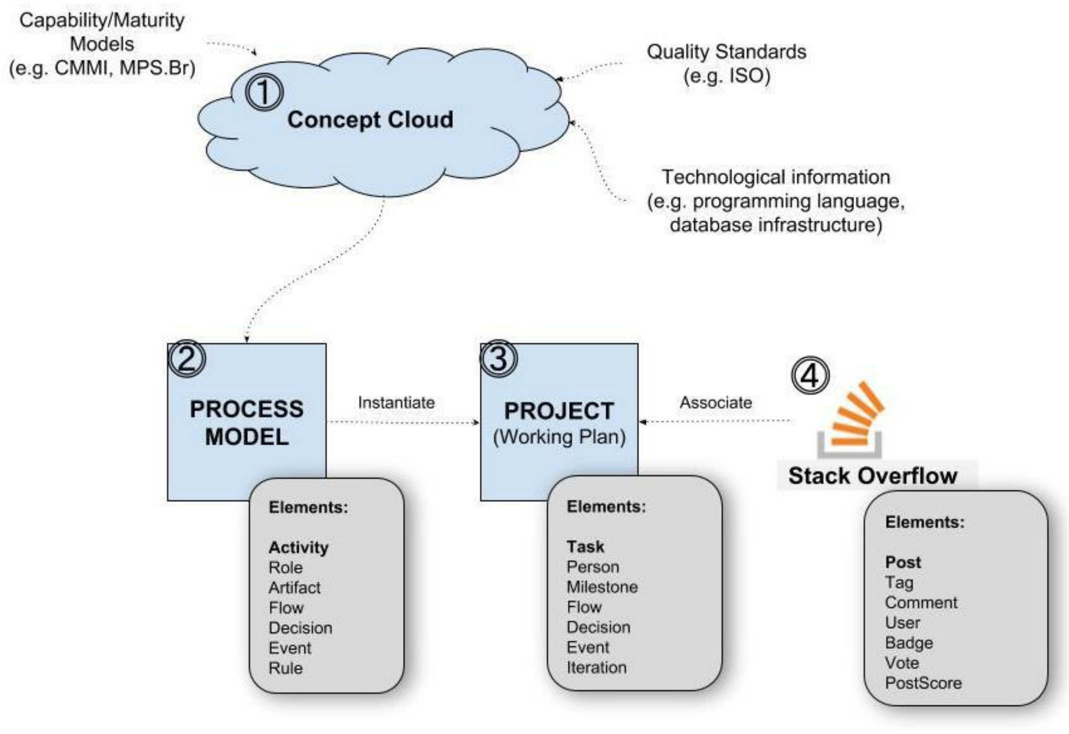
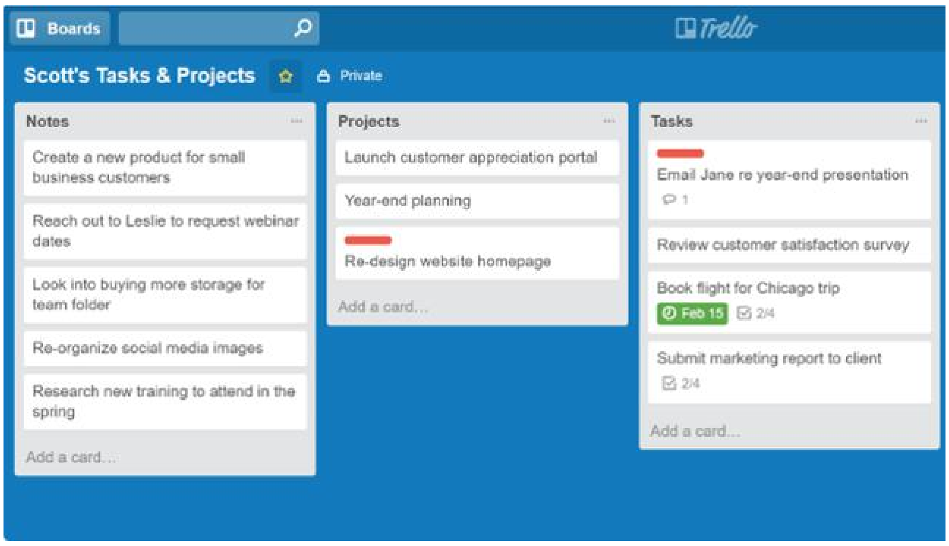

Capability/Maturity Models (e.g. CMMI, MPS.Br)
① Concept Cloud
Quality Standards (e.g. ISO)
Technological information (e.g. programming language, database infrastructure)
② PROCESS MODEL
Instantiate
③ PROJECT (Working Plan)
Associate
④ Stack Overflow
Elements: Activity Role Artifact Flow Decision Event Rule
Elements: Task Person Milestone Flow Decision Event Iteration
Elements: Post Tag Comment User Badge Vote PostScore

Figure 2.1: From process concept to project task.

2.2.1 Project Work Plan and Project Management Tools

When describing the Work Plan – Project (3) - presented in Figure 2.1, there are some elements in this Work Plan, such as: Task, Person, Milestone, Flow, Decision, Event and Iteration. Task element, is a task executed by a software developer or automatically (REIS, 2003). Tasks have attributes, such as start and end dates, the person responsible for working in the task (assignee), the complexity and priority of resolution, among other elements. In other words, Tasks are pieces of actions each software developer has to perform in a certain amount of time during the Project in order to obtain the final product: the software or part of the software. These tasks are materialized from project business needs and have to consider other aspects, such as the size of the team, the desired client's deadline, technology used to develop software, infrastructure configuration and other specific conditions. When project tasks are defined, there are tools that can be used to manage the execution of these tasks. Tools can be project management tools or even a white board with post-its for each task. Some examples of well-known project management tools used in industry are Trello[10], Redmine[11] and Jira[12], presented below in Figure 2.2, Figure 2.3 and Figure 2.4, and a white board as tool for managing tasks is presented in Figure 2.5. Figure 2.2

presents a project that is managed in the Project Management Tool Trello[10]. The project name in this example is *Scott's Tasks and Projects* and there is one column for *Notes*, another one for *Projects* and another for *Tasks*.



Scott's Tasks & Projects

Notes
- Create a new product for small business customers
- Reach out to Leslie to request webinar dates
- Look into buying more storage for team folder
- Re-organize social media images
- Research new training to attend in the spring
- Add a card...

Projects
- Launch customer appreciation portal
- Year-end planning
- Re-design website homepage
- Add a card...

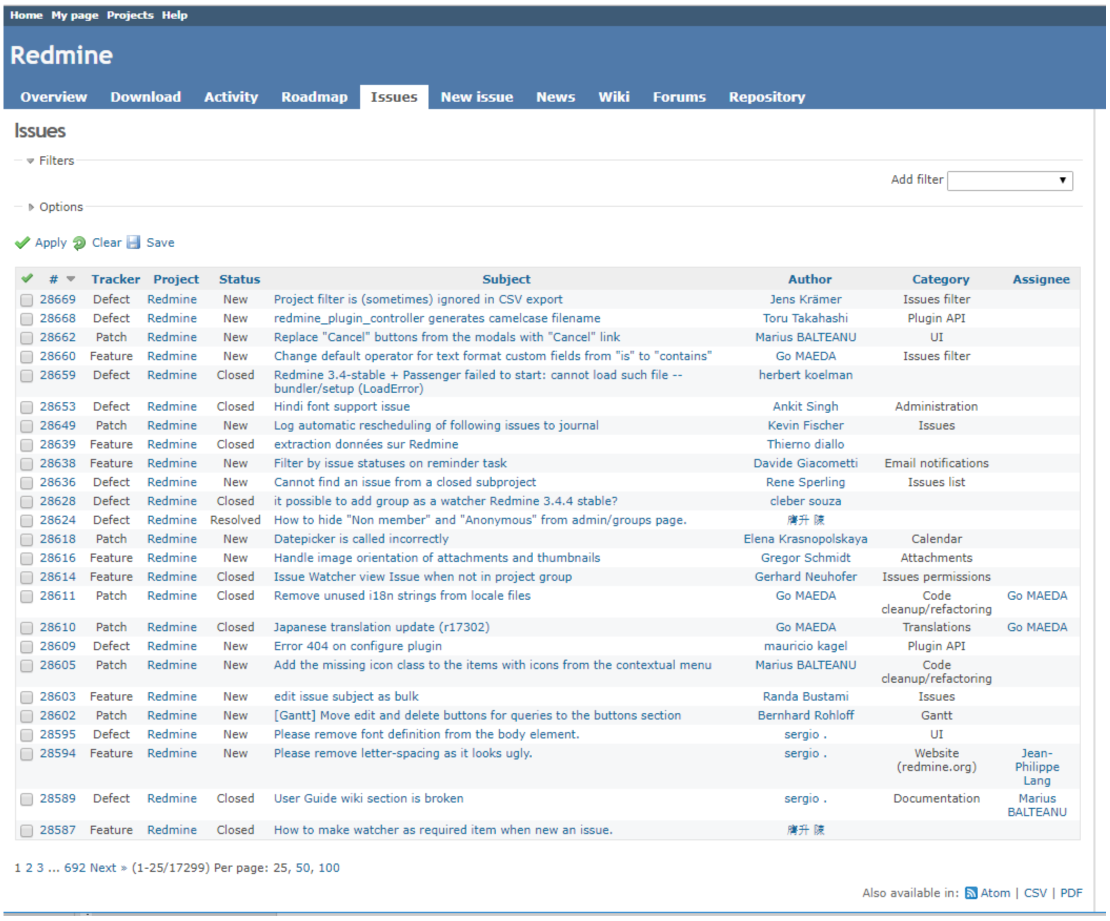
Tasks
- Email Jane re year-end presentation (1)
- Review customer satisfaction survey
- Book flight for Chicago trip (Feb 15, 2/4)
- Submit marketing report to client (2/4)
- Add a card...

Figure 2.2: Trello screen.

Source: https://blog.trello.com/why-your-to-do-list-is-going-nowhere

Another example of a task list is presented in Figure 2.3. This figure presents a list of project tasks from the *Redmine* project, which is publicly available. This list has the information of the type of the Task, identified by the *Tracker* column, and other columns such as also *Project, Status, Subtitle, Author, Category* and *Assignee*. One example of a task found in this list has as *Subtitle* "Error 404 on configure plugin".

Figure 2.4 presents another example of a task list, now in *Jira* tool (source: https://marketplace.atlassian.com/plugins/com.stiltsoft.connect.jira.todo/cloud/overview . There are tasks such as "Fixing localization errors" and "Fixing interface glitches". These are examples of tasks that guide developers during their workflow throughout the project. Another way to present and organize project tasks, for example, is to keep them in post-its in a board that is hanging on a wall, as presented in Figure 2.5 (source: https://goo.gl/images/iZ5daR).

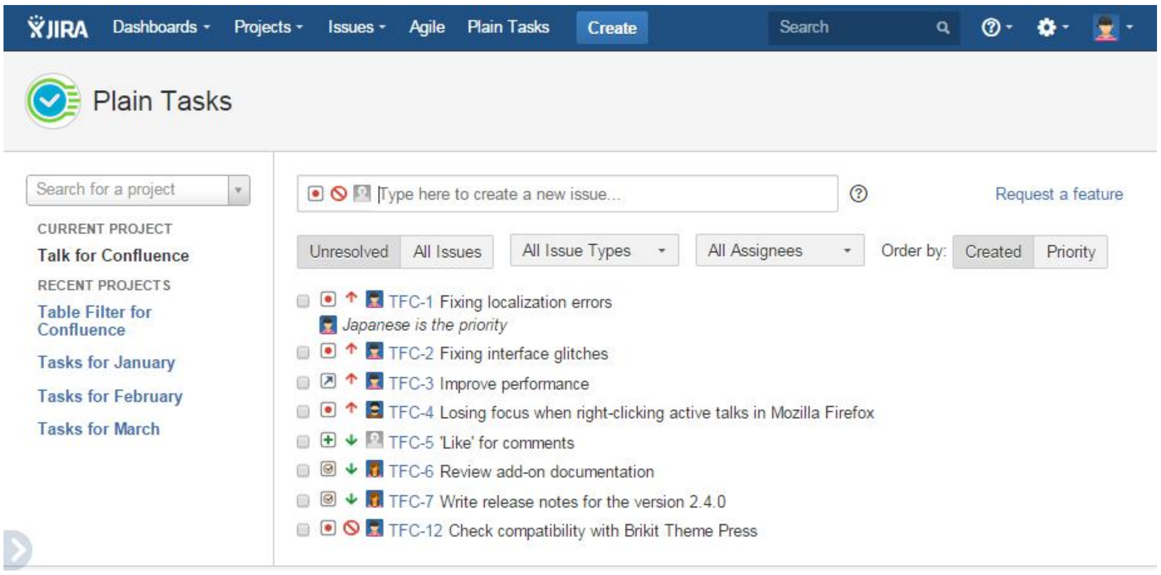

Home My page Projects Help
Redmine
Overview Download Activity Roadmap Issues New issue News Wiki Forums Repository
Issues
Filters
Add filter
Options
Apply Clear Save

✓	#	Tracker	Project	Status	Subject	Author	Category	Assignee
☐	28669	Defect	Redmine	New	Project filter is (sometimes) ignored in CSV export	Jens Krämer	Issues filter	
☐	28668	Defect	Redmine	New	redmine_plugin_controller generates camelcase filename	Toru Takahashi	Plugin API	
☐	28662	Patch	Redmine	New	Replace "Cancel" buttons from the modals with "Cancel" link	Marius BALTEANU	UI	
☐	28660	Feature	Redmine	New	Change default operator for text format custom fields from "is" to "contains"	Go MAEDA	Issues filter	
☐	28659	Defect	Redmine	Closed	Redmine 3.4-stable + Passenger failed to start: cannot load such file -- bundler/setup (LoadError)	herbert koelman		
☐	28653	Defect	Redmine	Closed	Hindi font support issue	Ankit Singh	Administration	
☐	28649	Patch	Redmine	New	Log automatic rescheduling of following issues to journal	Kevin Fischer	Issues	
☐	28639	Feature	Redmine	Closed	extraction données sur Redmine	Thierno diallo		
☐	28638	Feature	Redmine	New	Filter by issue statuses on reminder task	Davide Giacometti	Email notifications	
☐	28636	Defect	Redmine	New	Cannot find an issue from a closed subproject	Rene Sperling	Issues list	
☐	28628	Defect	Redmine	Closed	it possible to add group as a watcher Redmine 3.4.4 stable?	cleber souza		
☐	28624	Defect	Redmine	Resolved	How to hide "Non member" and "Anonymous" from admin/groups page.	膺升 陳		
☐	28618	Patch	Redmine	New	Datepicker is called incorrectly	Elena Krasnopolskaya	Calendar	
☐	28616	Feature	Redmine	New	Handle image orientation of attachments and thumbnails	Gregor Schmidt	Attachments	
☐	28614	Feature	Redmine	Closed	Issue Watcher view Issue when not in project group	Gerhard Neuhofer	Issues permissions	
☐	28611	Patch	Redmine	Closed	Remove unused i18n strings from locale files	Go MAEDA	Code cleanup/refactoring	Go MAEDA
☐	28610	Patch	Redmine	Closed	Japanese translation update (r17302)	Go MAEDA	Translations	Go MAEDA
☐	28609	Defect	Redmine	New	Error 404 on configure plugin	mauricio kagel	Plugin API	
☐	28605	Patch	Redmine	New	Add the missing icon class to the items with icons from the contextual menu	Marius BALTEANU	Code cleanup/refactoring	
☐	28603	Feature	Redmine	New	edit issue subject as bulk	Randa Bustami	Issues	
☐	28602	Patch	Redmine	New	[Gantt] Move edit and delete buttons for queries to the buttons section	Bernhard Rohloff	Gantt	
☐	28595	Defect	Redmine	New	Please remove font definition from the body element.	sergio .	UI	
☐	28594	Feature	Redmine	New	Please remove letter-spacing as it looks ugly.	sergio .	Website (redmine.org)	Jean-Philippe Lang
☐	28589	Defect	Redmine	Closed	User Guide wiki section is broken	sergio .	Documentation	Marius BALTEANU
☐	28587	Feature	Redmine	Closed	How to make watcher as required item when new an issue.	膺升 陳		

1 2 3 ... 692 Next » (1-25/17299) Per page: 25, 50, 100

Also available in: Atom | CSV | PDF

Figure 2.3: Redmine task list. Source: redmine.org.



TFC-1 Fixing localization errors
Japanese is the priority
TFC-2 Fixing interface glitches
TFC-3 Improve performance
TFC-4 Losing focus when right-clicking active talks in Mozilla Firefox
TFC-5 'Like' for comments
TFC-6 Review add-on documentation
TFC-7 Write release notes for the version 2.4.0
TFC-12 Check compatibility with Brikit Theme Press

Figure 2.4: Jira task list.

Figure 2.5: Scrum Tasks white board example.

2.2.1.1 Project Tasks and Issues

Project tasks are the steps towards the software product, as defined by (FEILER & HUMPHREY, 1993). Project Tasks might have information about technology as well as information regarding the software domain or business of application (LINDVALL & RUS, 2003). *Issues* are problems, such as errors and defects, which may occur during software development. Issues may or may not have information about the domain, as they are mostly errors and defects of the technologies that are being used during software development. Stack Overflow content can support both, although Stack Overflow does not have domain information. This is why when considering project tasks, the role of a developer is important to look for and find support. This clarification is necessary because some project management tools and even most of the Related Work reported in this study use *Issues* as samples for tests and development of their work and this dissertation work with project tasks from the project plan (which can also be a issue or bug a developer worked on as part of the project).

2.3 Q&A Websites as Knowledge Repositories

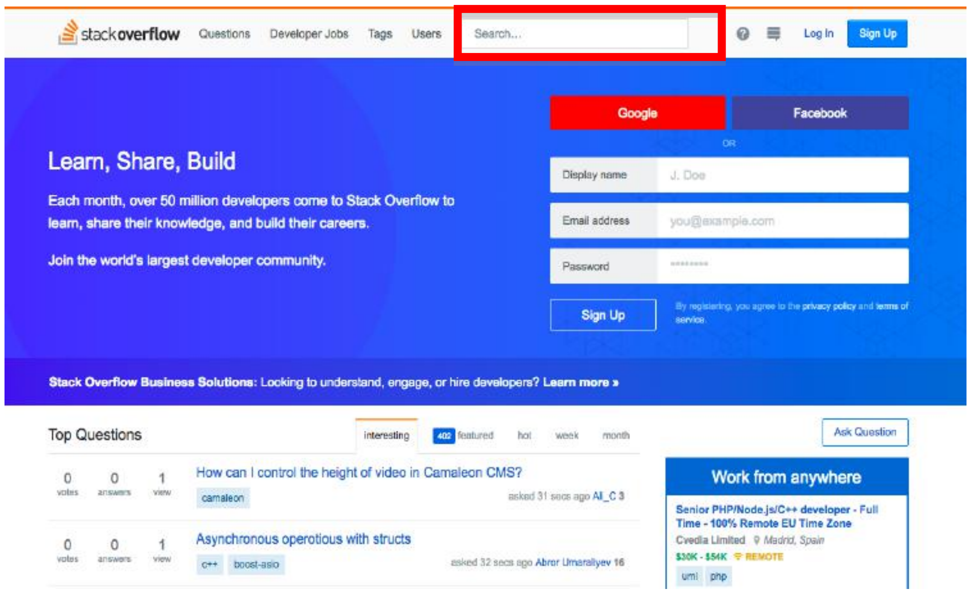
Question and Answer (Q&A) websites are online communities where people with specific interests post questions and other users attempt to provide answers (JAFFEE, 2005). There has been an evolution of such Q&A sites and often participants in a site have substantial expertise in specific domain areas, thereby increasing the value of the content. Q&A websites originally pursued to contribute to specific questions asked, however within time, there has been a change on how these questions are asked and answered. Instead of giving a very good answer to that specific question, a more community-driven focus has been created. Within this perspective change, larger audiences benefit from questions asked, as well as their solutions (ANDERSON *et al.*, 2012). Given this value for some Q&A sites and with the addition of mechanisms that monitor the quality of the questions and the answers, these sources of knowledge have become significant assets in supporting professional use. The work of MAMYKINA *et al.* (2011) investigated question-answering sites, such as Java Forum and Yahoo! Answers, the Korean Q&A site KiN, Slash(dot) and Stack Overflow.

Stack Overflow[1] is a Q&A website considered to be a significant knowledge base for highly technical software development (ANDERSON *et al.*, 2012) (YANG *et al.*, 2016) (PONZANELLI *et al.*, 2013) (EL-KORANY, 2013) (CORREA & SUREKA, 2013). Stack Overflow has over 90% of answered posts. Research states posts are answered in around 11 minutes, and users report Stack Overflow has become their primary resource when solving programming issues (MAMYKINA *et al.*, 2011). Although there are other important sources of knowledge for software developers, such as product manuals and guides, maturity models, Wikis and even a software developer's own experience (tacit knowledge), as consequence of its importance and extensive use, Stack Overflow is considered the knowledge source in this research. Other authors also considered Stack Overflow as the tool for software development knowledge support in their researches (CORREA & SUREKA, 2013; PONZANELLI et al., 2013; PONZANELLI et al., 2014; WANG et al., 2014; KOCHHAR, 2016).

2.3.1 Stack Overflow

Stack Overflow is a Q&A community of professional software developers and development enthusiasts. Over 50 million accesses each month gives Stack Overflow an importance of its own. It has become a valuable source of information for developers as well as a recruiting tool for companies. Companies recruiters analyze how engaged and the participation of potential employees, and use this information as another tool to gather the person's expertise information. Stack Overflow has questions and answers on a wide range of topics in software programming. Stack Overflow has been subject of conference mining challenges, as in MSR[2] in 2015. Stack Overflow's homepage can be viewed in Figure 2.6.

[1] stackoverflow.com
[2] http://2015.msrconf.org/challenge.php



Learn, Share, Build

Each month, over 50 million developers come to Stack Overflow to learn, share their knowledge, and build their careers.

Join the world's largest developer community.

By registering, you agree to the privacy policy and terms of service.

Stack Overflow Business Solutions: Looking to understand, engage, or hire developers? Learn more »

0 votes	0 answers	1 view	How can I control the height of video in Camaleon CMS? — asked 31 secs ago Al_C 3
0 votes	0 answers	1 view	Asynchronous operotious with structs — asked 32 secs ago Abror Umarallyev 16

Work from anywhere

Senior PHP/Node.js/C++ developer - Full Time - 100% Remote EU Time Zone

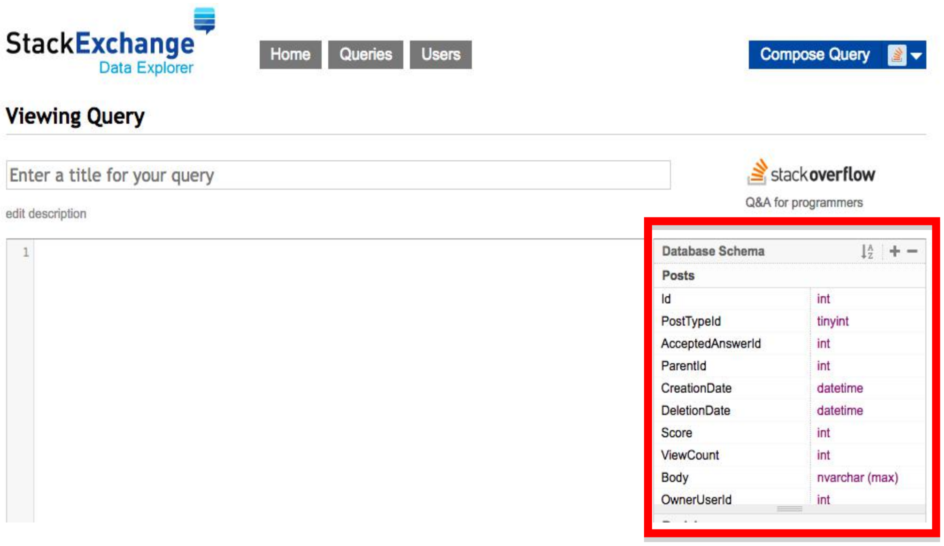
Cvedia Limited Madrid, Spain

$30K - $54K REMOTE

Figure 2.6: Stack Overflow home page.

In order to retrieve information, users can use the search box on the home page, presented in Figure 2.6 surrounded by a red box. What users search for are, for example, solutions for software errors or how to implement methods and pieces of code on certain programming languages, or how to use APIs. Stack Overflow was conceived to provide high quality information in an accessible way. Stack Overflow's creators, Joel Spolsky and Jeff Atwood, envisioned a combination of a collaborative tool, with ranked feedbacks and moderated content. There was a planned incorporation of strategies, thought by the site's creators, so they could guarantee that Stack Overflow would be used solely with a software programming focus, and that its content was actually used by software developers. These strategies were the incorporation of mechanisms such as Votes and Tags for Stack Overflow posts, among other strategies. Every user can vote if a question or answer is really useful, and these votes are summed up and displayed. Users are encouraged to vote on questions and/or answers they found most useful. Answers are organized on the post page by the number of votes received and question's rates are shown when they're queried and retrieved. This ensures highest quality answers are presented at the top, identifying the most likely solution among all available ones easily. The reputation of users with most voted answers increases, since they earn points when other users vote on their questions or answers. To optimize retrieval of questions or answers, they are associated with a variety of tags. Examples of tags are: *java, Android, jQuery*, and other technology specifications and defined fields that help programmers search directly for their expertise or area of interest (PONZANELLI *et al.*, 2013). Users define tags at their discretion.

Other than searching the homepage, there are APIs available that enables access to Stack Overflow data. One example is Stack Exchange API[3], which is a set of automatic components that allows direct access to data through other applications. All applications willing to access Stack Overflow's data have to follow the Terms of Use from Stack Exchange[4] and use the API methods during development. Stack Exchange community, which is a community of Q&A websites, also provides a dump with all website content. This dump is available online[5] and any person can download it. Another example of data access provided is the Stack Exchange Data Explorer, a web page interface that allows users to directly query Stack Overflow data using SQL syntax. To support users during query build, a database description in form of a list is provided. Figure 2.7 shows the query mechanism interface.



Database Schema

Posts

Id	int
PostTypeId	tinyint
AcceptedAnswerId	int
ParentId	int
CreationDate	datetime
DeletionDate	datetime
Score	int
ViewCount	int
Body	nvarchar (max)
OwnerUserId	int

Figure 2.7: Stack Exchange Data Explorer web page interface.

Through the database schema by Stack Overflow Query webpage, it was possible to create a partial model of what could be the Stack Overflow entities and its relations. We transformed the provided list (marked by the red box in Figure 2.7) into a relation-entity model. The model is illustrated in Figure 2.8. With the help of this model,

[3] https://api.stackexchange.com/docs

[4] https://stackexchange.com/legal/api-terms-of-use

[5] https://archive.org/details/stackexchange

it was possible to understand Stack Overflow’s structure and the domain this research is willing to support.

With Stack Overflow’s entities list provided by Data Explorer web page, and the visualization facilitated by the model, it was then possible to perceive Questions and Answers as types of a Stack Overflow Post. Figure 2.9 and Figure 2.10 present what comprises a Stack Overflow post, in a user view. Each Stack Overflow post comprises one question asked by a user, and one or more answers answered by one or several users. One post can also receive comments from users. Each question has a title and a description. The number in the left side of the image presents the votes for questions and answers, with arrows on top and below the number. These votes are transformed into scores for the authors of the content. Each time a user clicks an arrow, it counts as one vote.

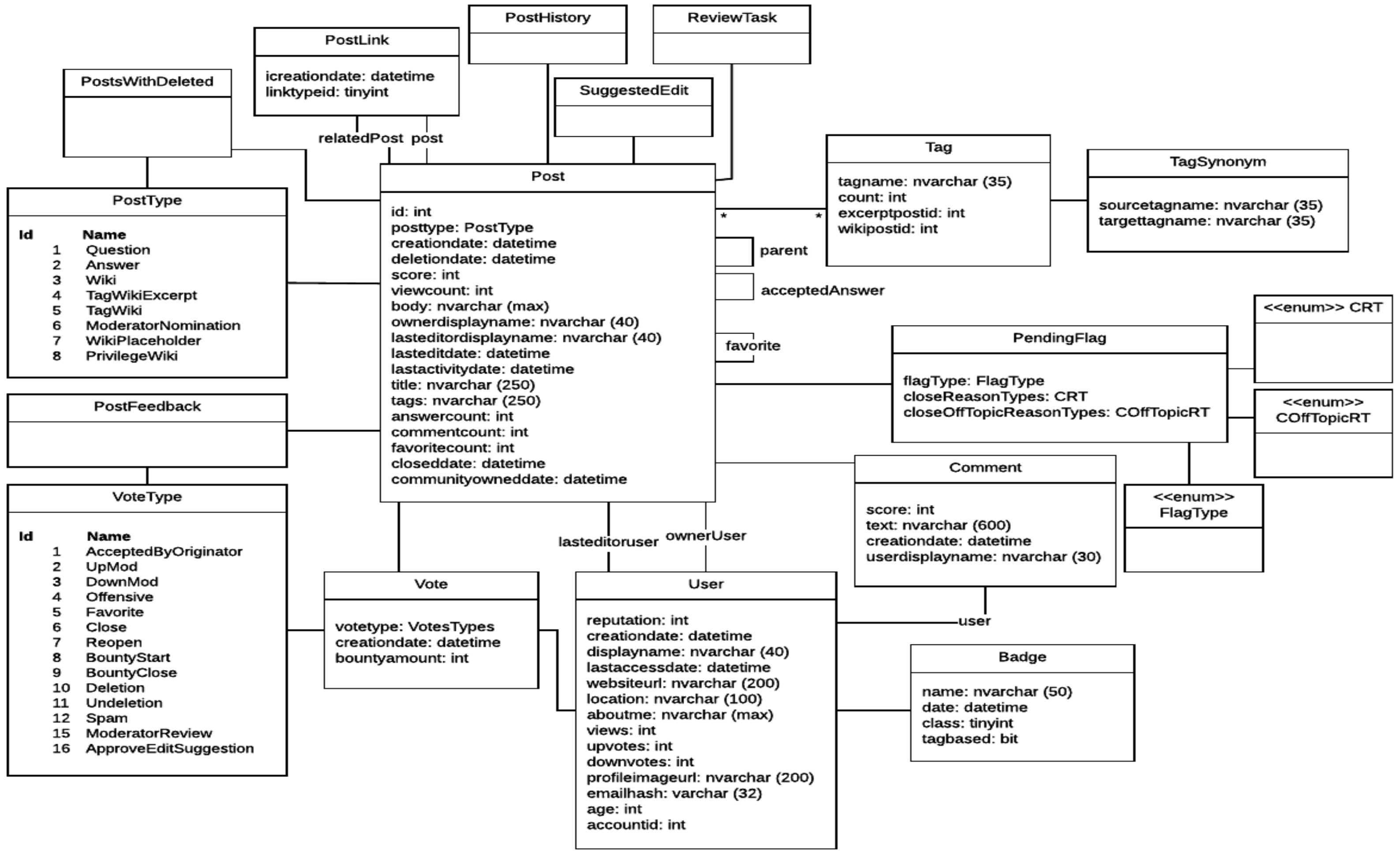
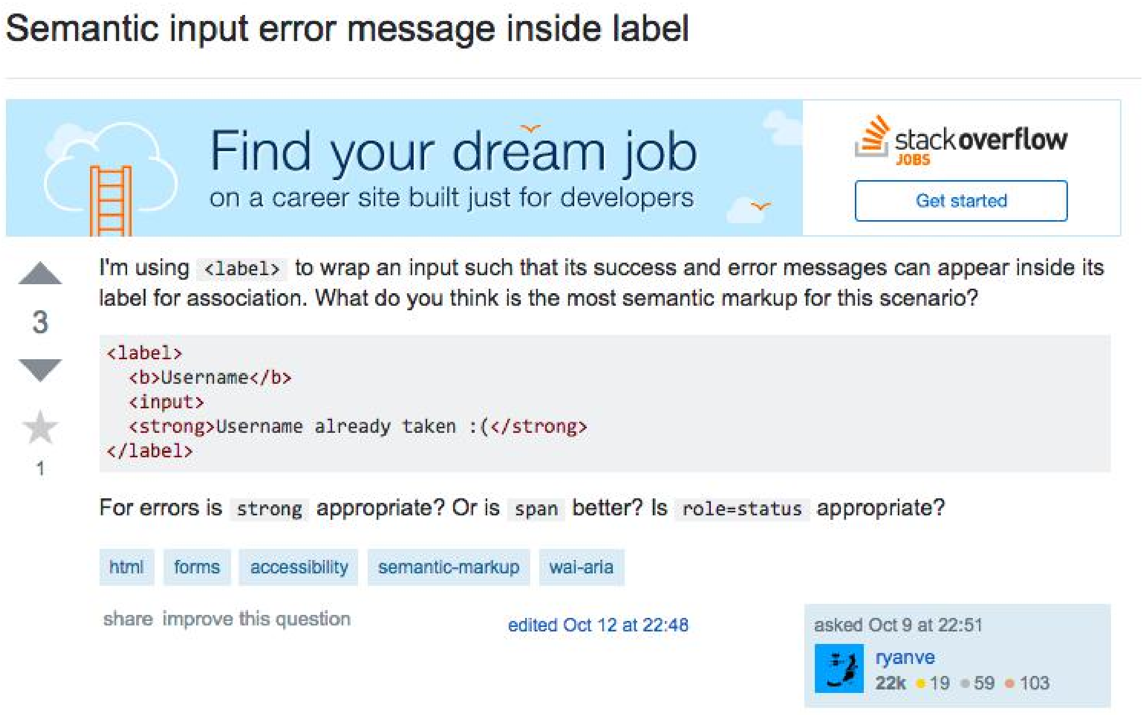


Figure 2.8: Stack Overflow database domain model.

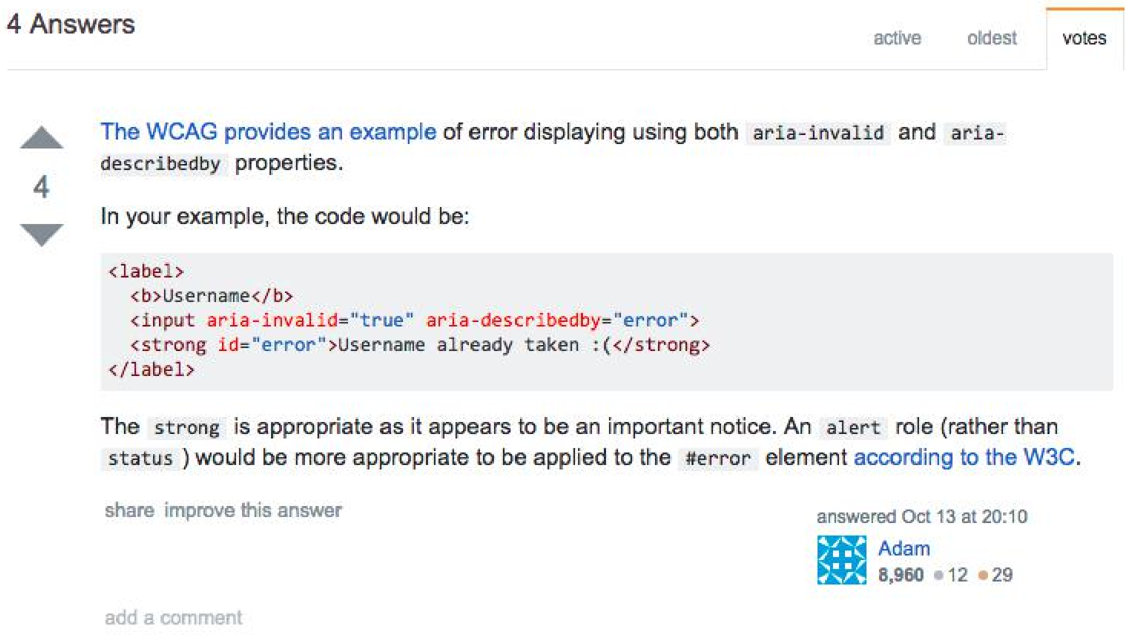

Semantic input error message inside label

I'm using `<label>` to wrap an input such that its success and error messages can appear inside its label for association. What do you think is the most semantic markup for this scenario?

```
<label>
  <b>Username</b>
  <input>
  <strong>Username already taken :(</strong>
</label>
```

For errors is `strong` appropriate? Or is `span` better? Is `role=status` appropriate?

Figure 2.9: Stack Overflow question.


4 Answers

The WCAG provides an example of error displaying using both `aria-invalid` and `aria-describedby` properties.

In your example, the code would be:

```
<label>
  <b>Username</b>
  <input aria-invalid="true" aria-describedby="error">
  <strong id="error">Username already taken :(</strong>
</label>
```

The `strong` is appropriate as it appears to be an important notice. An `alert` role (rather than `status`) would be more appropriate to be applied to the `#error` element according to the W3C.

Figure 2.10: Stack Overflow answer.

2.4 Data Mining

Data Mining is the discovery of "models" for data (D. ULLMAN & RAJARAMAN, 2013). These "models" can be defined as one of several things:

In statistical modeling, the construction of these data mining models aimed at trying to extract information that was not supported by the data;

In Machine Learning, which sometimes is wrongly used as synonym for data mining, data is used as training sets to algorithms. This approach is proved to be useful when there is no expectation regarding the results; in situations where it is possible to describe goals more directly, machine learning has not proved successful to mining data.

In a computational approach, where computer scientists look at data mining as an algorithm problem, the model is simply an answer to a query about it.

Most other approaches can be described as summaries of data or extracting most prominent features from data. Regarding features, typical feature extraction model looks for extreme examples of a phenomenon and represents data by these examples. If there is a complex relationship between objects, this relationship can be represented by finding statistical dependencies among these objects and using those in representing the connections among objects. Two large-scale data extraction features are frequent sets and **similar items**.

For similar items specifically, the goal is to find pairs of sets that have elements in common. An example is when an online store analyzes customer's profiles and what they have in common in order to recommend products similar customers have bought. This is called "collaborative filtering". The notion of similarity regarding finding sets with a relatively large intersection is called *Jaccard Similarity* (D. ULLMAN & RAJARAMAN, 2013)*. Jaccard similarity addresses an important class of problems, which is* finding textually similar documents in a large corpus. Note that *Jaccard* similarity is a character-level similarity, meaning it does not extract "similar meaning". More details in this similarity algorithm are described further in next sections, where we approach the similarity metrics implemented in this dissertation, specifically.

2.4.1 Recommendation Systems for Software Engineering

Although this dissertation does not propose a recommender tool, there are conceptual aspects shared by this proposal and recommender systems. This is the reason why this topic was included in this dissertation.

In the sense of what recommendation systems are, there is a definition proposed by the organizers of the ACM International Conference on Recommender Systems (RecSys 09; http://recsys.acm.org/2009), which is:

> *"[Recommendation] systems are software applications that aim to support users in their decision-making while interacting with large information spaces. They recommend items of interest to users based on preferences they have expressed, either explicitly or implicitly."*

When narrowing the approach to Recommendation Systems for Software Engineering (RSSE), there is the definition from Robillard et al. (ROBILLARD *et al.*, 2010) stating:

> *"An RSSE is a software application that provides information items estimated to be valuable for a software engineering task in a given context."*

Recommender Systems for Software Engineering (RSSE) are software applications that provide valuable information in support of a software engineering task (ROBILLARD *et al.*, 2010). RSSEs are applications that foresee user responsiveness to given options (D. ULLMAN & RAJARAMAN, 2013). The architecture of a typical RSSE comprises at least three basic characteristics: a data-collection mechanism, recommender engine and an interface for a user to give and receive recommendation inputs and outputs. Recommender systems deal with problems such as the cold-start problem, which addresses the lack of content to be recommended on initial states of the system, when there are little or no available data. This definition arises as the definition that has been used as standard for RSSEs (GASPARIC *et al.*, 2017).

In addition to presenting this definition, the work from Robillard et al. (ROBILLARD *et al.*, 2010) is an overview of the RSSE field, focused on presenting what good RSSE do for developers and also presented limitations for RSSE, mentioning the fact that most systems recommend source code, resulting in a lack of variety in the output type produced.

A Systematic Literature Review (SLR) published in 2016 (GASPARIC & JANES, 2016) (ANTUNES *et al.*, 2012) focused on presenting functionalities of RSSE systems, as well as research gaps and possible research directions. Results pointed that the current recommendation tools focus on source code, mainly supporting reuse actions and debugging. The papers mentioned by the SLR focus on:

(1) Task context as artifacts under development

(WARR & ROBILLARD, 2007) (KOBAYASHI *et al.*, 2012)

(2) Suggestion of code modification made by mining a database that has recorded the diffs of prior changes on code

(DOTZLER *et al.*, 2012)

(3) Recommendation of code reuse

(HOLMES *et al.*, 2009)

(4) Components recommendation regarding users browse history

(HOLMES *et al.*, 2009) (ICHII *et al.*, 2009)

(5) Where to look for information while debugging source code

(PIORKOWSKI *et al.*, 2012)

(6) Recommendation of artifacts and recommendation of tools

(ČUBRANIĆ *et al.*, 2004) (VIRIYAKATTIYAPORN & MURPHY, 2009).

These studies use a variety of strategies in order to accomplish recommendation. Considering the focus of this work is to work with an existing database of project tasks and use the text similarity of these tasks in order to discover how similar the tasks are and provide the Stack Overflow Post to the most similar ones, similarity measures and tools to implement these metrics will be presented in over the next sections.

2.4.2 Similarity Metrics

Similarity metrics are measures of how much two objects are alike. In other words, within the mining scenario, similarity measure is a distance with dimensions representing features of the compared objects. The smaller the distance, the higher is the degree of similarity between the objects. Objects can also usually be measured as the inverse, and are called distance metrics. Similarities are often measured in a range of 0 to 1. Similarity will be 1 if the objects are identical and will be 0 if the objects are completely different. There are some similarity metrics that are massively used by data scientists, for example the Euclidean distance, Manhattan distance, *Cosine* similarity, *Jaccard* similarity and others. Each of these metrics have their specifics, and as they derive from statistics, most of them have implementations in most programming languages today.

Recommendation systems mostly rely on Collaborative Filtering, which is the method of making automatic predictions - filtering - about the interests of a user. This method collects preferences or taste information from many users collaboratively. However, when no such data are available, or even if they are available but the amount of data is not large enough, other approaches are necessary. That is called the cold-start problem, which in other words is the lack of information to be recommended if there is no data to perform comparisons or relations between existing sets. One of the strategies used to overcome the cold-start problem, are approaches that use distance-based similarity calculations in multi-dimensional spaces, known as content-based recommendation. Some of the popularly distance-based similarity calculations used (YUNG-SHEN LIN *et al.*, 2014) for comparing two documents (text fragments) are: Cosine (HAN & MINING, 2000), *Jaccard* (TAN *et al.*, 2006) (D. ULLMAN & RAJARAMAN, 2013) and *Levenshtein* (*LEVENSHTEIN*, 1966). To explain how these metrics work, we will present details on *Levenshtein* and *Jaccard* algorithms, the algorithms implemented in this dissertation.

2.4.2.1 *Levenshtein*

A work that intended to develop binary codes that were self-correcting, strings searches were developed and the *Levenshtein* distance was presented (*LEVENSHTEIN*, 1966). The name, *Levenshtein*, comes after Vladimir *Levenshtein*, who presented this technique in the year of 1965. *Levenshtein* distance suits the needs to look for common features between texts and is an option for the cold-start problem. This distance counts the steps needed to transform one word into the other, meaning it checks the minimum effort required to transform one string into the other, by counting the number of insertions, deletions and substitutions. To get a similarity index, this distance is then calculated by proportionally comparing it to the size of the word (how many characters the word has). This similarity index is then inversely proportional to the distance. A distance of 0 implies maximum similarity. *Levenshtein* similarity is presented using the formula presented in Equation 2.1. LabelSimilarity function is the result of the formula. LevenshteinDistance represents the steps to transform label1 into label2. *Length* is the operation that calculates the size of the label, how many characters each word has.

$$LabelSimilarity(Label1, Label2) = 1 - \frac{Levenshtein\ Distance(label1, label2)}{length(label1) + length(label2)}$$

Equation 2.1: *Levenshtein* Distance – strings similarity metrics.

For example, the *Levenshtein* distance between the labels "kitten" and "kitchen" is 2, because:

- Step 1: kitten -> kitcen (substitution of “t” for “c”)
- Step 2: kitcen -> kitchen (addition of “h”)

Applying the formula to get the similarity index, once the word "kitten" has six characters and the word "kitchen" has seven characters:

$$LabelSimilarity(kitten\ , kitchen\ ) = 1 - \frac{2}{6+7}$$

Equation 2.2: Formula application example.

$$LabelSimilarity(kitten\ , kitchen\ ) = 0.84$$

Equation 2.3: Formula application example result.

This means the words *kitten* and *kitchen*, according to the *Levenshtein* distance, are 84% similar. A perfect match would get score 1 as a completely different word comparison would get score 0.

2.4.2.2 Jaccard

Jaccard index, also known as *Jaccard* similarity coefficient, is another statistic metric used to compare finite sets of data. This metric is used to compare the similarity and diversity of sample sets. It uses the ratio of the intersecting set to the union set as the measure of similarity. Thus, it equals to zero if there are no intersecting elements and equals to one if all elements intersect. The result of the metric is an index within a range of 0 and 1. The equations of *Jaccard* similarity and *Jaccard* distance are presented below in Equation 2.4 and Equation 2.5.

$$JaccardSim(item1, item2) = \frac{|item1 \cap item2|}{|item1 \cup item2|}$$

Equation 2.4: *Jaccard* similarity.

$$JaccardDistance(item1, item2) = 1 - \frac{|item1 \cap item2|}{|item1 \cup item2|}$$

Equation 2.5: *Jaccard* distance.

In order to exemplify, supposing there is the following set of items, where 8 is the total of items, and 3 items are in the intersection, the *Jaccard*Sim = 3/8 = 0.4 and the *Jaccard*Distance = 5/8 = 0.6. The set of the example is presented in Figure 2.11.

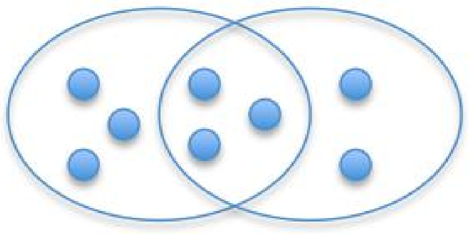


Figure 2.11: Jaccard set.

Using documents as examples, supposing there are the three following documents:

- Document1: “Word2”, “Word3”, “Word4”, “Word2”.
- Document2: “Word1”, “Word5”, “Word4”, “Word2”.
- Document3: “Word1”

Comparing *Document1* and *Document2,* as the total number of unique words of both documents is 5, and the number of shared words between the documents is 2, gives a *Jaccard* similarity of 2/5 = 0.4. Comparing *Document1* and *Document1,* gives a *Jaccard* similarity of 1. Comparing *Document1 and Document3,* the result is 0, because they share no similarities between the documents. As stated by (TAN *et al.*, 2006), this is the most often used metric for document comparison.

Two metric solutions were described. Both of these algorithms were used in different steps of this dissertation. The first one, *Levenshtein*, was used during an implementation of the proposal that was not used in the end, but served well as a maturing step, and the second one, *Jaccard*, was used in the similarity process presented as the proposal in this dissertation. There are automated tools that implement these algorithms when comparing documents. Tools and examples of tools are described over the next sections.

2.4.2.3 Tools for Text Similarity Algorithm Implementations

Automatic texting processing through tools is necessary and mainly used in order to extract information from the large amount of data generated everyday (RAQUEL FONSECA & EDGAR CASASOLA, 2017). To process text data and use the results as inputs for decisions is important in order to make predictions, extract information for decisions over products or for sentiment analysis Process text data can also promote the identification and classification of information in natural language texts. Text similarity metrics retrieves a number (index) that means how much two texts (or documents) are similar, mostly in a range of 0 to 1, 0 meaning not similar and 1 meaning the compared objects are equal. Tools can automate this process, as well as also perform extra evaluation steps for the metrics provided. We describe two tools that were used during the development of this work. The tools are Dandelion and *RapidMiner*. They are essentially different in their goals and application, and the differences are outlined in each section dedicated to each tool. How they were used in this work will be presented further in the development of this work, more precisely in Chapter 4.

2.4.2.4 Dandelion

Text Similarity tool Dandelion[6] provides semantic text comparisons as a service. Spaziodati[7], an Italian company that develops a range of solutions for big data, is responsible for developing and maintaining Dandelion. This tool provides APIs that can easily be integrated into any application solution through JSON calls, providing convenient usage. This tool was object of study and comparison to other tools, and some of its features performed better when comparing to others (RAQUEL FONSECA & EDGAR CASASOLA, 2017).

Dandelion Text Similarity API is based both on syntactic and semantic features. With respect to the semantic ones, it is able to identify semantic relationships by comparing the entities found in the two compared texts. Wikipedia, for example, if considered as a graph of semantic relationships, defined by the page links, then entities are more similar if they are "close" in this graph. A comparison of a Bag-Of-

[6] dandelion.eu
[7] spaziodati.eu

Words representation of two texts is used in the syntactic similarity retrieval. Presenting an example, the words “blue sky” and “sky blue” would be a perfect match, as presented in Figure 2.12.

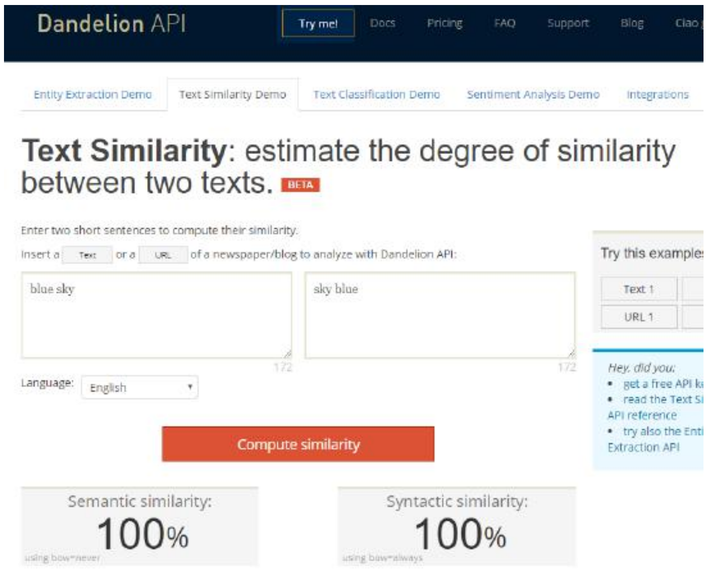
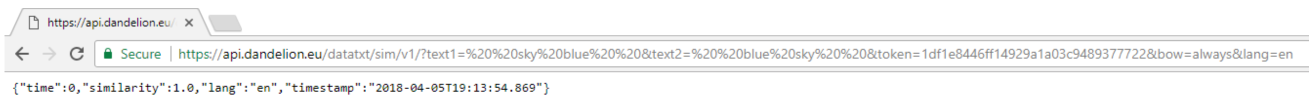

Dandelion API
Try me!
Docs
Pricing
FAQ
Support
Blog
Entity Extraction Demo
Text Similarity Demo
Text Classification Demo
Sentiment Analysis Demo
Integrations
Text Similarity: estimate the degree of similarity between two texts. BETA
Enter two short sentences to compute their similarity.
Insert a Text or a URL of a newspaper/blog to analyze with Dandelion API:
blue sky
sky blue
Language: English
Compute similarity
Try this example
Text 1
URL 1
Hey, did you:
get a free API key
read the Text Similarity API reference
try also the Entity Extraction API
Semantic similarity: 100%
using bow=never
Syntactic similarity: 100%
using bow=always

Figure 2.12: Dandelion text similarity webpage.

Besides the interface presented in Figure 2.12, this tool also provides an API that allows external application calls in order to get the comparison semantic index generated between sentences. This API also works by submitting a formatted URL into a browser or sending as a POST method8. This will generate a JSON response, which includes among other information, the similarity index between the compared sentences. The parameters that build this URL are described in Table 2.1.

Figure 2.13 presents an example of an URL submitted to a browser and how the response is returned. The parameters *text1* and *text2* used in the example are “blue sky” and “sky blue”. The parameter *lang* is set for English language and the token is the *token* for the author’s account. The URL submitted was https://api.dandelion.eu/datatxt/sim/v1/?text1=%20%20sky%20blue%20%20&text2=%20%20blue%20sky%20%20&token=1df1e8446ff14929a1a03c9489377722&bow=always&lang=en and the response received was {"time":0,"similarity":1.0,"lang":"en","timestamp":"2018-04-05T19:13:54.869"}.

[8] HTTP method for request-response between servers and clients.



```
https://api.dandelion.eu/datatxt/sim/v1/?text1=%20%20sky%20blue%20%20&text2=%20%20blue%20sky%20%20&token=1df1e8446ff14929a1a03c9489377722&bow=always&lang=en
{"time":0,"similarity":1.0,"lang":"en","timestamp":"2018-04-05T19:13:54.869"}
```

Figure 2.13: Dandelion browser submission example.

Table 2.1: Dandelion parameters. [9]

Parameter	Description	Type
text1|url1|html1|html_fragment1	These parameters define how you send to the Text Similarity API the first text you want to compare.	string
text2|url2|html2|html_fragment2	These parameters define how you send to the Text Similarity API the second text you want to compare, in the same way as the text1|url1|html1|html_fragment1 parameters.	string
lang	The language of the texts to be compared; currently English, French, German, Italian, Portuguese, Russian and Spanish are supported.	string
bow	The Text Similarity API normally uses a semantic algorithm for computing similarity of texts. It is possible, however, to use a more classical syntactic algorithm where the semantic one fails. This can be done with this parameter. ● "never" uses always the semantic algorithm; ● "both_empty" uses the syntactic algorithm if both the two texts have no semantic information; ● "one_empty" uses the syntactic algorithm if at least one of the two inputs have no semantic information; ● "always" uses always the syntactic algorithm.	string
token=<YOUR_TOKEN>	API token for each email account.	string

The response received has the following values:

- Time: approximately how long the comparison execution took.
- Similarity: similarity index computed. For the example, the similarity is 1.0, meaning the sentences are 100% similar.
- Lang: language used for comparison.
- Timestamp: timestamp of execution.

[9] Source: https://dandelion.eu/docs/api/datatxt/sim/v1/

Analogously to the *Levenshtein* distance, Dandelion's API solution compares two strings and returns a similarity index that informs the percentage of similarity between two text strings. Comparing one Task's context string to another Task's context string generates a similarity index. This tool was used during this study based on the documentation from other studies that use text similarity index comparisons. One research work point Dandelion API as one of the best solutions for text disambiguation (FILIPIAK et al., 2006). Another work (RAQUEL FONSECA & EDGAR CASASOLA, 2017) extracts entity recognition from short texts and compares Dandelion API with another solution called *AlchemyAPI*. Each algorithm was compared against the other in order to calculate their precision and recall. Dandelion API got better results with the given data. Dandelion API was also considered to perform well in another work that used the Name Entity Recognition feature (BAKER & VERSTOCKT, 2017). Based on the proposal needs, which were to compare text strings and obtain a numerical index that states the percentage of similarity between the compared strings, automatically and with rich documentation, Dandelion API was chosen during the implementation phase in this dissertation, as detailed in Section 1.3.

Dandelion is a very powerful tool, but as it belongs to a company, the similarity retrieval mechanisms are not open to public. Also, we needed another tool or extra implementation to evaluate the generated data. These were the reasons this solution was not used in this work's proposal. We are not aware of the algorithm (or combination of algorithms) used and how the similarity is extracted, and to the best of our knowledge, there are no documents available with this information. This hampers the reproducibility characteristic of this dissertation's proposal. Next, *RapidMiner*, a powerful customizable data mining tool is presented.

2.4.2.5 *RapidMiner*

Data mining statistical computing tools are created and can be used on a large amount of data types and analysis. These tools allow creating and monitoring data mining processes, helping whoever is implementing these processes to build as well as to manage them. A complete and very powerful data mining tool is *RapidMiner*[10]. The Artificial Intelligence Group of Katharina Morik at the Dortmund University of Technology developed *RapidMiner*, in 2001 (SCHLITTER *et al.*, 2013). The project was named YALE and in 2007 it changed to *RapidMiner*. The software is hosted by *SourceForge* and is offered free of charge as a Community Edition released under the

[10] rapidminer.com

GNU AGPL. *RapidMiner* is written in Java and runs user created processes. A process is an XML-File generated by the user and contains a sequence of tasks, which are represented by operators.

RapidMiner is used in research, education, prototyping, development and industrial applications. It is open source and has a lot of features available as operators, such as data cleaning, transformation, optimization, evaluation and visualization. *RapidMiner* also has a wide range of algorithms implementations, to classify, cluster and perform other analytical processes. An important characteristic of *RapidMiner* is the coding-free implementation, meaning there is no need to code to use the tool, although users can add their own code. The tool comes with a broad number of native operators, but third-party extensions are also available (SANTHANAKUMAR & COLUMBUS, 2015).

RapidMiner uses operators (there are around 500 built in operators), which act as plug-and-play programming-free features, and together are responsible for a data analysis process. Each operator performs a specific task on data, e.g., loading and storing data, transforming data, or inferring a model on data. Placing a combination of operators in a canvas and informing their input and output compose a process in *RapidMiner*. For more than 10 years, *RapidMiner* has been extended and many extensions were developed, making this tool an excellent option as a tool for data mining and analytics. This tool is very much used in academic courses and universities all over the world, as well as for industry solutions, as already mentioned (SCHLITTER *et al.*, 2013). An example of an implementation of a process is presented in Figure 2.14.

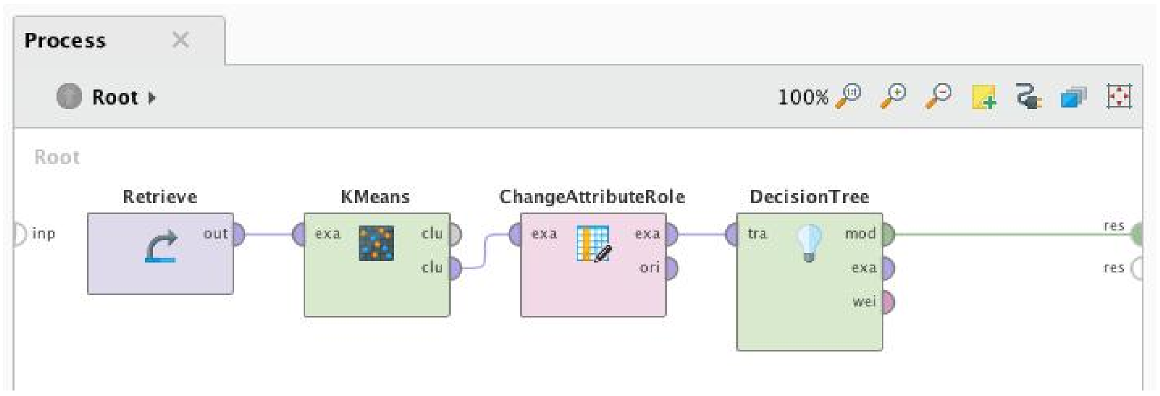


Figure 2.14: *RapidMiner* Process Example.

In this example, there are four operators implementing a Cluster Classification process. The first operator *Retrieve*, imports the data into the process. The second operator, *KMeans*, performs clustering tasks using the K-Means algorithm. Then, the *ChangeAttributeRole* operator changes the role of one or more attributes of the

KMeans operator output, and the *DecisionTree* operator generates a decision tree model, which can be used for classification and regression. All *RapidMiner* operator are documented in *RapidMiner* website docs.rapidminer.com. *RapidMiner* also supports performance evaluations with a wide variety of metrics implementations. There are implementations of a variety of similarity and distance metrics in *RapidMiner*, such as *Dice*, *Jaccard*, Euclidean and many others.

RapidMiner is used in this dissertation to implement the process created for the proposal, as well as the evaluation. The application of *RapidMiner* in this dissertation is described in Chapter 4 and the evaluation in Chapter 5.

2.5 Conclusion

This chapter introduces concepts on Software Processes, Question and Answer services, data mining, recommender systems and also, similarity metrics and tools. Software Processes are used during the software development workflow. Software development is described as knowledge-intensive, collaborative and technology-dependent, and because of that, knowledge is intensely present during software development. Question and Answer websites are important sources of knowledge, and software developers massively use one famous Q&A website, Stack Overflow, when external knowledge is needed. The use of a mechanism to store and further associate this acquired knowledge from Stack Overflow within the software project can bring benefits to software developers. Software developers invest time in searching for what piece of information (or pieces) should be used to help them during software development. By extracting similarities from project tasks that have a curated Stack Overflow Post associated to them can be a way to leverage software development and furthering the area of software development. Next Chapter presents a Systematic Mapping Study performed in order to obtain the state-of-the-art of how academia has been researching and presenting solutions to help software developers find, select and reuse Stack Overflow Posts in software projects.

3 Related Work

This Chapter presents the process and results of the Systematic Mapping Study conducted to assemble approaches that associates the software development projects with Stack Overflow, aiming at understanding how this area has developed and what are the current researches in the field. The findings expose the current state of the research in literature and were used to support the development of this work.

3.1 Introduction

This chapter presents a Systematic Mapping Study performed to investigate the state-of-the-art of the existing strategies that associate Stack Overflow Posts to the development environment. This fulfills this dissertation's first objective, layed out in Section 1.2. This study is aligned with the objectives proposed in Section 1.2 and represents step #4 of the methodology of this dissertation, described in Section 1.3. This Chapter is structured with the following sections: the current section presents an introduction on the importance of researching the association of Stack Overflow Posts to software development, then Section 3.2 presents the objectives of the Systematic Mapping Study. Section 3.3 presents the steps conducted during planning and Section 3.4 the execution of the Systematic Mapping Study. The analysis steps of the papers retrieved are presented in Section 3.5. The remainder Sections 3.6 and 3.7 present the threats to the validity of the systematic mapping study and the conclusions, respectively.

Stack Overflow is widely used by developers when help is needed during software development (MAMYKINA *et al.*, 2011). During this search, developers have to choose among a list of items, find a suitable solution for the problem they have, implement the solution and test it, to guarantee the chosen solution works for the problem the developer is facing. Time is invested in this *curation* task, and it depends on the tacit knowledge of developers. Due to the iterative characteristic of software development projects, this search can even occur more than once.

In order to aid software development by helping developers assess Stack Overflow Posts, several solutions were proposed. To have a broad perspective of what researchers have proposed and map the proposed strategies, a Systematic Mapping Study has been conducted. In this study, we used the protocol by (PETERSEN *et al.*, 2015), and also implemented strategies to increase the reliability of the study, such as the addition of control studies and the participation of a second person during the planning step of the study, when the search string has been assembled.

3.2 Search Objectives

The Systematic Mapping Study (SMS) approach is a common study to obtain evidence on a particular subject and provides categorized results that have been published in the area of research (PETERSEN *et al.*, 2015) (BARROS-JUSTO *et al.*, 2018). The SMS reported in this chapter was conducted in order to gather the state-of-the-art in the literature regarding the objectives presented in Section 1.2. The current section presents the protocol used to select studies for this research. This protocol is the process of building search strings and the definition of a search scope. These procedures were conducted to have a comprehensive examination of what has been published on the specific topic of strategies to associate development project with Stack Overflow, acknowledging what are the current proposals, the input information, the output, the evaluation method and the results. For this study, we performed an automated search in Scopus, an online database that indexes several other scientific databases.

The protocol suggested by (PETERSEN *et al.*, 2015) also uses the GQM approach (VAN SOLINGEN *et al.*, 2002) in order to define the goals for the SMS. The search developed in this work has as goal the analysis of articles that aim at associating software development with Stack Overflow. It is possible to replicate this SMS, considering how it was organized into plan, execution and analysis steps. This organization also contributes to summarize the articles in integrating Stack Overflow into the software development. The goals of this SMS are presented below.

analyze *proposals that associate software development with Stack Overflow*
with the purpose of *characterizing*
regarding *strategy, input, output, evaluation methods, metrics and results*
from the point of view of *researchers*
in the context of *software development projects*

Once the goal of this Systematic Mapping Study is defined, next sections present the planning and execution steps of the SMS according to the established goal.

3.3 Planning

The planning step encompasses three different activities: (1) defining the research questions for the SMS, according to the objectives, and (2) using PICO to aid the search strings and from that, (3) creating a search string. According to the defined goal for this SMS, the following research question is presented:

What are the existing association strategies, input and output information, evaluation methods and results that propose the association of software development projects and Stack Overflow Posts?

This research question is necessary because it is important to identify existing studies in this area and create a summary and categorization of the information these papers present.

In order to organize and structure the search string based on the objective and research question, PICO approach was used - Population of interest, evaluated Intervention, Comparison, Outcomes (PAI *et al.*, 2004). Table 3.1 presents PICO description and goals.

Table 3.1: PICO (PAI *et al.*, 2004) for SMS.

(P) Population	Software Development
(I) Intervention	Research that aims at relating development project with stack overflow
(C) Comparison	Not applicable, as the goal of this study is to identify the state-of-the-art, not compare.
(O) Outcomes	Solutions that associate software development to Stack Overflow.

Attempting to improve the reliability of the search string, the research goals were presented to the Computer Science Library Liaison of the University of Waterloo in January 2018. With the author of this dissertation, this researcher defined what would be the search strings according to what understanding of the research objective. The search strings (and synonyms) is presented in Table 3.2.

Table 3.2: Search string – January 2018.

```
"stack overflow" OR "online community" OR wiki OR "web 2.0" OR "crowd sourcing" OR
"crowd sourced" OR "crowd knowledge" OR "knowledge base" OR "reference base" OR
"reference manual"
AND (recommend* OR suggest* OR propos* OR offer*)
AND ((context* AND (word* OR post*)) OR "text mining" OR "text based" OR "text
attributes" OR "natural language")
AND (((manag* OR organiz* OR plan* OR project* OR track*) AND (tool* OR program*
OR software OR app OR application* OR platform*)) OR "issue tracker” OR "issue
tracking")
```

For a second execution of this SMS, aiming at raising the confiability of the SMS and update with recently published articles, used the following search string presented in Table 3.3.

Table 3.3: Search string – September 2018.

```
"stack overflow” OR ”stackoverflow"
AND (issue OR  task  OR  context )
AND (recommend* OR suggest* OR associat*  OR  link  OR  integrat* )
AND ("software" OR  "software process"  OR  "develop*" )
```

The differences between the two strings are the number of synonyms used in the second string. Regarding the results, they were almost the same as reported by the first execution, with the following exceptions:

1) Fishtail (SAWADSKY & MURPHY, 2011) was included at first, but the second SMS didn’t consider this article because there are no reported results;
2) Articles published in 2018 were added in the second execution (SMS update);
3) The articles (total of 3) from Ponzanelli et al., all about the tool Seahawk, are represented by only one study after the SMS update, instead of three in the first SMS.

Scopus (http://www.scopus.com/) scientific database was the data source to retrieve the articles. This database indexes a wide amount of very important scientific databases. The Scopus database search string is presented below.

(TITLE-ABS-KEY ("stack overflow" OR "stackoverflow")
AND TITLE-ABS KEY (issue OR task OR context)
AND TITLE-ABS KEY (recommend OR suggest* OR associat* OR link OR integrat*)*
AND TITLE-ABS-KEY ("software" OR "software process" OR "develop"))*

For the selection of articles, inclusion and exclusion criteria were established to support the decision of which papers should be read or not. The selection steps performed are outlined below:

- Inclusion criteria:
    - Articles that deal with stack overflow or crowdsourcing communities where developers search for external support, presenting how it could be integrated to the software development workflow; OR
    - Articles dealing with recommendation/suggestion/link of software project artifacts; OR
    - Articles that address the use of issue or task information in order to find Stack Overflow posts;

- Exclusion criteria:
    - Articles not in the computer science area; OR
    - Articles whose research do not focus in Software Engineering; OR
    - Proposals that are not applied to software development; OR
    - Articles not written in English language; OR
    - Articles that do not state clearly the association strategy, input, output, evaluation method and results between Stack Overflow Posts and software development.

A group of known articles was defined; this being the most common strategy for evaluating the search (PETERSEN *et al.*, 2015). The group of known articles that should be mandatorily retrieved by the search strings is:

PONZANELLI, L. et al. Mining StackOverflow to turn the IDE into a self-confident programming prompter. **Proceedings of the 11th Working Conference on mining software repositories**, p. 102-111, May 31, 2014.

PONZANELLI, L.; BACCHELLI, A.; LANZA, M. Seahawk: stack overflow in the IDE. **Proceedings of the 2013 International Conference on software engineering**, p. 1295-1298, May 18, 2013.

CAMPOS, E.C.; SOUZA, L.B.L.; MAIA, M.D.A. Searching crowd knowledge to recommend solutions for API usage tasks. **Journal of Software: Evolution and Process**, Chichester, v. 28, n. 10, p. 863-892, Oct 2016.

CORREA, D.; SUREKA, A. Integrating Issue Tracking Systems with Community-Based Question and Answering Websites. **2013 22nd Australian Software Engineering Conference**, p. 88-96, 2013.

The selection of articles was conducted following three steps:

- **Step 1 - Preliminary selection of publications:** Running the search string in Scopus database will perform the preliminary selection of articles.
- **Step 2 - Selection of relevant publications – 1st filter:** For an initial selection of relevant articles, the title and abstracts of each returned article should be read and assessed according to the inclusion and exclusion criteria defined in the search planning. In some articles, only the reading of abstracts may not be sufficient, generating doubts as to whether or not they should be included. In this case the articles should be selected for full reading.
- **Step 3 - Selection of relevant publications - 2nd filter:** All articles selected in the 1st filter will be read and analyzed if they fit in according to the inclusion and exclusion criteria set.

3.4 Execution

According to the procedures for selecting articles defined in the planning of this structured search, the next step is then to perform the execution of the search string in each of the selected sources. Table 3.4 presents the results of the execution. This table presents the year of the publication and the amount of papers published each year, as well as the final count of papers retrieved. After applying all planned steps, Table 3.6 presents the related work of this dissertation, and Table 3.5 presents the selected papers after reading only Title and Abstract, which were fully read and some were excluded for not comprising the criteria.

After reading all titles and abstracts from the 69 articles, 25 articles were selected, according to the research objectives. Table 3.5 presents these articles, organized by year of publication, authors and the title of each article.

Table 3.4: Number of articles retrieved from database.

Publication Year	**Number of publications**
2018	18
2017	16
2016	11
2015	8
2014	5
2013	6
2012	3
2007	1
2006	1
Total	**69**

Table 3.5: Articles selected after reading Title and Abstract.

Year	Authors	Title
2018	Sirres R., Bissyand T.F., Kim D., Lo D., Klein J., Kim K., Traon Y.L.	Augmenting and structuring user queries to support efficient free-form code search
2018	Wei Q., Liu J., Chen J.	A method for recommending bug fixer using community Q&A information
2018	Greco C., Haden T., Damevski K.	StackInTheFlow: Behavior-driven recommendation system for stack overflow posts
2018	Wu D., Jing X.-Y., Chen H., Zhu X., Zhang H., Zuo M., Zi L., Zhu C.	Automatically answering API-related questions
2018	Etemadi V., Bushehrian O., Akbari R.	Association rule mining for finding usability problem patterns: A case study on StackOverflow
2018	Gao S., Xing Z., Ma Y., Ye D., Lin S.-W.	Enhancing Knowledge Sharing in Stack Overflow via Automatic External Web Resources Linking
2017	Liu X., Shen B., Zhong H., Zhu J.	EXPSOL: Recommending online threads for exception-related bug reports
2017	Fumin S., Xu W., Hailong S., Xudong L.	Recommendflow: Use topic model to automatically recommend stack Overflow Q&A in IDE
2016	Ponzanelli L., Bavota G., Di Penta M., Oliveto R., Lanza M.	Prompter: Turning the IDE into a self-confident programming assistant
2016	Campos E.C., de Souza L.B.L., Maia	Searching crowd knowledge to

	M.D.A.	recommend solutions for API usage tasks
2016	Sahu T.P., Nagwani N.K., Verma S.	An empirical analysis on reducing open source software development tasks using stack overflow
2015	Nagy C., Cleve A.	Mining Stack Overflow for discovering error patterns in SQL queries
2015	Zheng X.-L., Chen C.-C., Hung J.-L., He W., Hong F.-X., Lin Z.	A Hybrid Trust-Based Recommender System for Online Communities of Practice
2015	Amintabar V., Heydarnoori A., Ghafari M.	ExceptionTracer: A Solution Recommender for Exceptions in an Integrated Development Environment
2015	Wang W., Malik H., Godfrey M.W.	Recommending posts concerning API issues in developer Q&A sites
2015	Wang T., Yin G., Wang H., Yang C., Zou P.	Automatic knowledge sharing across communities: A case study on android issue tracker & stack overflow
2014	Ponzanelli L., Bavota G., Di Penta M., Oliveto R., Lanza M.	Mining stackoverflow to turn the IDE into a self-confident programming Prompter
2014	Ponzanelli L., Bavota G., Di Penta M., Oliveto R., Lanza M.	Prompter: A self-confident recommender system
2014	Rahman M.M., Yeasmin S., Roy C.K.	Towards a context-aware IDE-based meta search engine for recommendation about programming errors and exceptions
2013	Rahman M.M., Yeasmin S., Roy C.K.	An IDE-based context-Aware meta search engine
2013	Ponzanelli L., Bacchelli A., Lanza M.	Seahawk: Stack overflow in the IDE
2013	Correa D., Sureka A.	Integrating issue tracking systems with community-based question and answering websites
2013	Ponzanelli L., Bacchelli A., Lanza M.	Leveraging crowd knowledge for software comprehension and development
2012	Bacchelli A., Ponzanelli L., Lanza M.	Harnessing Stack Overflow for the IDE
2012	Zagalsky A., Barzilay O., Yehudai A.	Example overflow: Using social media for code recommendation

The article from (ČUBRANIĆ *et al.*) (2004), retrieved in the beginning of this research during *ad-hoc* literature reviews was considered an important asset regarding suggestion of development artifacts considering the project history. We consider important to include one well-referenced and important article in the field that might not only strengthen the importance of associating external information to the software development, but also because authors might use interesting strategies to associate other information rather than Stack Overflow Posts. The article included by manual search (PETERSEN *et al.*, 2015) technique was:

ČUBRANIĆ, D. et al. Learning from project history. **Proceedings of the 2004 ACM conference on computer supported cooperative work**, p. 82-91, Nov 6, 2004.

This article does not have Stack Overflow as a base for discovered knowledge association, nor uses text similarity. Instead, it suggests a new approach to aid software development by identifying actions performed during software development and suggesting artifacts to the developer that were identified by the creation of a history of iterations while the software project was running. This approach is very similar to the approach that will be proposed in this dissertation, as it mines existing project history to feature possible future suggestions.

All papers retrieved by the search string on Scopus and the paper added manually – were fully read (Step 3 of SMS), and considering the inclusion and exclusion criteria, 8 papers were finally selected. The selection/elimination process with the amount of papers is presented in Figure 3.1.

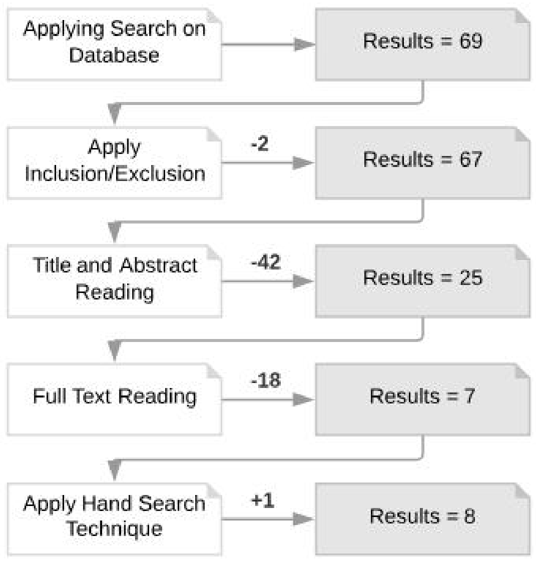

Applying Search on Database
Results = 69
Apply Inclusion/Exclusion
-2
Results = 67
Title and Abstract Reading
-42
Results = 25
Full Text Reading
-18
Results = 7
Apply Hand Search Technique
+1
Results = 8

Figure 3.1: Number of articles included in selection process.

The 18 articles that were excluded with the reasons for exclusion after full text reading are presented in Appendix D – SMS exclusion form. Table 3.6 presents the 8 selected articles (7 by database search + 1 manually). This table presents the selected articles, the year of publication, authors and title of the article. In this table, we created an identifier for each article. This identifier will be used when there is a need to make reference to each of these articles.

Table 3.6: Systematic Mapping Study selected papers.

Article ID	Year	Authors	Title
S1	2004	ČubraniĆ, Davor, Gail C. Murphy, Janice Singer, and Kellogg S. Booth (ČUBRANIĆ et al., 2004)	Learning from project history: a case study for software development (Hipikat)
S2	2018	Greco C., Haden T., Damevski K. (GRECO et al., 2018)	StackInTheFlow: Behavior-driven recommendation system for stack overflow posts
S3	2018	Wu D., Jing X.-Y., Chen H., Zhu X., Zhang H., Zuo M., Zi L., Zhu C. (WU *et al.*, 2018)	Automatically answering API-related questions
S4	2016	Campos E.C., de Souza L.B.L., Maia M.D.A. (CAMPOS *et al.*, 2016)	Searching crowd knowledge to recommend solutions for API usage tasks
S5	2015	Wang T., Yin G., Wang H., Yang C., Zou P. (WANG *et al.*, 2015)	Automatic knowledge sharing across communities: A case study on android issue tracker & stack overflow
S6	2014	Rahman M.M., Yeasmin S., Roy C.K. (RAHMAN et al., 2014)	Towards a context-aware IDE-based meta search engine for recommendation about programming errors and exceptions (SurfClipse)
S7	2013	Correa D., Sureka A. (CORREA & SUREKA, 2013)	Integrating issue tracking systems with community-based question and answering websites
S8	2013	Ponzanelli L., Bacchelli A., Lanza M. (PONZANELLI et al., 2013)	Leveraging crowd knowledge for software comprehension and development (Seahawk)

Each of these articles was read, and to extract data from the identified articles, and the criteria extracted is presented in Table 3.7. Each field has a criteria item and a description of the criteria. The data extraction form presented in Table 3.7 was elaborated according to the SMS research question.

All articles presented in Table 3.6 were fully read and the criteria (Table 3.7) were extracted and are presented in Table 3.8. The analysis of the extracted data is presented in next section.

Table 3.7: Criteria extracted from selected articles.

Criteria	Description
Study ID	Identifier of study (to match Table 3.6)
Association Strategy	The strategy mapped in each study used to associate software development with Stack Overflow
Input	What was the input information used in each study in order to generate the association between software development and Stack O verflow
Output	What was the output information generated by the association strategy
Evaluation	How is the study evaluated, what sources of information were used to evaluate the proposal
Results	The results of the evaluation

3.5 Analysis

The analysis of the information extracted from the articles gathered during execution is presented in this section. For this work, a thematic rationale in order to categorize the details of the papers is proposed, by gathering articles according to a common subject discussed, and pointing out specific issues and differences from what is proposed in this dissertation. The author of this dissertation was responsible for extracting the information. The next subsections describe the analysis. This categorization process follows the systematic mapping study protocol proposed by (PETERSEN et al., 2015). Based on the information extracted from the studies, it was possible to answer the research question of this SMS by formulating Table 3.8 and next sections further detail these studies into categories.

3.5.1 Searching for and delivering Stack Overflow Posts

Most research on searching for and recommending Stack Overflow Posts have been carried out starting in 2012. Most of the related works have as output a Stack Overflow Post or a code snippet that is embodied in a Stack Overflow Post. Hipikat (S1) is the exception, as it does not recommend Stack Overflow Posts, but it recommends other artifacts from the software project, such as project documents, source file versions and others. All the other papers (S2, S3, S4, S5, S6, S7 and S8) deliver minimally a Stack Overflow Post or snippet from a Post to developers.

Table 3.8: Results extracted from articles.

Article ID	Association Strategy	Input	Output	Evaluation	Results
S1	Hipikat infers links by combining information contained within the project artifacts and the meta-information about the artifacts from different information sources	Artifacts (tasks, source file versions, messages on forums, project documents) produced during development	Artifacts	Qualitative. New comers using Hipikat x experienced developers	Interview: Newcomers found related files faster, although experienced performed better in easy and difficult tasks.
S2	Uses the source code to generate Stack Overflow queries. Manual, difficulty, user action, error.	Text similarity between Code or Exception on IDE	Stack Overflow code snippets or Posts	Qualitative. Log analysis of tool's performance. Clicking in a query's result indicate effectiveness.	794 queries logged. 53% manual, 19% difficulty, 15% user action, 13% error.
S3	Associate text of API tutorial with Stack Overflow using Cosine Similarity	API Tutorial	Stack Overflow Post	Quantitative. 30 SO Posts for each API: JodaTime, Math, Official Collections, Jenkov Collections Smack APIs	Precision: Top5: 24.32% Top10: 20.67% Top15: 19.18% MRR: 34.42%.
S4	Text Similarity between issue's text and Stack Overflow posts text	Issues from cookbooks for Swing, Boost and LINQ	Stack Overflow Post	Quantitative. 35 cookbook tasks were submitted to a SO dump (similarity). Results were manually analyzed by authors qualitatively.	NDCG: 0.3583 (35%) for Relevance 0.5243 (52%) for Reproducibility.
S5	Text Similarity between issue's text and Stack Overflow posts text	Android's Issue text	Stack Overflow Post	Quantitative. Uses Android Issue Tracker Tests Semantic Similarity w/ Stack Dump, Temporal Similarity and, Temporal and Internal Links.	Precision: Top10 0.49, 0.56, 0.62. MRR: 0.29, 0.33, 0.37.

Article ID	Association Strategy	Input	Output	Evaluation	Results
S6	Text Similarity between IDE exceptions text and Stack Overflow posts text	IDE exception's errors text	Stack Overflow Post and search queries	Builds a Stack Overflow corpus context (exception / code). 38 unique traces from 6 grad students. 37 common exceptions from Java were included.	MeanPrecision: Top10: 0.1229 Top20: 0.0736 Top30: 0.0538 Accuracy[11]: Top10: 68% Top20: 73.33% Top30: 74.66%
S7	Text Similarity between issue's text and Stack Overflow posts text	Android's Issue text	Stack Overflow Post	Quantitative. Compare Issue's Title with Stack Overflow's Title. Used 2 Samples. Compare SO's return with SO's links on issue trackers.	Precision: Top10: 33:16% Top20: 36:88% Top100: 47:27%
S8	Suggests Stack Overflow code snippets based on the context inside Eclipse	Code	Stack Overflow Post (Snippets)	Quantitative. 3 experiments with 35 Java Exercises from a Book.	NDCG: 0.0907 (9,07%) for Relevance. Precision: Top10: 0.0243 Top20: 0.0135 Top30: 0.0099 Accuracy: Top10: 18.92% Top20: 18.92% Top30: m18.92%

[11] Implemented w/ other approaches (ANTUNES et al., 2012, PONZANELLI et al., 2013).

Seahawk (S8) uses pieces of the code written by developers in Eclipse in order to search for entire code snippets, saving developers the time they spend in typing an entire block of code that is already in Stack Overflow. The approach used in Crosslink (S5) is different, in the sense that it uses text similarity to match Android Issues with Stack Overflow issues, and recommends the most similar posts. Similarly, the work from Correia et al. (S7) also uses text similarity in order to associate Stack Overflow and Android Issues. This work also analyzes contextual features (such as question tags representing the topic) to recommend Stack Overflow questions in response to bug reports. The approach proposed by Souza et al. (S4) classifies Stack Overflow pairs of question/answers, comparing these pairs with the developer's code through text similarity, and also focuses on retrieving code snippets. The work from Wu et al. (S3) associates' text from API Tutorials with Stack Overflow Posts, using Cosine Similarity. It retrieves questions regarding the API the developer is currently working on. StackInTheFlow (S2) is a recent work – published in 2018 – and similarly to other papers, it is a tool integrated to the IDE that uses the source code to generate Stack Overflow queries. It uses information and events occurring on the environment to generate the queries and present the results to developers. These events can be the occurrence of an error, for example, then the tool automatically searches Stack Overflow with the error message that was presented. The tool also analyzes when developers are having difficulties in coding, these difficulties can be deleting a wide range of code or not typing anything for a certain amount of time. It starts a search using the code the user is currently editing. Users can also use this tool to search for Stack Overflow Posts manually.

3.5.2 Defining a common context and using text similarity

Researchers have attempted to evaluate the impact of defining a context and using text similarity to find correlations between them to recommend artifacts from the project history, such as Hipikat (S1). Hipikat (S1) does not use Stack Overflow information for the project, it uses other artifacts that have been created during the project, such as other tasks, source file versions, messages on forums and project documents. There is no external context associated, as for example a Stack Overflow Post. SurfClipse (S6) uses a context aware tool in order to identify possible search queries for exceptions presented in the IDE. The strategy is to search for similar texts presented in the exception and other context in the IDE and through this context; it builds a Stack Overflow corpus context containing the exception and the code that should help the developer fix the exception.

3.5.3 Considerations regarding the association between software development context and Stack Overflow

Hipikat (S1) uses as association strategy inference of links by combining information contained within the project artifacts and the meta-information about these same artifacts from different information sources. Some of the authors of this article have worked on Mylar (KERSTEN & MURPHY, 2005), which is a very famous relationship heuristic for software development artifacts. The importance of this article relies on the association rationale, which is very similar to the one proposed in this dissertation.

There are papers that deal specifically with issues from software projects, such as S4, S5 and S7. As detailed in Section 2.2.1.1, issues are essentially different from project tasks, but are important artifacts to use as input to search for Stack Overflow Posts that should help with these issues. As issues might not contain information regarding the domain of the system, only technological related problems, it’s somehow easier to think about a direct search in Stack Overflow using issues text. Still, the direct connection does not result in high accuracy, and to try to overcome that, the work S5 uses other strategies other than text association to increase accuracy results, such as Temporal Similarity and the discovery of Internal Links of Stack Overflow in issue’s text body.

3.5.4 Considerations regarding evaluation Samples

When considering characteristics of the evaluations of our Related Work, there are articles using both qualitative and quantitative evaluation methods. The area of recommendation struggles with cold-start problem. Other than that, when considering software engineering, there is the need for people to engage in studies that take time and sometimes depend on industry collaboration. Analyzing the selected Related Work, we realized that most of the articles use small samples to validate their studies (this also being a challenge faced in this dissertation). This motivated this section, to expose a discussion regarding the samples used to evaluate our related work.

S1 and S2 evaluate qualitatively and the remainder papers evaluate qualitatively. S1 uses an evaluation method that compares how easy it was for newcomers in a company to get artifacts relations using Hipikat and compared the results with tasks executed by experienced developers. S2 analyzed the log of the tool's usage and considered the high number of clicks as success. The other papers that evaluated quantitatively, used samples with the following number of items in the samples: S3 used 5 APIs and 30 Stack Overflow Posts for each; S4 used 35 questions from cookbooks from 3 different technologies, being 12 for the first, 14 for the second and 9 for the third cookbook. S5 used a large number of issues from Android Issue Tracker. Android Issue Tracker, in the time of the collection of the dataset, had a total of 151,815 issues and 30,572 threads, and a total of 653 direct links to Stack Overflow. The time period reported is between November 2007 until September 2013. S6 uses 75 traces from the IDE, being 38 stack traces from logs of 6 grad students and 37 common exceptions from Java traces. S7 uses two samples from Google Chronimum and Android Issue Tracker. Both of these platforms contain issues that software developers around the world have inserted. They matched the explicit links to Stack Overflow encountered in both platforms to the actual text of the Issue. They also surveyed software maintenance professionals in order to investigate solutions to common problems of the area. By the time this paper was published, the results have not been collected. And lastly, S8 uses as sample 3 experiments with 35 java exercises from books.

3.6 Threats to Validity

While Planning: Protocol and process were used in order to enable the search replication as well as minimize bias. In order to mitigate the recurrent issue on inconsistent terminology and also reduce bias, the Computer Science Library Liaison of the University of Waterloo was consulted in the beginning of 2018. After being aware of the research, a revision of the search strings' and their synonyms and keywords was proposed, in an attempt to increase the search string's reliability. The inclusion and exclusion criteria might cause us to miss some studies.

While Executing: The use of only one research database can lead to missing studies. One researcher performed the execution phase.

While Analyzing Results: The interpretation of data is a concern once bias can be introduced.

The author of this dissertation executed the planning, execution and analysis phases twice, first in January 2018 and then in September 2018, in an attempt to perceive if the author followed the same rationale throughout the SMS.

3.7 Conclusion

This Chapter presented the Systematic Mapping Study performed in order to find related works. There are several articles discussing the idea of reusing information generated during the software development in order to support future steps in the same or other projects. Our motivation is similar. However, the approach reported in this work highlights the importance of the curation work performed by developers. This dissertation also proposes searches in the project history for similar tasks, assuming similar tasks reuse the same curated Stack Overflow Posts to aid subsequent task issues. A Systematic Mapping Study found related work with approaches that use code, development artifacts and other information to leverage software development.

Some articles found uses and suggest code snippets (S2, S8) or API documentation content (S3). Most articles suggest automated searches for the occurrence of exceptions or issues during project (S2, S4, S5, S6, S7). Projects that are managed in languages other than English language might not find so many relevant content in Stack Overflow, as the great majority of Stack Overflow Posts are written in English; another consideration is that research shows that the text overlap between issues and Stack Overflow is not greater than 16% (CORREA & SUREKA, 2013). We can also mention that when considering only issues and associating them directly to Stack Overflow, project information is not considered, only technological information regarding what was the error or issue. None of our related work has mentioned the importance of the developer effort to formulate queries and pursue results (curation). Developers search for sources of support that can help them solve any kind of problem they might encounter, or even if the task they are working on depends on knowledge beyond they possess.

S1 uses artifacts created within the project, which is similar to what is proposed in this dissertation, and records what is called "project memory". This history called “project memory“ is a logical association of artifacts, providing the developers a source of preset associations of recurrent use of artifacts. The approach considers that a high amount of artifact association is useful to suggest developers which artifacts they should then review, in case it matches what is stored in the “project memory”.

This dissertation considers not only curation performed by developers, as it goes beyond: it uses as source the project and information regarding project in order to find similar project tasks, and suggest the curated work to a task that is likely to need it.

These characteristics indicate this work goes beyond the scope of some of the related work. Next Chapter discusses the approach proposed in this dissertation in detail.

4 Study on reusing curated Stack Overflow Posts

This chapter describes the study that investigates the possibility to associate project tasks with curated Stack Overflow Posts. It first introduces the study and gives an overview of the problem, then it is divided in two parts: the report of a preliminary study performed and the main study proposed.

4.1 Introduction

Software developers rely on Q&A websites (such as Stack Overflow) in order to solve technology-related issues while working on project tasks (KOCHHAR, 2016). Developers use keywords that help them retrieve Stack Overflow Posts about similar or even identical problems other developers already faced and reported. Once the developer who needs support finds a similar solution, it is then implemented and tested, and the software system should work as required, the developer then completes the project task. The information that supported the developer to completing the project task is not saved nor indexed anywhere within the project and therefore it is never associated with the development workflow and the project explicitly. Also, there was time invested in choosing what the suitable solution would be among a list of Stack Overflow retrievals. If other team members have the same or similar problems, they will have to re-execute the search by repeating all the steps, including reasoning about a search string, reviewing possible solutions among all retrieved search results, selecting and implementing the solution (or more than one solution, in case developers are not sure if the chosen solution works) and testing it. In this work, we call these steps *curation*, given the fact that developers have to select among a list of possibilities that they will be implementing in the project.

Research shows that developers spend more than 10% of their time searching the web looking for work solutions (MEYER *et al.*, 2017). The support obtained is not explicitly integrated into the software project. Because software development is iterative, similar – or even identical – tasks can be executed several times during iterations. Attaching what was used to support developers in the project should allow reusing Stack Overflow Posts based on a task similarity discovery engine. Avoiding repeated searches results in greater productivity to software development (ROBILLARD *et al.*, 2014). In order to be able to suggest curated Stack Overflow Posts based on project task similarity, there are a number of procedures that need to be implemented to allow Stack Overflow Posts suggestion. There are benefits of integrating Stack Overflow posts with software development, according to (MEYER *et al.*, 2017), such as:

- It is beneficial in terms of keeping relevant information in the project;
- It avoids repetitive searches/selection of the same information;
- It is a practice that can help less experienced developers know how experts are working;
- It reduces workflow interruptions; and
- When this information is proactively recommended, it can also result in productivity for the development of the software.

This dissertation proposes a study on the viability of reusing curated Stack Overflow Posts when identifying similar project tasks. In this Chapter, three important steps of this study are laid out: a preliminary assessment, a project task context identification and finally, an implementation of a process that retrieves similar project tasks and is able to perform evaluations regarding the similarities retrieved.

4.2 Study Overview

Searching for Stack Overflow Posts is a typical procedure among daily activities of software developers (PONZANELLI *et al.*, 2013). The act of searching and selecting one solution on Stack Overflow is referred to as *curation* in this dissertation.

The curation of Stack Overflow Posts occurs according to the following steps:

1. A developer has a problem, and creates a search string that might retrieve satisfactory results;
2. The search string is submitted to Stack Overflow;
3. Stack Overflow executes the search, according to internal algorithms and lists the results according to the search string created by the developer in the first step; and
4. The developer selects one (or a set of) Stack Overflow Post that helps.

Each of these steps can be executed repeatedly, until the developer is satisfied with the results listed and the developer chooses a Stack Overflow Post that helps the project task resolution. Once a solution is chosen, curation is over. The curation steps are represented in a BPMN process in Figure 4.1.

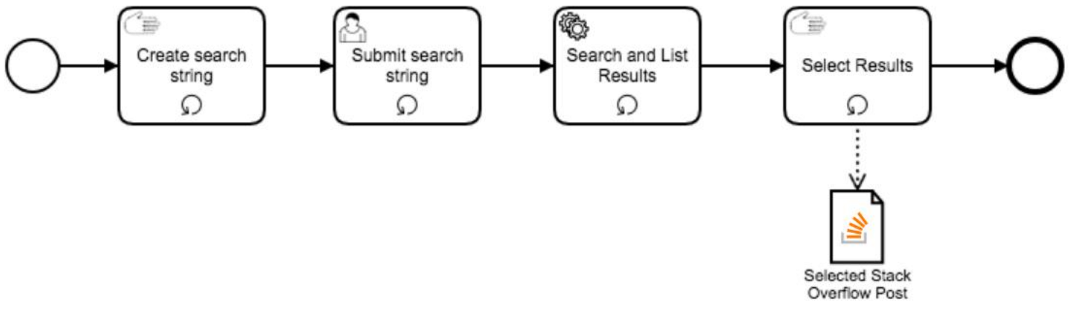


Figure 4.1: BPMN process representing curation steps.

Commonly, after a solution from a Stack Overflow Post is selected to be used by developers, the post from where the solution was extracted is not associated with the project hence cannot be reused by developers working in the same or similar projects. There is a growing body of literature that recognizes the importance of associating external knowledge to the development (CORREA & SUREKA, 2013; PONZANELLI et al., 2013; PONZANELLI et al., 2014; WANG et al., 2014; KOCHHAR, 2016). Stack Overflow is an important source of knowledge for software developers, and given its importance, Stack Overflow is the source of support considered in this dissertation. This support is presented in the form of curated Stack Overflow Posts. These posts are pairs of one question and answers, along with comments and users' information that was previously selected among a wide list by a developer and implemented during the task solution. Throughout the description of this proposal, curated Stack Overflow's Posts will often be referred to as *KPost or KPosts*. Developers themselves accomplish the work of choosing the correct KPost among a list of results, and implement the solution to resolve a specific project task this developer is working on. This selection work is referred to in this dissertation as curation. Researchers have already proposed approaches to retrieve Stack Overflow Posts for developers during the development of the software product, and although they are able to suggest Stack Overflow information, the articles either fail in covering a broad aspect of project tasks other than suggesting code lines or artifacts, or they fail in not considering the curation effort at all. The direct association between software project context and Stack Overflow can have some other drawbacks, such as low precision in the information retrieval and important curation efforts not considered, and therefore, KPost reuse is hampered. The aforementioned conditions motivated this dissertation, which presents an approach to aid the automatic correlation of curated Stack Overflow Posts to a project task, enabling the reuse of KPosts.

The proposed study aims at using project task information and a text similarity retrieval process to aid the suggestion of Stack Overflow Posts during software development, according to the objectives presented in Section 1.2. Through the identification of similar project tasks, KPosts that were once used to support a task during its solution could be automatically linked to a similar task encountered. The process functionality is receiving as input project tasks, submitting these tasks to a similarity retrieval process, and by discovering the similarity among pairs of tasks, linking the Stack Overflow Post of a project task to the most similar project task discovered by the similarity retrieval process. The main study features are illustrated in Figure 4.2. In this illustration, there are four tasks in the Project (1). Each project task is composed of information, which we refer to as "Context" (2). Three of these tasks (*task1, task2* and *task3*) are resolved, represented by the white color of the "Task Status" caption (3), and have Stack Overflow Posts (4) associated to them. One task is not resolved (*task4*), neither have a Stack Overflow Post associated to it. When all the tasks are submitted to a similarity retrieval algorithm (5), the similarities between the project tasks are retrieved (6). These similarities are expressed in a range of 0 to 1. In the example, when comparing *task4* to *task1*, the similarity index between them is 0.6, meaning these tasks are 60% similar. When comparing *task4* to *task2*, the retrieved similarity is 0.7, meaning these tasks are 70% similar. Finally, when comparing *task4* to *task3*, the result is 0.2, meaning these tasks are 20% similar. According to the proposed approach, as the most similar tasks should share the same Stack Overflow Post (7), it is then suggested that *task4* should also be related to the Stack Overflow Post associated to tasks *task1* and *task2*, as they are the most similar tasks according to the similarity index extracted in the illustration.

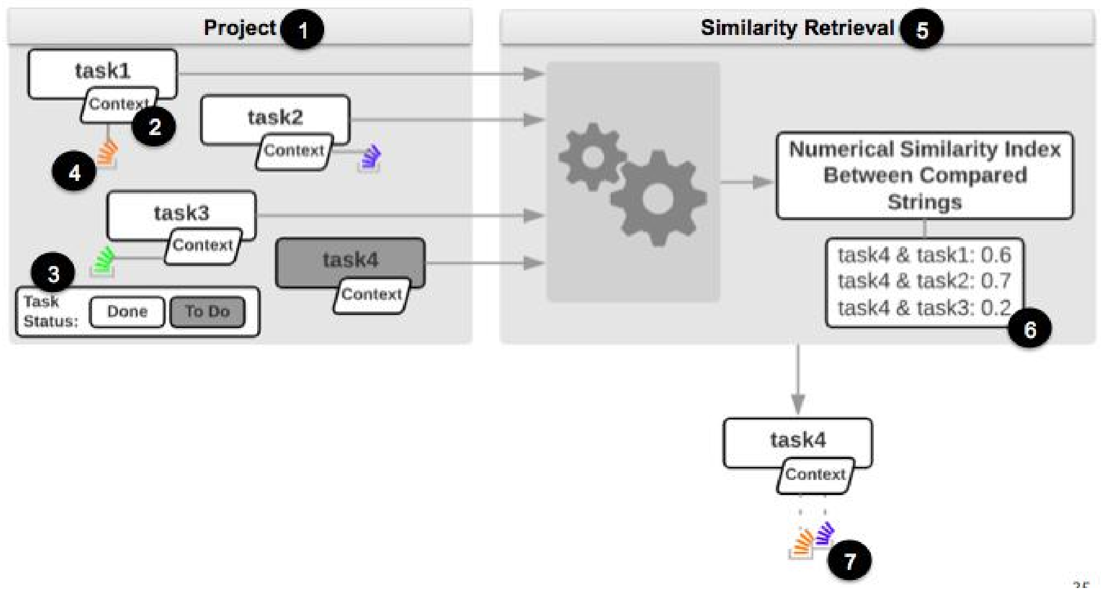


Figure 4.2: Proposed approach overview.

Developers perform the work of associating Stack Overflow Posts to project tasks (4), after they have curated this KPost. Therefore, part of the solution is the developer's responsibility. The associations' suggestions created automatically by the similarity retrieval execution are represented in Figure 4.2 by the marker #7.

To assess the viability of this study before deepening the proposal research and implementation, we decided to run a preliminary assessment study. This preliminary assessment is described in Section 4.3. After, as the preliminary assessment returned positive results, we kept pursuing the proposal research, and we present the studies performed to characterize a project task context and the implementation for the similarity between project tasks from Section 4.4 on.

4.3 Preliminary Assessment

According to our methodology, we first performed a preliminary assessment in order to verify the feasibility of the proposal. This section presents the preliminary assessment study performed. We considered evaluating the recommendation of Stack Overflow Posts based on the association of a curated Stack Overflow Post to the project task and further discovery of similar project tasks, in order to retrieve to developers KPosts that could possibly be reused. We decided that it was important to first check within an industry scenario if discovering the text similarity of project tasks would be a good approach to suggest Stack Overflow Posts, if this relation was somehow feasible. We ran this reported study and the results were later published in CibSE Conference, 2018 (MELO *et al.*, 2018).

4.3.1 Planning

The aim of the preliminary assessment was to assess the usefulness of the associations of KPosts based on project task's similarities. In other words, the goal of the experiment is to discover similarities between project tasks and verify if the KPost associated to one task could support similar project tasks, in an industry practitioner's view. We conducted a structured interview (KAJORNBOON, 2005), where the same question with the same wording was asked in the same order. The aim of this type of interview is, according to (KAJORNBOON, 2005),

> "… to be given exactly the same context of questioning. This means that each respondent receives exactly the same interview stimulus as any other. The goal of this style of interview is to ensure that interviewees' replies can be aggregated … Questions are usually very specific … "

One of the strengths of this methodology, proposed by (KAJORNBOON, 2005), is the fact the researcher who is interviewing has control of the topic and questions asked. As the question we aim to ask is very specific and has one direct answer, we chose this methodology for this study. The following steps were then performed in this preliminary assessment:

1. Software developers from a company were asked to identify project tasks that they could associate with Stack Overflow posts, informing which Stack Overflow post helped which task (tasks that had curated Stack Overflow Posts).

2. Once we had this result, a similarity discovery is performed using a java implementation of the *Levenshtein* distance metric (Appendix B – Preliminary Assessment Implementation Code contains the source code) to discover similar tasks to each one of the tasks the developers provided.
3. Once the similarity index results were retrieved, we created a form in the format of a table with all discovered tasks that were 50% similar or above to the ones that had a Stack Overflow Post associated;

4. Developers that were willing to participate in this study answered, for each similar task found, the following question: "**Can possibly use the same Stack Overflow post?** " where the developers should answer YES

or NO to each similarity discovered by the algorithm, in a structured form sent to them with all similar tasks listed;

5. This form was sent to the software engineers of the company, in the form of a questionnaire, and the ones who were familiarized with the project tasks, answered the questions;
6. The exact same question was submitted to all developers that participated, following the structured interview method (KAJORNBOON, 2005).

The company and participant selection criteria were from our professional network. The criteria for selection were participants that had more than five (5) years working with software development, which worked in companies that managed project tasks in automated project management tools (which allows an automated similarity discovery) and used, at least partially, some degree of process definition (iterations, repeatability). We did not select participants who knew about this research to avoid bias.

4.3.2 Execution

We sent emails to selected contacts from our professional network that work in different kinds of software development businesses:

- **Contact 1**: Full-time software engineer from a software factory that mostly works with public sector software projects;
- **Contact 2:** Full-time project manager from one big retail chain business;
- **Contact 3:** Full-time software architect from a small software factory that works in the private and public sectors software projects;
- **Contact 4:** Full-time software architect from a North American startup company;

They were asked to provide project tasks and their associated Stack Overflow Post, to each task. We also explained how the study was supposed to occur after they provided the tasks with their associated Stack Overflow Post, that their help would be needed for a second time, in order to answer the questionnaire. We received answers from all contacts. ***Contact 1*** answered that could not provide an association with Stack Overflow Posts to their tasks, although it was part of their routine during software development. ***Contact 2*** answered that he could look among the project management tool some Stack Overflow Posts that helped him or developers of his team solve tasks, but his manager did not allow us to access the company's project management tool data in order to look for similar tasks, as the second step of this study would require. ***Contact 3*** answered the email sent with three tasks from one project, and also a tabulated file containing all tasks from this same project created until that date, extracted from the project management tool. ***Contact 4*** answered similarly to ***Contact 2***, stating he could look for old tasks that had references to Stack Overflow Posts, but when he talked to his manager about providing other tasks in order to find similar tasks, the manager was not helpful, according to him, due to the novelty aspect of new features of systems the company is currently developing, and also because his manager was afraid he would be disrespecting some North American law he was not aware of. The data of the company ***Contact 3*** works in were then used for this study. We had all data we needed: tasks with associated Stack Overflow Posts and all the tasks from the same project, which were more than 4000.

The company was founded in 1998 and is focused on software solutions for companies with a wide range of application needs in both the private and public sectors in Brazil. For the last 10 years, the company has mostly developed software for logistics and cargo transportation in container ships and trucks. With the help from the company's software architect and one mid-level software developer, we were able to gather information of tasks, their possible related Process Activities and also, the Stack Overflow posts that each developer used on the solution for each task. We were able to access data from the company's project management tool with the help of developers that work in the company. The software developer of this company suggested the project to be analyzed in this study, based on the fact that this was the oldest project in the company with the most project tasks and had been using the SCRUM process successfully during its development.

After we characterized the tasks we received, the similarity retrieval was executed using the entire task database (dump file) of the same project of the company, including archived tasks (around 4000 tasks extracted in a file at the time of the assessment). After executing a similarity search over the project database records and retrieving similar tasks, we created a questionnaire and returned to the company, asking if the same Stack Overflow posts could be reused during the resolution of similar tasks that were discovered. For the second phase of the study, we received answers from one software developer in this company. This software developer worked in this company for 5 years and has a total of 6 years of experience developing software.

The tasks descriptions (e.g., title, context) are in Portuguese, and were not translated into English during the similarity search in order to keep the original text and because this proposal is intended to be multilingual. But for information purposes, the three chosen tasks are presented in this dissertation with their English translation.

Developers associated three Tasks with Stack Overflow Posts. The associations are presented in Table 4.1.

Table 4.1: Manual association between Task and Stack Overflow posts.

Task ID	**Task Title (Portuguese) + Task existing Context**	**Task Title (English)**[12]	**Stack Overflow Post**
Task1	[2.10.2] Evolutiva 39 - Inclusão de combo no campo E-mail envio NFS-e + Construir e testar código	[2.10.2] Evolution 39 - Inclusion of combo in E-mail sending NFS-e field	29174164
Task2	[2.10.13.1] [Recepção] [Cheio] Sistema não registra ocorrência de envio de email ao criar pedido	[2.10.13.1] [Reception] [Full] System does not record occurrence of email sending when creating order	25636091
Task3	[2.10.2] Erro ao subir aplicação em homol após inclusão do campo listaEmailNFSe	[2.10.2] Error uploading test application after inclusion of lis-taEmailNFSe field	25996758

4.3.3 Reporting

In the second phase of this study, we submitted each of the three (3) tasks to a text similarity comparison with all tasks from the same project using a Java implementation of the *Levenshtein* algorithm. The implemented code is presented in Appendix B – Preliminary Assessment Implementation Code. The most similar discovered tasks (>= 50% similar) are presented in Table 4.2, Table 4.3 and Table 4.4, for each task. A company's software developer provided the answers regarding if the same Stack Overflow post could possibly be reused by the similar tasks discovered by the java implementation.

[12] for information purpose only

Table 4.2: *Levenshtein* similarity calculations and developer's opinion about Stack Overflow post recommendation – Task1.

% of similarity (approximated)	Description	Can possibly use the same Stack Overflow post?
		DeveloperAnswers
52%	[HOMOL][Evolutiva] Envio de e-mail – Interceptor	Yes
51%	[HOMOL] Alteração do texto do e-mail Meio Ambiente	No
51%	[TESTE] Alteração do texto do e-mail Meio Ambiente	No
52%	[TESTE][Evolutiva] Envio de e-mail - Interceptor	Yes
50%	[Tela 3] - Criticar exclusão de transportador do pedido	No
50%	[2.2] Alteração do nome de procedure de DTC	No
50%	[HOMOL 2.2] Evolutiva na Consulta Boletim de Pesagem	Yes
50%	[DTC] E-MAIL - Envio de e-mail com PDF	No
51%	[RETIRADA] Importação - Alteração do formato do campo CEMercante para string	No
51%	[2.5.0]Erro ao finalizar Recepção Vazio Embarque Direto	No
51%	[2.3.8] HPU - Alteração do nome do bloqueio	No
50%	[Navis] Validação da solução de envio multiThreads	No

Table 4.3: *Levenshtein* similarity calculations and developer's opinion about Stack Overflow post recommendation – Task2.

% of similarity (approximated)	Description	Can possibly use the same Stack Overflow post?
		DeveloperAnswers
51%	[Reprogramar Recepção] Sistema está exibindo mensagem de erro ao reprogramar	Yes
53%	[ConsultaPedidoRecepçãoCheio] TESTAR Rever acesso do Botão Editar Pedido	No
54%	[Criar Pedido CHEIO] SIstema não gera ocorrencia de criação de HPU (BLOQUEIO NAVIS) na criação do PEDIDO CHEIO	Yes
51%	[HOMOL_2.3.8] Recepção Vazio Embarque Direto - Envio e-mail PDF	No
50%	[2.5.0] Recepção Cheio e Vazio - Alteração de restrição de utilização de container	No
54%	[PRODUÇÃO] RECEPÇÃO - Sistema não exibe mensagem de taxa não paga ao aprovar	No

54%	[TRUNK] RECEPÇÃO - Sistema não exibe mensagem de taxa não paga ao aprovar	No
51%	[RECEPÇÃO] Registrar no LOG que o usuário cancelou o envio de email automático	No

Table 4.4: *Levenshtein* similarity calculations and developer's opinion about Stack Overflow post recommendation – Task3.

% of similarity (approximate)	**Description**	**Can possibly use the same Stack Overflow post?**
		DeveloperAnswers
51%	[Tela 2] Erro como Administrador do Sistema	Yes
51%	[Tela 1] - Aplicar Exceção E4 como pré-condição da tela 1	No
50%	[2.5.0]Erro ao finalizar Recepção Vazio Embarque Direto	Yes

The results collected and presented in Table 4.2, Table 4.3 and Table 4.4 were analyzed and they are discussed in the next section.

4.3.4 Discussion

According to the company's software developer who answered the assessment question, for all three tasks, there was at least one similar task that could possibly reuse a Stack Overflow post. As presented in Table 4.2 the Task1 scenario for encountered 12 tasks that are at least 50% similar to Task1, and from a group of 12 similar tasks, a quarter of them can possibly reuse the associated Stack Overflow post. In the Task2 scenario, presented in Table 4.3 from 8 similar tasks, 2 can possibly reuse the same Stack Overflow post associated with Task2. Task3 has 3 similar tasks and 2 of them could have reused the associated Stack Overflow post, as presented in Table 4.4. The developer who provided the 3 initial tasks and their associated Stack Overflow Post and the developer who answered the assessment question were two different people. The first one is a senior developer who has been working in this company for 7 years and has a total of 10 years of experience as a software developer. The professional who answered the assessment question is a software architect with 15 years of experience as a developer and who has been working for 10 years in the company. The person responsible for inserting the project tasks on the project management tool varies. In this case, for the 3 selected tasks, the senior developer inserted two of them, and one other team member, who is a project manager in the company, inserted the other one.

The presented study result provides posts that are relevant to developers executing specific tasks. Preliminary, though modest, results indicate that, through the proposed solution, it is possible to bridge project tasks and Stack Overflow posts and to reuse Stack Overflow Posts when similar tasks are found. The proposed approach – using project task similarity to reuse curated Stack Overflow Posts – was able to generate recommendations within all analyzed scenarios, according to the developer interviewed. Also, according to this developer, the recommended Stack Overflow post could be useful for the execution of at least 25% of similar tasks. This finding, while preliminary, suggests that KPosts can be reused by identifying similar tasks and that the proposal was worth to keep pursing further.

4.3.5 Threats to Validity

As threats to validity for this study we can mention:

(1) The sampling used for this interview might not be representative enough; this is due to the fact that it was difficult to find companies that were willing to share their data, and more importantly, that had explicit associations between project tasks and Stack Overflow Posts in their documents, histories, version control services and project management tools.

(2) The study used project tasks that were already resolved (done). A more accurate approach would be to include new tasks, perform suggestions, and ask if the suggestions provided were useful for developers, in real time. As this study was only preliminary, we considered this as a threat, not as a circumstance that makes this study impracticable.

4.3.6 Conclusion

A preliminary assessment was performed aiming to assess the proposal base idea through the point of view of an industry practitioner. Therefore, invited participants contributed in this assessment, which consisted in a structured interview about similarity findings provided, using real data from a project in a software development company.

From the results, we could perceive that the approach is valid, according to the conclusions about the software developer’s answers, and we also concluded that the approach needed further improvement and a quantitative evaluation. The study improvement is reported in the next sections of the current chapter, and the evaluation of the proposal is reported in Chapter 5.

4.4 Study on reusing curated Stack Overflow Posts

The next sections of this chapter describe how the proposal was further developed after the preliminary assessment conclusions. First, we researched what comprises the context elements of a project task. After, we implemented a process using the investigated context and used a data science platform that aids data solutions for implementation. This process provides similarity indexes when comparing text strings from project tasks context. Through the similarities retrieved in this implementation, the decision regarding associating Stack Overflow Posts was evaluated and described in Chapter 5.

4.4.1 Project Task Context

Project Task Context is a set of information that composes a project task. In other words, contexts are specific characteristics from one project task, which makes a project task unique. Project tasks are the project assets used in this approach, since the suggested association of a curated post to a project task will be provided according to the similarity of project tasks, the possibility to reuse Stack Overflow Posts arises from the similarity among project tasks. Because this project task context is an asset in this research, an investigation on project task context elements was performed. From this investigation it is possible to identify a group of project task context elements that are suitable to be submitted to the similarity retrieval process implementation. The suitability is determined by the characteristics of each project task to be used to find similar tasks. We consider this project task context investigation important in order to contribute to the guidelines of what information from project tasks are suitable for similarity comparisons, as well as the possibility to remove researcher’s bias by analyzing the information on how both academia and project management tools perceive project tasks in software engineering.

We performed an *ad-hoc* literature review regarding what kind of information exists in software projects and an investigation of three project management tools. We were able to identify project task context elements from the theoretical foundation literature and also from project management tools. What was found in the literature was that software engineering is knowledge-intensive due to its dynamism and the massive amount of technology used activity (DI CICCIO *et al.*, 2015) (LINDVALL &

RUS, 2003). According to Lindvall et al., software engineering has two types of knowledge associated with it: *technical* and *business domain information*. Technical knowledge refers to design (design patterns, heuristics, best practices, technical constraints and estimation models), programming (programming languages and development tools) and software processes (methodology, code testing and debugging procedures). Business domain knowledge refers to information regarding aspects of a specific application (the customer's business processes, business rules, activities, stakeholder needs, business goals for software). This dissertation does not consider application domain information; it only considers technical information because this proposal's aim is to be agnostic to business characteristics. From this analysis, a need for a context element that can store technological information was perceived. Therefore, we propose a context element to store technological information of project tasks. This element can be expressed as *tags*. Tag is a term related to a piece of information. In this case, the terms are any technical information directly related to the task that can characterize it.

To broaden our study and capture what is currently used by industry, we also included project management tools used in software engineering in the project task context identification. Researchers also believe it is important to consider other sources in grounding research other than formal literature in software engineering (GAROUSI et al., 2016); considering other sources allows a broader theoretical aspect and brings practical insights to the work. Therefore, we analyzed project management tools information regarding project tasks. In this search, we concluded that some of the project task elements identified in the literature were also reported in software development tools that support project workflow. The tools analyzed were JIRA[13], Trello[14] and Redmine[15]. With this analysis, we were able to identify context elements such as a task's title (subject), a task's description, a project to which a task belongs, and to what process information the task can be associated. Another element identified was the category of a task. This category is determined by how the company wants to classify the project tasks. Categories can be, for example, development and testing, or a type that classifies a task. After performing analysis from both *ad-hoc* literature search and of project management tools, we present a list of the identified context elements in Table 4.5. This table has the following attributes: the name of the context element identified, a description of each context, and also presents where each context

[13] atlassian.com/Jira
[14] trello.com
[15] redmine.org

element was identified. For a context element found in the literature, the reference of the article will be presented; if found in a project management tool, the name of the tool is presented.

Table 4.5: Context description and source of elements.

Context Element	Context Description	Source of context element identification
Project/Board	The name of the project that tasks belongs to	Redmine Trello (Board) JIRA
Project Tag	Tags related to the project	(LINDVALL & RUS, 2003)
Process	Process information that can be associated to the task	(LINDVALL & RUS, 2003)
Task Title	The title of the task	Redmine Trello JIRA
Task Description	The description of the task	Redmine Trello JIRA
Category/Type	A task category or type; a classification used to divide tasks into different niche	Redmine (Category) JIRA (Type)
Task Tag	Tags related to the task	(LINDVALL & RUS, 2003) JIRA Trello

Project is a context element identified in all project management tools investigated. In *Trello*, it can also be identified as a *Board*, as each project has its own board view. *Process* as context for project tasks was found in the literature. Interestingly, although not identified in project management tools as default information for project tasks, in the example of project task context in Section 4.4.1.1 the company that provided the example had created a customized field to control Process information for each task. *Category*, *Task Title* and *Task Description* are found in all project management tools. Finally, task *tags* are general technological information related to the specific project or task that identifies them, as found in the literature and also in two of the investigated project management tools. Tags can be unlimited and created at the discretion of the team member responsible for creating or editing tasks. Technological information related to the task that can briefly characterize the task, using one or at most two words, is a tag. Tags are also suited for projects. Each project has specific context in respect to product (SANTORO *et al.*, 2006), such as technological characteristics that every task will inherit necessarily, indicating the need for a project tag. Examples of tags contexts' contents are presented in Table 4.6, for clarification.

Table 4.6: Examples on Tags for Project and Task.

Identified Task Context	Examples
Programming Language (ProjectTag or TaskTag)	java, sql, jQuery, C++
Application Middleware (ProjectTag or TaskTag)	jboss, weblogic, tomcat
Automation Tool (ProjectTag or TaskTag)	maven, jenkins
Persistence Framework (ProjectTag or TaskTag)	hibernate
Database (ProjectTag or TaskTag)	oracle, sqlserver, mysql
IDE (ProjectTag or TaskTag)	eclipse, notepad, visual studio
Error Message (TaskTag)	ORA-0600, javaPersistenceError

Using another perspective to represent the identified project task context elements, a domain model is presented in Figure 4.3. This domain model also presents Stack Overflow as an external package associated to the task through curation.

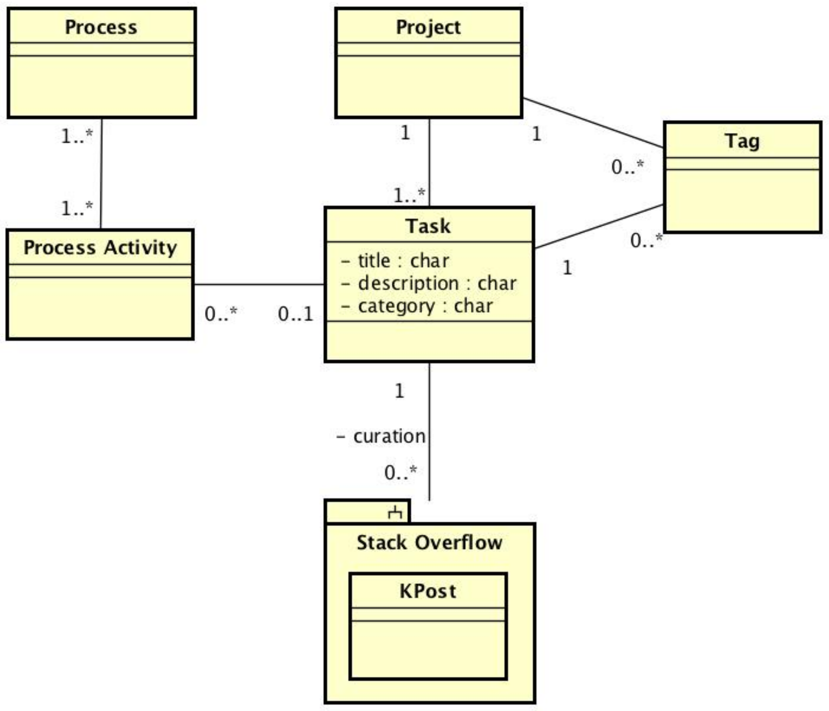


Figure 4.3: Domain project task context elements' model.

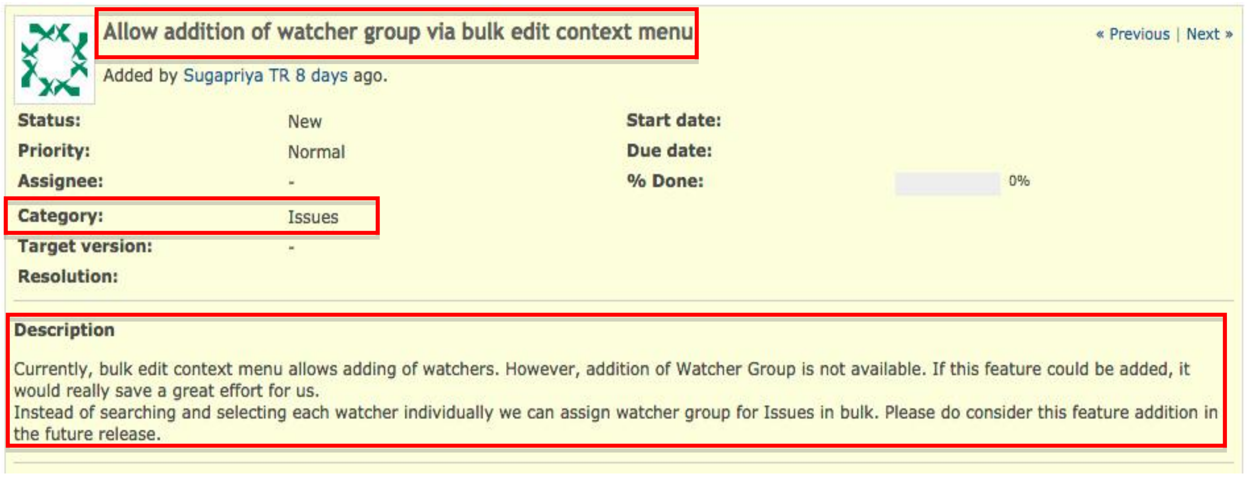
In this section, project task context elements were identified in both literature and project management tools. The next section presents an example of a project task from *Redmine* project, and what would be this randomly selected tasks context elements.

4.4.1.1 Project Task Context Examples

Two examples of project tasks are presented to illustrate project task contexts within real project task scenarios.

The project management tool *Redmine* has publicly available data regarding feature requests for the tool, issue reports, and Wikis. It is therefore, an example to illustrate project task contexts data from the *Redmine* project. We randomly chose a project task to use as example. Figure 4.4 presents the screen of one registered project task retrieved from the *Redmine* project. The red boxes identify the contexts of the specific task we were able to identify.

Figure 4.5 presents another example from a project task. This example was provided by a Brazilian software development company. The data used was from the same company described in the preliminary assessment (Section 4.3).



Allow addition of watcher group via bulk edit context menu

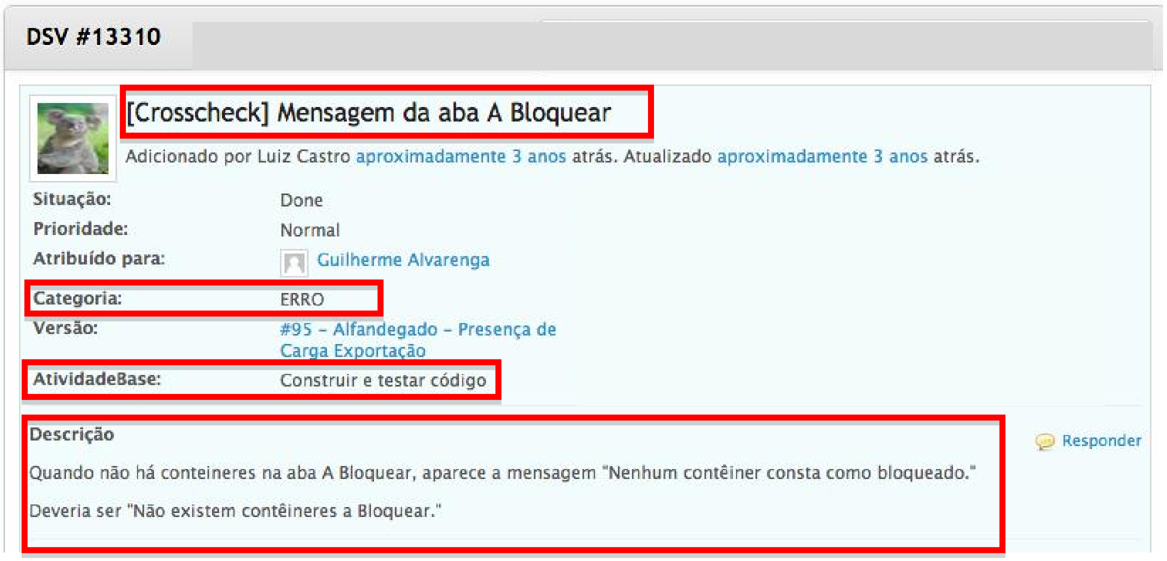
Added by Sugapriya TR 8 days ago.

Status:	New	Start date:	
Priority:	Normal	Due date:	
Assignee:	-	% Done:	0%
Category:	Issues		
Target version:	-		
Resolution:			

Description

Currently, bulk edit context menu allows adding of watchers. However, addition of Watcher Group is not available. If this feature could be added, it would really save a great effort for us.
Instead of searching and selecting each watcher individually we can assign watcher group for Issues in bulk. Please do consider this feature addition in the future release.

Figure 4.4: Task #29501 from Redmine project.


DSV #13310

[Crosscheck] Mensagem da aba A Bloquear

Adicionado por Luiz Castro aproximadamente 3 anos atrás. Atualizado aproximadamente 3 anos atrás.

Situação: Done
Prioridade: Normal
Atribuído para: Guilherme Alvarenga
Categoria: ERRO
Versão: #95 - Alfandegado - Presença de Carga Exportação
AtividadeBase: Construir e testar código

Descrição

Quando não há conteineres na aba A Bloquear, aparece a mensagem "Nenhum contêiner consta como bloqueado."

Deveria ser "Não existem contêineres a Bloquear."

Figure 4.5: Task #13310 from industry project.

The identified project task context from tasks #29501 and #13310 are described in Table 4.7. The first column presents the identified context elements in Section 4.4.1 and the following columns present the context retrieved from each example task.

Table 4.7: Context for Task #29501.

Context	**Task #29501 Redmine Project**	**Task #13310 Industry Example Project**
Project	Redmine (all tasks belong to project *Redmine*, although this information is not presented in this view)	Portal de Serviços (information not presented in this view)
Process	N/A	Construir e Testar Código
Category	Issues	ERRO
Task Title	Allow addition of watcher group via bulk edit context menu	[Crosscheck] Mensagem da Aba a Bloquear
Task Description	Currently, bulk edit context menu allows adding of watchers. However, addition of Watcher Group is not available. If this feature could be added, it would really save a great deal of effort for us. Instead of searching and selecting each watcher individually we can assign a watcher group for issues in bulk. Please do consider this feature addition in the future release.	Quando não há contêineres na aba A Bloquear, aparece a mensagem "Nenhum container consta como bloqueado". Deveria ser "Não existem contêineres a bloquear".
Project Tag	N/A	N/A
Task Tag	N/A	N/A

As presented in Table 4.7, it is possible to visualize the retrieved context from project tasks. The fields filled with “N/A” were the ones identified in Section 4.4.1 - Project Task Context but that were absent in the example; in the examples, the tool does not allow tags for tasks and the projects do not use custom fields to classify tasks using tags.

The next section presents the process implementation that retrieves the similarity among project tasks contexts and evaluates the study.

4.4.2 Study Implementation

After project task context elements were identified in both industry and project management tools, we proposed an implementation to obtain the similarity index between the project task context elements. This implementation comprises methodology step #6, which describes the process that allows the association suggestions of curated Stack Overflow Posts to project tasks. This process, in summary, should be prepared to receive as input a dataset containing project tasks associated with Stack Overflow Posts, then to retrieve similarities from pairs of project tasks and evaluates if the Stack Overflow Posts are the same (indication of reuse) between tasks with a high degree of similarity.

The proposal in this dissertation compares one project task (task’s context elements identified) with a set of tasks, and after identifying the most similar task from a list of existing tasks, it associates the curated Stack Overflow Post (or posts) of the most similar project task found among comparisons. This means that the tasks that are closer to the similarity index of 1 (100% similar) among all compared ones will have its KPost associated to the task requesting the KPost. The platform *RapidMiner*[16] was used to implement the process. *RapidMiner* is a very powerful data science platform, requiring a small learning curve to be used. There are reports of wide use in the academic field (SCHLITTER et al., 2013). This process generates a table with all the similarity indexes retrieved for each pair of compared tasks and is able to evaluate this similarity table. *RapidMiner* has embedded into its functionalities a wide number of algorithms to extract text similarity, including the *Jaccard* algorithm.

This current section presents how the process in *RapidMiner* is implemented and all tasks performed. This process retrieves the similarity indexes between the project tasks of a given dataset and evaluate if the comparisons with higher similarities refer to

[16] rapidminer.com

the tasks that use the same Stack Overflow Post. The implemented process receives as input a dataset with project tasks, gathers the similarity between each pair of project tasks and evaluates the similarity generated by providing metric results. Note that the referred similarity table is different from the dataset. The dataset is a list of project tasks. The dataset is used as input to a *RapidMiner* process. This process generates the similarity table during the execution of the process and evaluates the data of the similarity table.

The development of the process was performed using *RapidMiner* Studio version 8.2, the desktop version of *RapidMiner* platform. Each implemented step (represented by *operators* in *RapidMiner*) and the final process are described below. After all operators are described, we present the execution of the process the generated similarity index and the results in Chapter 5. The process created in *RapidMiner* is presented in Figure 4.6. Each operator will be detailed over the next sections. Operators that run sequentially compose the process. Each operator has a different responsibility, and the combination and order of operators can change the result of a process. In this process, the first operator loads the sample file with project tasks. A straight line connects operators. Each operator is a box that runs a unique procedure and the result is an input to the next operator. Every operator has semicircles that are ports for inputs and outputs, except for the *Retrieve* operator, that has no input, because this operator represents a loaded file. These semicircles are labeled icons on the side of operators. The inputs and outputs of operators are:

1. **out:** output port.
2. **ori:** the original data of the sample.
3. **exa:** the generated set modified by operators.
4. **sim:** similarity table generated.
5. **lab:** labeled data. A label input is applied in the example set and is delivered in this port.
6. **per:** performance vector for selected attributes.
7. **doc:** document or document set.
8. **res:** connector represents the end of the process.

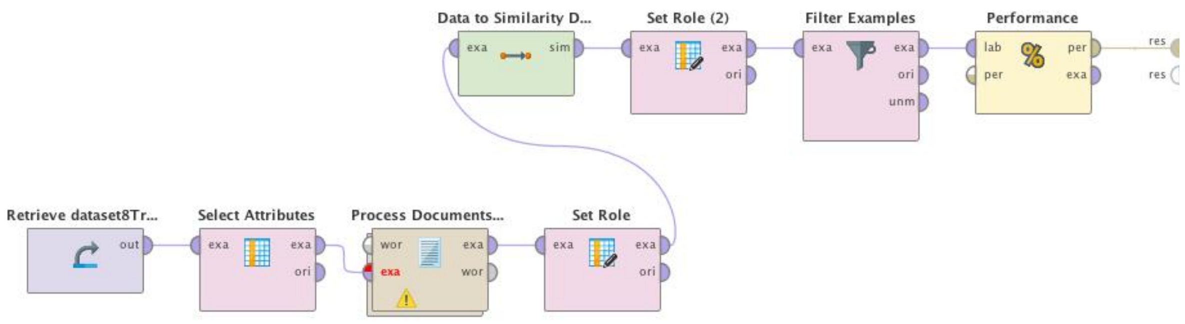


Figure 4.6: *RapidMiner* process.

Each operator and its functionality are described below.

Retrieve Operator: The first operator in the process is the *Retrieve* operator. This operator represents the dataset import in the process. According to *RapidMiner* documentation, this operator loads the desired repository into the process. It is necessary to inform the platform where the physical file is and also configure a few characteristics of the dataset. *RapidMiner* provides guided user interfaces (GUI) to aid the needed configurations. The configurations needed are: encoding, defining a specific character for comments, and column separator. The GUI also helps the user set column types. The result of the configuration is the *Retrieve* operator referencing a configured data sample. After creating this operator, it is necessary to select which attributes of the dataset (project task context elements) will be used to generate the similarities. The operator responsible for selecting the attributes is *Select Attributes*.

Select Attributes: The *Select Attributes* operator is used in order to select what attributes from the dataset will be the project task context elements to generate the similarity index extraction. This operator selects a subset of attributes of a dataset and does not consider the other attributes that were not selected. This attribute is then linked to the *Retrieve* attribute *out* port to its *exa* in port. After selecting project task context elements, they are submitted to a text pre-processing process. The *exa* output port of the *Select Attributes* operator is then connected to the *exa* input port of the *Process Documents from Data* operator.

Process Documents from Data: this operator is a sub process, responsible for the text pre-processing transformations. The transformations executed are the transformation of characters to lowercase (Transform Cases), the removal of every character that is not an alphanumerical character (Tokenize), stop-words filtering (Filter Stopwords) and lastly, the transformation of inflected words into a base or root form of the word (Stem). The sub process is presented in Figure 4.7. Label *doc* represents the inputs and output ports.

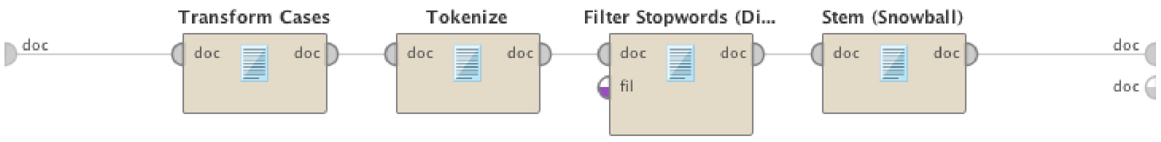


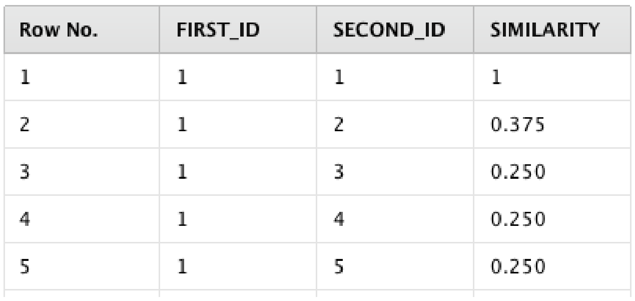
Figure 4.7: Process Document from Data sub process.

This operator's output, which is the text with which all text pre-processing configurations, is connected to the next operator input port, *Set Role*.

Set Role: This operator changes the role of an attribute of the dataset. It is needed by the next operator´s (Data to Similarity Data) input. This operator identifies which information from the input dataset is the dependent variable, meaning, information that will be suggested and submitted to evaluation further in the process.

Data to Similarity Data: this operator is responsible for creating the similarity table. It receives as input the configured and selected attributes of the imported dataset and provides as output the similarity table, containing all similarity indexes extracted from each project task comparison. Part of a similarity table example is presented in

Figure 4.8.



Row No.	FIRST_ID	SECOND_ID	SIMILARITY
1	1	1	1
2	1	2	0.375
3	1	3	0.250
4	1	4	0.250
5	1	5	0.250

Figure 4.8: Similarity table example.

The similarity table has four columns: *Row No.*, *FIRST_ID*, *SECOND_ID* and *SIMILARITY*. *Row No*. is an identification number of each generated row. *FIRST_ID* is the identifier of the row number of the dataset used as a base for comparison. This row is compared to the row informed in the *SECOND_ID* column*.* The *SIMILARITY* column is the discovered similarity index result as comparison between *FIRST_ID* and *SECOND_ID.* We can read this table information as: the first row of the similarity table (Row No. = 1) presents the first project task of the dataset (FIRST_ID = 1), compared to the first row of the dataset (SECOND_ID = 1) and the result of this comparison is a similarity index of 100% (SIMILARITY = 1). The second row of the similarity table (Row

No. = 2) presents the first project task of the dataset (FIRST_ID = 1), compared to the second row of the dataset (SECOND_ID = 2), and the result of this comparison is a similarity index of approximately 38% (SIMILARITY = 0.375).

Data to Similarity Data operator has two parameters: *Measure Type* and the algorithms available for each measure type. The parameter *Measure Type* is used for selecting the type of measure to be used for calculating similarity. The available measure types are: *mixed measures*, *nominal measures*, *numerical measures,* and *Bregman divergences*. These parameters define how to calculate distances for the attributes of the input dataset. This parameter is configured according to the dataset's configurations and characteristics. For this work, considering the dataset has text columns only, the option selected for *Measure Types* is "Nominal Measures". When this parameter is selected, the second parameter changes dynamically. The parameters tab changes and presents "Nominal Measures" as label, and this parameter has the suited algorithms for textual measure type as options.

Nominal measure algorithms are described below. Considering "e" as number of attribute for which both examples have equal and non-zero values, "u" the number of attribute for which both examples have not equal values and "z" the number of attribute for which both examples have zero values, the available algorithms are:

1. **NominalDistance:** Distance of two values is 0 if both values are the same and 1 otherwise.
2. **DiceSimilarity:** With the above-mentioned definitions the DiceSimilarity is 2*e/(2*e+u)
3. ***Jaccard*Similarity:** With the above-mentioned definitions the *Jaccard*Similarity is e/(e+u)
4. **KulczynskiSimilarity:** With the above-mentioned definitions the KulczynskiSimilarity is e/u
5. **RogersTanimotoSimilarity:** With the above-mentioned definitions the RogersTanimotoSimilarity is (e+z)/(e+2*u+z)
6. **RussellRaoSimilarity:** With the above-mentioned definitions the RussellRaoSimilarity is e/(e+u+z)
7. **SimpleMatchingSimilarity:** With the above-mentioned definitions the SimpleMatchingSimilarity is (e+z)/(e+u+z)

Jaccard algorithm is broadly used in text similarity retrieval (*LEVENSHTEIN*, 1966) (YUNG-SHEN LIN *et al.*, 2014), and the algorithm most often used for document comparison (TAN *et al.*, 2006). It compares two strings and retrieves an index that shows how similar both strings are. The similarity indexes retrieved as a result of the execution of similarity algorithms have a range from 0 to 1 and can be interpreted in percentages. This operator compares each document to all other documents (n^2). For example, if there are 25 examples in the given dataset, there will be 625 (i.e., (25*25)) similarity comparisons in the resultant similarity table. This operator is connected to another *Set Role* operator, which has a different responsibility in this step of process.

Set Role (2): This operator sets roles for specific attributes. The input information for this operator is the similarity table and its four columns. In this *Set Role(2)* operator, column *FIRST_ID* is set to “label” and column *SECOND_ID* is set to “prediction”. The *label* attribute serves as a target for comparison, and the *prediction* attribute is the prediction of a process. In other words, this means that the information on the *SECOND_ID* is the expected prediction and the *FIRST_ID* column is the base information for the prediction. In this dissertation, the information we are studying is Stack Overflow Posts. This means both columns should present the Stack Overflow Post associated to that compared project task. This way we can evaluate if project tasks with a high degree of similarity share the same Stack Overflow Post, in case they are equal in both columns.

Filter Examples: is the operator that sets a threshold of similarities. We defined a threshold of 50% similarity (similarity index >= 0.5).

Performance: *Performance* operator is used for statistical performance evaluation of classification tasks. This operator delivers a list of performance criteria values of the classification task. The classification task is the similarity extraction and that classification has the instances of data we wish to evaluate with the performance operator. To use this operator, it is mandatory to set roles to attributes from the similarity table as “label” and “prediction” roles, for which the *Set Role(2)* operator was responsible. The “label” attribute stores the actual observed values whereas the “prediction” attribute stores the values of *label* predicted by the classification process under analysis. This operator is connected to the “res” port of the process, indicating the end of the process. The output of this operator is a confusion matrix of the similarity table, and all metrics are calculated from this confusion matrix. The confusion matrix has two dimensions: label and prediction. It allows visualizing the performance of the algorithm. Each row (first dimension) represents the labels and the columns (second dimension) represent the predictions (or vice-versa). For example, if a system is trained to distinguish between elements chairs and pens, and there are 5 chairs and 2 pens, the resulting confusion matrix can look like the one presented in Table 4.8.

Table 4.8: Confusion matrix example.

		Actual (label)	
		Chair	Pen
Predicted	Chair	3	2
	Pen	2	0

From this table, we can conclude that from 5 occurrences of chairs, 3 were predicted right. And for 2 occurrences of pens, 0 were predicted right. We should be looking for the row where the label and prediction are the same elements. This last operator is connected to the *res* port, ending the process construction.

For didactic purposes, we divided the methodology step #6 into two chapters. Chapter 4 comprises the entire process implementation, including the operators responsible for evaluating, and the details on the *RapidMiner* operators used in the implementation. Chapter 5 continues the study description, as it evaluates the dataset sample collected and presents the results.

4.5 Conclusion

This chapter investigated project task context elements and proposed an implementation of a data mining process that is able to load a dataset of project tasks and retrieve statistical results with the possibility of reusing curated Stack Overflow Posts associated to project tasks when project tasks are similar (have a high similarity index). This implementation fulfills two of the specific objectives of this dissertation: **Identification of project task contexts** and **Implementation of a process to retrieve project task similarities.**

Next chapter continues the study, as it presents the evaluation of the process implemented in Chapter 4.

5 Evaluation

In this chapter, the process implemented in Chapter 4 will be evaluated. A sample using data from industry is used in this evaluation. There are two different results observed: the comparison of the results with the related work, and the results regarding context combinations. The evaluation methodology is presented, as well as all results obtained.

5.1 Introduction

In Chapter 4, a process that is able to retrieve similarities between project tasks and evaluate curated Stack Overflow reuse was presented. This process is an implementation of a data process in *RapidMiner*, and includes features to read data, select what are the project task context elements, create a similarity table and verify the results of the implementation, presenting metrics. In this chapter, we present the execution of the process built in Chapter 4 and the results provided by the process implementation. There are two evaluations result of this implementation: in the first, we collect results from the process executed, having as input the project task contexts identified in Section 4.4.1, using a sample gathered from a company. In the second evaluation, using the same sample, different project task contexts are combined and metrics of each different combination are provided. Evaluation variations are possible due to the offline experiment characteristic, which enables controlled data manipulation by changing the presence of variables of the dataset. Applying variables variations within the same set of tasks allows building basis for comparisons and analysis of diverse formats of methods or metrics (EKSTRAND *et al.*, 2010).

Offline experiments are usually performed to evaluate recommender systems. There are challenges on selecting metrics to quantitatively evaluate systems that suggest data based on existing information, as there is a lack of standardization of a metric in collaborative filtering systems (HERLOCKER *et al.*, 2004). Accuracy is a metric commonly used in this scenario (SHANI & GUNAWARDANA, 2011). According to these authors and the measures from our Related Work (S3, S5, S6, S7, S8), accuracy and precision are common metrics used for evaluating collaborative filtering systems. Due to the characteristic of our implementation - having a dataset with preset values allowing multiple tests and results comparisons - the evaluation for this proposal is an offline evaluation. Although it is possible to run multiple tests and there is no need for people involved in the experiment, there are two weaknesses regarding offline evaluations: the lack of user ratings and the fact they are limited to objective evaluation. No offline analysis can determine if users prefer a system because of the quality of its recommendations or because it has a good interface, for example.

The following sections present details regarding the evaluation metrics and the methodology, objectives, its execution, results gathered and a discussion regarding the obtained results, as well as the threats to validity identified.

5.2 Metrics

Precision is the proportion of Predicted Positive cases that are correctly Real Positives (POWERS, 2011). The equation for Precision is presented below in Equation 5.1.

$$Precision = \frac{relevant\ results\ returned}{number\ of\ results\ returned}$$

Equation 5.1: Precision formula.

The accuracy is calculated by taking the percentage of correct predictions over the total number of examples. Correct predictions (accuracy) means the examples where the value of the prediction attribute is equal to the value of label attribute. Regarding Precision, *RapidMiner* provides a mean of the weights of precisions extracted. For this dissertation, each precision has the same weight, meaning the metric, although presented as *Weighted Mean Precision*, or *WMP*, is the mean precision, as there are no weights defined.

Regarding Accuracy, researchers aim to find solutions that provide better predictions, because there is an assumption that users will prefer a system that predicts better. The accuracy is the most discussed property in the literature of recommender systems. This metric is usually interface-independent and can be used in an offline experiment. The accuracy of the suggestions is measured by the prediction accuracy of a user study, and is closer to the true accuracy in the real system (SHANI & GUNAWARDANA, 2011).

Next section presents the methodology of the evaluation, providing the research questions formulated as well as the evaluation objectives.

5.3 Methodology

We base our evaluation according to the guidelines proposed by (SHANI & GUNAWARDANA, 2011). We have implemented part of the methodology that addresses offline experiments, proposed by these authors. It is expected from an offline evaluation that the data used matches as closely as possible to the data the system will use when deployed, because the introduction of bias is a concern. When data is collected from an existing system where there is no recommendation available, there is a tendency to exclude and pre-filter the data to exclude items with low costs (SHANI & GUNAWARDANA, 2011). To avoid this, we decided not to collect random samples or alter any of the data content, and we also believe the project task context combination evaluation mitigates the possible inclusion of bias when selecting project task context element variables. We defined a hypothesis, controlling variables and generalization power to serve as guidelines for this evaluation, also as part of the evaluation methodology proposed by (SHANI & GUNAWARDANA, 2011).

Hypothesis: Project task similarity can provide effective suggestions of curated Stack Overflow Posts. We test this hypothesis by verifying whether highly similar tasks share the same Stack Overflow Post.

Controlling Variables: Considering this study uses only one dataset, there is no concern regarding having fixed controlled variables. In fact, we propose a study considering different variable combinations to analyze the effects of the absence or presence of variables on precision and accuracy.

Generalization Power: Considering this study uses only one dataset, we believe the results are indications rather than generalization. All results found in this research concern this dataset or a dataset with very similar characteristics, and can be used as indications for general conclusions.

This experiment aims at gathering precision and accuracy metrics for the proposed study in this dissertation. The study proposes indications that there is a correlation between project tasks with curated Stack Overflow Posts, meaning that it is very likely that project tasks with high similarity share the same curated Stack Overflow Post. We present the goal of this evaluation in a GQM (Van Solingen et. al., 2002) approach format:

analyze *precision and accuracy metrics*
with the purpose of *verifying the effectiveness of reusing curated Stack Overflow Posts*

***regarding** the identification of similar project tasks*
***from the point of view of** process implementation*
***in the context of** collected industry sample*

This objective being considered, the research questions for this study are:

RQ1: What is the precision and the accuracy metrics for the collected sample? It is important to verify these metrics in order to have means of comparison with the related work, while verifying the effectiveness of considering similar project tasks to reuse curated Stack Overflow Posts.

RQ2: What are the impacts in precision and accuracy when different context elements are combined? It is important to evaluate different project task context combinations because these contexts can vary in each project. A project can maintain records of processes and another project might not, for example. Given this variation, it's important to understand the impacts of different project task context combinations.

In next sections, we present the execution of the process built in Chapter 4. The following sections present details regarding the dataset used during the execution of the process, and also the Results, Discussion and Threats to Validity of this evaluation.

5.4 Executing the implemented *RapidMiner* process

To execute the implemented process presented in Chapter 4, a dataset with project tasks have to be loaded in the process (*Retrieve* operator). We gathered a dataset from a company in Brazil. This company has been developing software products for more than 20 years, and has a total of 30 employees. The software development projects follow agile guidelines and the project tasks are managed on both a project management tool and a white board. We were able to gather 25 project tasks with associated Stack Overflow Posts to each of the 25 tasks. Other contacts from our professional network were contacted, but other companies don't concentrate their knowledge into the project management tool, or don't even have processes to control the information that was used to support project tasks at all. The tasks needed for this study should have two mandatory requirements: all of them should have at least one Stack Overflow Post associated and different tasks should share the same Stack Overflow Post, once our goal is to evaluate if different tasks have context similarities, and therefore could share the same Stack Overflow Post. All project tasks are listed in Appendix C – Dataset Sample. This dataset contains the following information for each task:

1. **TaskID:** The ID generated by the Project Management Tool that identifies a task uniquely.
2. **StackOverflow:** The ID of the Stack Overflow Post associated to the task, the post that was used to support the project task.
3. **Project:** The name of the Project the task belongs to.
4. **Category:** A classification for the task. The categories are determined by the project managers, and can be: Corrective, Planned, Not Planned, a name of a specific branch/iteration and other customized information to characterize a task and its purpose.
5. **Iteration:** The name of the iteration the task belongs to.
6. **Title:** The title of the task. This is filled at discretion of the responsible.
7. **Description:** The description of the task. This is filled at discretion of the responsible. The description should contain details about the task.
8. **Process:** The Process Activity related to the task.

For this study, as we have concluded in Section 4.4.1, technological information is an information intrinsic to software development. Because of that, we asked the software developers of this company to also inform us the technological characteristics of the projects and each task, and that information were added to the dataset as Tags (Task Tags and Project Tags). With this provided dataset added as input to the process, we were able to execute the implementation in *RapidMiner*.

The dataset is uploaded in *Retrieve* operator, and in *Select Attributes* operator, the information from the dataset that will be used to retrieve similarity indexes are chosen. *TaskID* and *StackOverflow* information from the dataset are not selected when retrieving the similarities. This is due to the fact *TaskID* is an identification number that is different for each project task and *StackOverflow* is the information we are trying to predict, and therefore should not be included in the similarity index retrieval. For the analysis of ***RQ1***, we also do not consider the *Iteration* information from the dataset, as it was not identified in Section 4.4.1, when project task contexts were identified. For **RQ2** we will consider *Iteration* in order to analyze the impact of the insertion of the information *Iteration*. *Set Role* operator indicates which information from the dataset will be predicted, in this case, *StackOverflow*. *StackOverflow* is set to the role "id", meaning this is the information the process is predicting. The *Data to Similarity Data* operator, which implements a similarity algorithm and generates as output a similarity table with all similarity indexes of compared project tasks, was configured to use *Jaccard* algorithm as parameter.

The second *Set Role* operator – presented in the model as *Set Role(2)* because there is already one *Set Role* operator - between the operators *Data to Similarity Data* and *Filter Examples* has as input the similarity table generated by the *Data to Similarity Data operator*. *Set Role(2)* operator needs to set roles for the attributes *FIRST_ID and SECOND_ID* for the generated similarity table. It then sets *FIRST_ID* to role “label” and *SECOND_ID* to role “prediction”. The *Filter Examples* operator is filtering the rows in the similarity table that are above or equal 50%, setting a similarity threshold for analysis. The *Performance* operator is used for statistical performance evaluation; therefore, it is responsible to retrieve the precision and accuracy results. *Performance* operator is connected to the “*res*” end port of the process, being the last operator in the implemented process. The output of this operator is a confusion matrix of the similarity table, and all metrics are calculated from this confusion matrix.

The results from the implemented evaluation process are presented in next section.

5.5 Results

After executing the evaluation with the dataset, the precision and accuracy are calculated, using *Jaccard* algorithm. Results are presented in Table 5.2. As results for this work, only Precision is compared to the Related Work. The confusion matrix result of the execution is presented in Table 5.1. The grey cells are the counts when the same Stack Overflow Posts are identified in the similarity table in both label and predictions attributes (correct prediction). Both dimensions contain Stack Overflow Post’s IDs from the dataset.

Table 5.1: Confusion matrix.

	25636091	25996758	29174164	21168521	2607289	11345926	25472378	28117615	13944222	15549931	38510879	25098307	class precision
25636091	27	3	0	0	2	2	0	0	0	0	0	0	79.41 %
25996758	3	24	0	0	0	0	0	0	0	0	0	0	88.89 %
29174164	0	0	10	1	0	0	0	0	0	0	0	0	90.91 %
21168521	0	0	1	1	0	0	0	0	0	0	0	0	50.00 %
2607289	2	0	0	0	1	1	0	0	0	0	0	0	25.00 %
11345926	2	0	0	0	1	1	0	0	0	0	0	0	25.0%
25472378	0	0	0	0	0	0	1	0	0	0	0	0	100%
28117615	0	0	0	0	0	0	0	1	0	0	0	0	100%
13944222	0	0	0	0	0	0	0	0	1	1	0	0	50%
15549931	0	0	0	0	0	0	0	0	1	1	0	0	50%
38510879	0	0	0	0	0	0	0	0	0	0	1	0	100%
25098307	0	0	0	0	0	0	0	0	0	0	0	1	100%
class recall	79.41%	88.89%	90.91%	50.00%	25.00 %	25.00%	100%	100%	50.00%	50.00%	100%	100%	

Table 5.2: Evaluation Results – RQ1.

Accuracy	**WMP (Precision)**
77.78%	**71.60%**

Table 5.2 presents the results for **RQ1**. The accuracy for the given dataset with the elements identified in Section 4.4.1 is **77.78%** and the mean of precisions is **71.60%**. These results consider the similarities above or equal 50%. Table 5.3 presents all collected results from the Related Work in order to be able to easily compare the results. All the metrics presented from Related Work are the highest scores gathered in each of the related works.

Table 5.3: Comparison of results with Related Work.

Related Work Articles	**Precision**	**Accuracy**
Wu *et al.*, 2018 (S3)	24.32%	
Wang *et al.*, 2014 (S5)	62%	
Rahman *et al., 2014* (S6)	11%	88%
Correa & Sureka, 2013 (S7)	47:27%	
Ponzanelli *et al.*, 2013 (S8)	2.43%	18.92%
Current Dissertation	**77.78%**	**71.60%**

To answer **RQ2**, we evaluated different combination of project task contexts. The precision and accuracy metrics were extracted for each combination, and results are presented in Table 5.4. The *Select Attribute* operator was an important asset for this evaluation in particular, considering the selection of the elements was facilitated by *RapidMiner* interface usage of this specific operator. Table 5.4 presents the attributes' combinations selected, the precision and accuracy extracted, and the attribute change summary, as it might not be easy to perceive from the attribute list which attribute was selected and which one was not selected in the combination.

As presented in Table 5.4, when considering the project task context elements originally identified, the outcomes of the evaluation are higher precision and accuracy among all combinations. When considering the *Iteration* context, which was informed in the dataset sample gathered in industry, precision and accuracy are lower. This can indicate that either this context element is bad for the approach, or in the dataset sample, the information was not properly set. The same is perceived when removing other elements of the context, such as *Process*, *ProjectTags* and *TaskTags*, and *Title* and *Description*. More information regarding other context combinations are presented in Appendix F – Combinations on project task context elements.

Table 5.4: Evaluation of combinations of project task context elements – RQ2.

Selected Attributes		**Precision**	**Accuracy**	**Attribute(s) change**
Category Description Process Project	Project Tags Task Tags Title	77.78%	71.60%	-
Category Description Process Iteration	Project Project Tags Task Tags Title	61.17%	70%	Included Iteration
Category Description Project	Project Tags Task Tags Title	48.01%	54.69%	Removed Process
Category Description	Process Project Title	69.81%	66.67%	Removed ProjectTags TaskTags
Category Process	Project Project Tags Task Tags	45.64%	37.12%	Removed Title Description

The results also indicate that the hypothesis of the evaluation ***Project task similarity can provide effective suggestions of curated Stack Overflow Posts*** is correct. There is an indication project task similarity can provide accurate associations between project tasks and curated Stack Overflow Posts, as the prediction and accuracy are as high as 70%. This result, above of most of our Related Work, also indicates that the predictions performed by the process are among a reasonable scale.

Next section presents the threats identified in this evaluation and means to mitigate the threats.

5.6 Threats to validity

Following the model proposed by (WOHLIN *et al.*, 2012), we present threats to the validity of this evaluation in this section. According to this model, there are events that can impact or limit the study. The events and threats of each event are described below.

Internal Threats: events not controlled by the researcher that can produce distortions in the expected result. In this study, a threat to internal validity is the small sample we were able to get. A way to mitigate this would be gather more samples from other companies or implement the solution in companies and verify ratings for suggestions/recommendations. The fact we have a dataset made the offline experiments possible. Although they are controlled experiments, they have as drawback, no user interaction (SHANI & GUNAWARDANA, 2011).

External Threats: events that cause jeopardy to the generalization of the study results. A feasibility study does not aim to have its results generalized to other contexts; even so, some threats to external validity have been identified. The first is with respect to the representativeness of the study, which was executed with only one sample of a software organization. To minimize this threat, real project tasks were collected from a software development organization and the proposal was based on research in the literature and tools that are largely used by industry, as well as the construction of a model using widespread algorithms and metrics.

Construction Threats: events that may impact correct measures in the study. One of the threats to the validity of construction would be the incorrect definition of the measures in the study, as well as the selection of methods that could harm its collection. To mitigate this type of threat, the measures used in the study were based on highly referenced literature for evaluation of recommendation systems.

Conclusion Threats: events that hamper the establishment of statistical relationships. In this evaluation study, statistical tests were used to analyze the data, and compared to the related work. Therefore, the results can be considered conclusive: we believe they are indications of the applicability of the proposed solution. Also, the fact that the study was conducted within one industry scenario might indicate the tendency that the same developers fill the text of project tasks, meaning there might occur a standardization in project task texts. This is mitigated by selecting tasks from different projects where different teams work on, but there is still no guarantee this problem can’t occur.

5.7 Conclusion

This section performed two different evaluations proposed by this dissertation. The first arises from **RQ1: What is the precision and the accuracy metrics for the collected sample?** The answers are 77.78% and 71.60%, respectively, indicating effectiveness when considering similar project tasks to reuse curated Stack Overflow Posts. The second proposed evaluation, which arises from **RQ2: What are the impacts in precision and accuracy when different context elements are combined?**, has as results that the removal of some of the project task context elements decrease the precision and accuracy, when comparing to a baseline containing the specific group of contexts identified in Section 4.4.1. The same developer did not create the project tasks used in the dataset, as the projects are different and different team members work in each project. Two developers identified the Stack Overflow Posts associated to the tasks within their personal notes, or project management tool history and code versioning programs.

6 Conclusion

This chapter concludes the dissertation, presenting the final considerations and contributions, discussing limitations and future work.

6.1 Introduction

This dissertation presented research on reusing Stack Overflow Posts during software development, through the identification of similar project tasks. First, it introduces the research motivation and problem, in Chapter 1. Then, it discusses central concepts behind software projects, how project tasks are conceived and presents project management tools that are widely used to manage software development. Then it discusses Q&A websites in general and narrows to Stack Overflow, which is nowadays one of the most important sources for software development support. The dissertation introduces data mining and recommender systems concepts, mentioning widely implemented algorithms and tools used on mining data, which can be used for recommendation purposes.

After concluding conceptualization – presented in Chapter 2 - an investigation on if the idea of reusing Stack Overflow Posts using project task similarity would be feasible initiated. We performed a preliminary assessment, using data from one company, and the results indicated the study was worth pursuing further. Besides this conclusion, we also coined a new term in software development: *curation*. As we observed, there is a relevant part of software developers' daily routine they invest in searching and selecting information to support their development. We simplified the *searching* and *selection* effort into one word: *curation*.

The objectives for the research were defined and in order to accomplish the first specific objective ***Obtaining the state-of-the-art of existing approaches that associate software development projects with Stack Overflow***, a Systematic Mapping Study was performed. The first study was performed in January 2018, and then it was updated in September 2018. After performing the Systematic Mapping Study, we were able to have a broad view on researches that have implemented solutions to associate Stack Overflow with the development environment. Other than that, we got insights on what to expect regarding sample sizes, metrics mostly used in the field, and we confirmed the importance of Stack Overflow, as there are many, and recent, papers focusing in Stack Overflow data. This study is described in Chapter 3.

The second and third specific objectives **Identification of project task contexts** and **Implementation of a process to retrieve project task similarities** were accomplished by furthering the study development. An investigation on project task context elements, with an *ad-hoc* literature review and project management tools was performed. A set of project task context elements was identified and described. Also, we implemented a data mining process. This process and each of its operators were created according to the objectives defined in this dissertation. These studies, along with the preliminary assessment, are described in Chapter 4.

The fourth and last specific objective was to **Evaluate** the implemented process. A dataset collected in a company that develops software for over 10 years was used in this evaluation. This dataset is composed by project tasks and each task has one Stack Overflow Post associated to it. Two research questions guided the evaluation. The first, **RQ1: What is the precision and the accuracy metrics for the collected sample?** determined the implemented process should retrieve both metrics. The accuracy and precision (**77.78%** and **71.60%**, respectively) were the results of this evaluation. For the second evaluation proposed, **RQ2: What are the impacts in precision and accuracy when different context elements are combined?** different combinations of the project task context elements were proposed. It is possible to conclude that the original combination of contexts, as suggested by the study described in Section 4.4.1, is the best combination for the given dataset, considering both accuracy and precision are the highest, when comparing to the other combinations. The evaluation is presented in Chapter 5.

We conclude that by associating curated Stack Overflow Posts with the specific project task that prompted the developer to look for support, it is possible to reuse the curation effort when similar project tasks are identified.

6.2 Contributions

The main contribution of this work is to provide a study and quantitative and qualitative results on the possibility to capture and reuse curated Stack Overflow Posts during software development, using existing project task context information. Also, we were able to perceive the correlation between curated Stack Overflow posts and project tasks. The following secondary contributions may be highlighted:

- **Preliminary Assessment:** in order to verify the feasibly of the intended study, a qualitative preliminary assessment was performed. A company in the Brazilian software project industry scenario participated in this study, that allowed us to receive insights on the usefulness of the intended research development.
- **Systematic Mapping Study:** a Systematic Mapping Study on approaches that associate Stack Overflow to software development was performed. The result of this study is a detailed mapping on association strategies, input and output information, evaluation methods and results for the selected papers.
- **Project Task Context investigation:** we investigated, in literature and project management tools, what is the context for project task that would aid similarity retrieval. A list with the contexts identified is the result of this investigation.
- **Data Mining Process Implementation:** this contribution allows both development and evaluation of the study. A process using a set of operators was implemented, and through this implementation, it was possible to compare a set of project tasks, retrieve their similarity and evaluate the precision and accuracy of the most similar project tasks. This same process can be used to evaluate different datasets.
- **Study on accuracy and precision of different project task context combinations:** this study allowed the verification what was the best set of contexts chosen for the tested dataset.
- **Stack Overflow database model**: Stack Overflow provides a list of tables available for querying content. From this list, we provided a entity-relation model, which can aid decisions regarding Stack Overflow data.

Other contributions:

- Paper publication in the Iberoamerican Conference of Software Engineering, 2018, main Software Engineering track.
- Collaboration with University of Waterloo, provided by ELAP (Emerging Leaders of the Americas) Scholarship.
- Teaching internship in Software Quality course (2016.3, 2017.3 and 2018.3).

6.3 Limitations

Some limitations were identified, considering the execution of the studies. The main limitations identified were:

- **Need for human intervention in the curation result association**: the fact that developers need to proactively associate the selected Stack Overflow post after curation can hamper the study functionality, as this step is completely human dependent.
- **Sample for evaluation**: Although the sample size of the dataset used to perform the evaluation in this dissertation is small, we identified small samples also in Related Work. Even though, this is considered a limitation of this dissertation. Also, we were able to evaluate using only one sample. The ideal situation would be to have at least three different samples. The results are considered indications and cannot be generalized. Still regarding the sample and project tasks in it, there is a concern in how project tasks are created, who writes the text of tasks and how detailed the task is. Considering the dataset used, the company maintain a serious quality standard regarding tasks for two main reasons: clients have full access to the project management tool and the company, although not officially certified by an institute that guarantee the level of maturity of processes, is strict in relation to maintaining descriptions of quality artifacts, emphasizing the verbal and written communication of artifacts.
- **Preliminary Assessment**: this study was not performed in other companies or with different set of data. Therefore, results are considered indications and cannot be generalized, as the sample might not representative enough.
- **Systematic Mapping Study:** the use of only one research database for the Systematic Mapping Study can result in papers not being considered.
- **Curation and scalability:** when considering scalability of the proposal, the list of suggested Stack Overflow Posts can get long when more curation results are associated with time. This can lead to a new curation of the already curated results. Ranking curation results and improving the role of context elements when finding similar tasks can mitigate this problem. If real similar tasks are found, the amount of curated results suggested tends to have increased quality.

6.4 Future Work

Considering the conclusions and the research conducted in this dissertation, it is possible to suggest future work opportunities.

Future work might involve development of a recommendation tool using the strategy proposed in this dissertation. A tool would allow the incorporation of rating mechanisms for given suggestions. The tool was not developed in this work, as we intended to focus on the study of the impacts of associating the curation information with development. The development of a tool would demand a considerable amount of effort and there wouldn’t be enough time to test the tool in a proper environment, such as an industry scenario, or tested by a team with a proper amount of people to participate.

A deeper understanding of the role of project task context elements should be pursued, in a broader perspective. This would allow better combinations of contexts, as well as the possibility to define weights for given contexts. This was not possible in this study, since this proposal focus on the possibility to achieve results comparable to some of the Related Work, when considering curation to associate software development with Stack Overflow. However, the results of the second evaluation proposed indicate that context elements have deep impacts on results, leading to conclude each context element can have a different importance, represented by weights.

Finally, a comparison of different algorithms would allow verifying implementations that can possibly perform better in this software project task scenario.

References

ANDERSON, Ashton et al. Discovering value from community activity on focused question answering sites: a case study of stack overflow. In: **Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining.** ACM, 2012. p. 850-858.

ANTUNES, Bruno; CORDEIRO, Joel; GOMES, Paulo. An approach to context-based recommendation in software development. In: **Proceedings of the sixth ACM conference on Recommender systems.** ACM, 2012. p. 171-178.

AURUM, A.; DANESHGAR, F.; WARD, J. Investigating Knowledge Management practices in software development organisations – An Australian experience. **Information and Software Technology**, Amsterdam, v. 50, n. 6, p. 511-533, 2008.

BAKER, K.; VERSTOCKT, S. Cultural Heritage Routing. **Journal on Computing and Cultural Heritage (JOCCH)**, v. 10, n. 4, p. 1-20, Jul 31, 2017.

BARROS-JUSTO, J.L.; BENITTI, F.B.V.; CRAVERO-LEAL, A.L. Software patterns and requirements engineering activities in real-world settings: A systematic mapping study. **Computer Standards & Interfaces**, v. 58, p. 23-42, 2018.

BRANDT, Joel et al. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In: **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.** ACM, 2009. p. 1589-1598.

CAMPOS, E.C.; SOUZA, L.B.L.; MAIA, M.D.A. Searching crowd knowledge to recommend solutions for API usage tasks. **Journal of Software: Evolution and Process**, Chichester, v. 28, n. 10, p. 863-892, Oct 2016.

CORREA, Denzil; SUREKA, Ashish. Integrating issue tracking systems with community-based question and answering websites. In: **Software Engineering Conference (ASWEC), 2013 22nd Australian.** IEEE, 2013. p. 88-96.

ČUBRANIĆ, Davor et al. Learning from project history: a case study for software development. In: **Proceedings of the 2004 ACM conference on Computer supported cooperative work.** ACM, 2004. p. 82-91.

CAMPOS, Andre LN; OLIVEIRA, Toacy. Software processes with BPMN: an empirical analysis. In: **International Conference on Product Focused Software Process Improvement.** Springer, Berlin, Heidelberg, 2013. p. 338-341.

LESKOVEC, Jure; RAJARAMAN, Anand; ULLMAN, Jeffrey David. **Mining of massive datasets**. Cambridge university press, 2014.

DA ROCHA, ANA REGINA CAVALCANTI; J.C. MALDONADO; K.C. WEBER. **Qualidade de software: teoria e prática**: Prentice Hall, 2001.

DE SOUZA, Lucas BL; CAMPOS, Eduardo C.; MAIA, Marcelo de A. Ranking crowd knowledge to assist software development. In: **Proceedings of the 22nd International Conference on Program Comprehension**. ACM, 2014. p. 72-82.

DI CICCIO, Claudio; MARRELLA, Andrea; RUSSO, Alessandro. Knowledge-intensive processes: characteristics, requirements and analysis of contemporary approaches. **Journal on Data Semantics**, v. 4, n. 1, p. 29-57, 2015.

DOTZLER, Georg; VELDEMA, Ronald; PHILIPPSEN, Michael. Annotation support for generic patches. In: **Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering**. IEEE Press, 2012. p. 6-10.

EL-KORANY, ABEER. (2013). Integrated Expert Recommendation Model for Online Communities. **International journal of Web & Semantic Technology**. 4. 10.5121/ijwest.2013.4402.

EKSTRAND, M.D.; RIEDL, J.T.; KONSTAN, J.A. Collaborative Filtering Recommender Systems. **Human–Computer Interaction**, v. 4, n. 2, p. 81-173, 2010.

FEILER, Peter H.; HUMPHREY, Watts S. Software process development and enactment: Concepts and definitions. In: **Software Process, 1993. Continuous Software Process Improvement, Second International Conference on the**. IEEE, 1993. p. 28-40.

FILIPIAK, D.; WECEL, K.; FILIPOWSKA, A. Semantic Annotation to Support Description of the ArtMarket. **SEMANTiCS 2015 Vienna, Austria**, v. 23, n. 4, p. 291-305, 2006.

FUGGETTA, A. **Software process** Chichester [u.a.]: Wiley, 1996.

FUMIN, Sun et al. Recommendflow: Use Topic Model to Automatically Recommend Stack Overflow Q&A in IDE. In: **International Conference on Collaborative Computing: Networking, Applications and Worksharing**. Springer, Cham, 2016. p. 521-526.

GAROUSI, Vahid; FELDERER, Michael; MÄNTYLÄ, Mika V. The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature. In: **Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering**. ACM, 2016. p. 26.

GASPARIC, M.; JANES, A. What recommendation systems for software engineering recommend: A systematic literature review. **The Journal of Systems and Software**, New York, v. 113, p. 101-113, Mar 1, 2016.

GASPARIC, M.; MURPHY, G.C.; RICCI, F. A context model for IDE-based recommendation systems. **The Journal of Systems and Software**, v. 128 Jun 1, 2017.

GRECO, Chase; HADEN, Tyler; DAMEVSKI, Kostadin. StackInTheFlow: behavior-driven recommendation system for stack overflow posts. In: **Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings**. ACM, 2018. p. 5-8.

HAN, J.; MINING, M.K.D. Concepts and Techniques. **Morgan Kauffman, San Francisco** 2000.

HERLOCKER, J. et al. Evaluating collaborative filtering recommender systems. **ACM Transactions on Information Systems (TOIS)**, New York, v. 22, n. 1, p. 5-53, Jan 1, 2004.

HOLMES, Reid et al. Automatically recommending triage decisions for pragmatic reuse tasks. In: **Proceedings of the 2009 IEEE/ACM International Conference**

on Automated Software Engineering. IEEE Computer Society, 2009. p. 397-408.

ICHII, Makoto et al. Software component recommendation using collaborative filtering. In: **Proceedings of the 2009 ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation**. IEEE Computer Society, 2009. p. 17-20.

JAFFEE, L. Finding Growth in a Mature Business. **Medialine**, p. 6, Oct 1, 2005.

KAJORNBOON, Annabel Bhamani. Using interviews as research instruments. **E-journal for Research Teachers**, v. 2, n. 1, p. 1-9, 2005.

KERSTEN, Mik; MURPHY, Gail C. Mylar: a degree-of-interest model for IDEs. In: **Proceedings of the 4th international conference on Aspect-oriented software development**. ACM, 2005. p. 159-168.

KOCHHAR, Pavneet Singh. Mining testing questions on stack overflow. In: **Proceedings of the 5th International Workshop on Software Mining**. ACM, 2016. p. 32-38.

LEVENSHTEIN, V.I. **Binary codes capable of correcting deletions, insertions, and reversals.**, v. 10, n. 8, p. 707-710, 1966.

LINDVALL, M.; RUS, I. Knowledge management for software organizations. In: Anonymous **Managing software engineering knowledge**: Springer, 2003.

LIU, Xiaoning et al. Expsol: Recommending online threads for exception-related bug reports. In: **2016 23rd Asia-Pacific Software Engineering Conference (APSEC)**. IEEE, 2016. p. 25-32.

MAMYKINA, Lena et al. Design lessons from the fastest q&a site in the west. In: **Proceedings of the SIGCHI conference on Human factors in computing systems**. ACM, 2011. p. 2857-2866.

RAHMAN, Mohammad Masudur; ROY, Chanchal K. Surfclipse: Context-aware meta-search in the ide. In: **Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on**. IEEE, 2014. p. 617-620.

VALLE, Arthur M.; SANTOS, Eduardo AP; LOURES, Eduardo R. Applying process mining techniques in software process appraisals. **Information and software technology**, v. 87, p. 19-31, 2017.

MELO, G., TELEMACO, U., OLIVEIRA, T., ALENCAR, P., COWAN, D. Towards using task similarity to recommend Stack Overflow posts. **Avances en Ingenieria de Software a Nivel Iberoamericano**, CIbSE 2018, pp. 199-211.

MEYER, André N. et al. The work life of developers: Activities, switches and perceived productivity. **IEEE Transactions on Software Engineering**, v. 43, n. 12, p. 1178-1193, 2017.

MÜNCH, Jürgen et al. **Software process definition and management**. Springer Science & Business Media, 2012.

PAI, Madhukar et al. Systematic reviews and meta-analyses: an illustrated, step-by-step guide. **The National medical journal of India**, v. 17, n. 2, p. 86-95, 2004.

PETERSEN, Kai; VAKKALANKA, Sairam; KUZNIARZ, Ludwik. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and Software Technology**, v. 64, p. 1-18, 2015.

PILLAT, Raquel M. et al. BPMNt: A BPMN extension for specifying software process tailoring. **Information and Software Technology**, v. 57, p. 95-115, 2015.

PONZANELLI, Luca; BACCHELLI, Alberto; LANZA, Michele. Leveraging crowd knowledge for software comprehension and development. In: **Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on**. IEEE, 2013. p. 57-66.

POWERS, David. (2008). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Mach. Learn. Technol.. 2.

PIORKOWSKI, David et al. Reactive information foraging: An empirical investigation of theory-based recommender systems for programmers. In: **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems**. ACM, 2012. p. 1471-1480.

PONZANELLI, Luca; BACCHELLI, Alberto; LANZA, Michele. Seahawk: Stack overflow in the ide. In: **Proceedings of the 2013 International Conference on Software Engineering**. IEEE Press, 2013. p. 1295-1298.

PONZANELLI, Luca et al. Mining StackOverflow to turn the IDE into a self-confident programming prompter. In: **Proceedings of the 11th Working Conference on Mining Software Repositories**. ACM, 2014. p. 102-111.

RAHMAN, Mohammad Masudur; YEASMIN, Shamima; ROY, Chanchal K. Towards a context-aware IDE-based meta search engine for recommendation about programming errors and exceptions. In: **Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on**. IEEE, 2014. p. 194-203.

CANALES, Raquel Fonseca; MURILLO, Edgar Casasola. Evaluation of entity recognition algorithms in short texts. **CLEI ELECTRONIC JOURNAL**, v. 20, n. 1, 2017.

REIS, Carla Alessandra. **Uma abordagem flexível para execução de processos de software evolutivos. 2003. 267 f**. 2003. Tese de Doutorado. Tese (Doutorado)-Curso de Ciência Da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre.

MINELLI, Roberto; MOCCI, Andrea; LANZA, Michele. I know what you did last summer: an investigation of how developers spend their time. In: **Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension**. IEEE Press, 2015. p. 25-35.

ROBILLARD, M.; WALKER, R.; ZIMMERMANN, T. Recommendation Systems for Software Engineering. **IEEE Software**, Los Alamitos, v. 27, n. 4, p. 80-86, 2010.

SANTOS, Renata et al. Mining software development process variations. In: **Proceedings of the 30th Annual ACM Symposium on Applied Computing**. ACM, 2015. p. 1657-1660.

SAHU, T.P.; NAGWANI, N.K.; VERMA, S. An empirical analysis on reducing open source software development tasks using stack overflow. **Indian Journal of Science and Technology**, v. 9, n. 21 2016.

SANTHANAKUMAR, M.; COLUMBUS, C.C. Web usage based analysis of web pages using rapidminer. **Wseas Transactions On Computers**, v. 14 2015.

SANTORO, Flávia Maria; BRÉZILLON, Patrick; DE ARAUJO, Renata Mendes. Context dynamics in software engineering process. In: **International Conference on Computer Supported Cooperative Work in Design**. Springer, Berlin, Heidelberg, 2006. p. 377-388.

SAWADSKY, Nicholas; MURPHY, Gail C. Fishtail: from task context to source code examples. In: **Proceedings of the 1st Workshop on Developing Tools as Plug-ins**. ACM, 2011. p. 48-51.

SCHLITTER, Nico et al. Distributed data analytics using *RapidMiner* and BOINC. In: **Proceedings of the 4th *RapidMiner* Community Meeting and Conference (RCOMM 2013)**. 2013. p. 81-95.

SHANI, G.; GUNAWARDANA, A. Evaluating Recommendation Systems. In: Anonymous **Recommender Systems Handbook**. Boston, MA: Springer US, 2011.

TAN, P.; M. STEINBACH; V. KUMAR. **Introduction to data mining**. Pearson internat. ed. ed. Boston ; Munich [u.a.]: Pearson Addison Wesley, 2006.

VAN SOLINGEN, R. et al. Goal question metric (gqm) approach. **Encyclopedia of software engineering** 2002.

VASANTHAPRIYAN, Shanmuganathan; TIAN, Jing; XIANG, Jianwen. A survey on knowledge management in software engineering. In: **Software Quality, Reliability and Security-Companion (QRS-C), 2015 IEEE International Conference on**. IEEE, 2015. p. 237-244.

VIRIYAKATTIYAPORN, Petcharat; MURPHY, Gail C. Challenges in the user interface design of an IDE tool recommender. In: **Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering**. IEEE Computer Society, 2009. p. 104-107.

WANG, Tao et al. Linking stack overflow to issue tracker for issue resolution. In: **Proceedings of the 6th Asia-Pacific Symposium on Internetware on Internetware**. ACM, 2014. p. 11-14.

WANG, Tao et al. Automatic knowledge sharing across communities: a case study on android issue tracker and stack overflow. In: **Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium on**. IEEE, 2015. p. 107-116.

WARR, Frederic Weigand; ROBILLARD, Martin P. Suade: Topology-based searches for software investigation. In: **Software Engineering, 2007. ICSE 2007. 29th International Conference on**. IEEE, 2007. p. 780-783.

YANG, Di; HUSSAIN, Aftab; LOPES, Cristina Videira. From query to usable code: an analysis of stack overflow code snippets. In: **Proceedings of the 13th International Conference on Mining Software Repositories**. ACM, 2016. p. 391-402.

LIN, Yung-Shen; JIANG, Jung-Yi; LEE, Shie-Jue. A similarity measure for text classification and clustering. **IEEE transactions on knowledge and data engineering**, v. 26, n. 7, p. 1575-1590, 2014.

WOHLIN, C. et al. **Experimentation in software engineering**: Springer Science & Business Media, 2012.

Appendix A – Dandelion + R Implementation Code

```
    #install.packages("jsonlite")  #Json get and post
    #install.packages("httr")
    #install.packages("stringr")

    library(jsonlite)
    library(httr)
    library(stringr)
    library(tm)
    library(SnowballC)
    #
    #
    # Importing the dataset
    dataset = read.delim('dataset6.tsv', quote = '', stringsAsFactors = FALSE )

    #Preparing the dataset - removing punctuation and numbers from specific
dataset fields - without tm - version dependent
    dataset$Title = gsub('[0-9]+','',dataset$Title)
    dataset$Description = gsub('[0-9]+','',dataset$Description)
    dataset$Title = gsub('[[:punct:] ]+',' ',dataset$Title)
    dataset$Description = gsub('[[:punct:] ]+',' ',dataset$Description)
    dataset$Project.Tags = gsub('[[:punct:] ]+',' ',dataset$Project.Tags)
    dataset$Task.Tags = gsub('[[:punct:] ]+',' ',dataset$Task.Tags)

    dataset$Title = chartr("áéíóúãõâêîôû", "aeiouaoaeiou", dataset$Title)
    dataset$Description = chartr("áéíóúãõâêîôû", "aeiouaoaeiou",
dataset$Description)
    dataset$Project = chartr("áéíóúãõâêîôû", "aeiouaoaeiou", dataset$Project)

    dataset$Title = sapply(dataset$Title, tolower)
    dataset$Description = sapply(dataset$Description, tolower)
    dataset$Project.Tags = sapply(dataset$Project.Tags, tolower)
    dataset$Task.Tags = sapply(dataset$Task.Tags, tolower)

    # Use only the 100th characters of description for simpliciy sake
    dataset$Description = substr(dataset$Description, 1,100)

    #Building Dandelion API URL
    url_part1 = "https://api.dandelion.eu/datatxt/sim/v1/?text1="
    url_part2 =" &text2="
    url_part3 =
"&token=1df1e8446ff14929a1a03c9489377722&bow=always&lang=pt"   #token for
user glauciamelo@gmail.com
    #url_part3 =
"&lang=pt&token=9951341d2ed648ffb907f78fba091066&bow=always"  #token for user
gmelodos@uwaterloo.ca

    #1
    #Dynamic Data Frame (Table) that receives similiarities indexes using all
contexts
```

```
        df_allContext = data.frame(Task1 = numeric(),
                    Task2 = numeric(),
                    method = character(),
                    similarityindex = character(),
                    t1Stack = character(),
                    stringsAsFactors = FALSE)


        #1
        #Goes through each line of dataset, comparing and getting similarity index from
Dandelion API
        for (i in 1:nrow(dataset))   {
          for (j in 1:nrow(dataset))   {
         #   for (i in 1:5)   {
           #   for (j in 1:5)   {
            #Creating url submission string to Dandelion API of each line of dataset
            #Submit call to Dandelion API and get JSON answer

            #Contex: all variables
              if (dataset$ID[[i]] != dataset$ID[[j]] ) {
                url_set = gsub(" ", "%20", (paste(url_part1,
                                    dataset$Project[[i]],                    dataset$Title[[i]],
dataset$Description[[i]],       dataset$ProcessActivity[[i]],       dataset$Project.Tags[[i]],
dataset$Task.Tags[[i]],
                                    url_part2,
                                    dataset$Project[[j]],                    dataset$Title[[j]],
dataset$Description[[j]],       dataset$ProcessActivity[[j]],       dataset$Project.Tags[[j]],
dataset$Task.Tags[[j]],
                                    url_part3)),
                        fixed=TRUE)
          #Extracts similarity index from Json response from Dandelion Server
          data1 = fromJSON(url_set)
          names(data1)
          similarity =  data1$similarity

          # Populates Data Frame
          df_allContext = rbind(df_allContext, data.frame(
                            Task1 = dataset$ID[[i]],
                            Task2 = dataset$ID[[j]],
                            method = "allContext",
                            similarityindex = similarity,
                            dataset$Stack.Overflow[[i]],
                            dataset$Stack.Overflow[[j]]
                            )
          )
            }
          }
        }


        #2
        #Goes through each line of dataset, comparing and getting similarity index from
Dandelion API
        for (i in 1:nrow(dataset))   {
          for (j in 1:nrow(dataset))   {
```

```
      #   for (i in 1:5)   {
      #   for (j in 1:5)   {
      #Creating url submission string to Dandelion API of each line of dataset
      #Submit call to Dandelion API and get JSON answer

      #Contex: no Description
      if (dataset$ID[[i]] != dataset$ID[[j]] ) {
        url_set = gsub(" ", "%20", (paste(url_part1,
                                   dataset$Project[[i]],                      dataset$Title[[i]],
dataset$ProcessActivity[[i]], dataset$Project.Tags[[i]], dataset$Task.Tags[[i]],
                                   url_part2,
                                   dataset$Project[[j]],                      dataset$Title[[j]],
dataset$ProcessActivity[[j]], dataset$Project.Tags[[j]], dataset$Task.Tags[[j]],
                                   url_part3)),
                     fixed=TRUE)
        #Extracts similarity index from Json response from Dandelion Server
        data1 = fromJSON(url_set)
        names(data1)
        similarity =  data1$similarity

        # Populates Data Frame
        df_allContext = rbind(df_allContext, data.frame(
          Task1 = dataset$ID[[i]],
          Task2 = dataset$ID[[j]],
          method = "noDesc",
          similarityindex = similarity,
          dataset$Stack.Overflow[[i]],
          dataset$Stack.Overflow[[j]]
        )
        )
      }
    }
  }


  #3
  #Goes through each line of dataset, comparing and getting similarity index from
Dandelion API
  for (i in 1:nrow(dataset))   {
    for (j in 1:nrow(dataset))   {
      #   for (i in 1:5)   {
      #   for (j in 1:5)   {
      #Creating url submission string to Dandelion API of each line of dataset
      #Submit call to Dandelion API and get JSON answer

      #Contex: no description and no title
      if (dataset$ID[[i]] != dataset$ID[[j]] ) {
        url_set = gsub(" ", "%20", (paste(url_part1,
                                   dataset$Project[[i]],          dataset$ProcessActivity[[i]],
dataset$Project.Tags[[i]], dataset$Task.Tags[[i]],
                                   url_part2,
                                   dataset$Project[[j]],          dataset$ProcessActivity[[j]],
dataset$Project.Tags[[j]], dataset$Task.Tags[[j]],
                                   url_part3)),
                     fixed=TRUE)
```

```
            #Extracts similarity index from Json response from Dandelion Server
            data1 = fromJSON(url_set)
            names(data1)
            similarity =  data1$similarity

            # Populates Data Frame
            # Populates Data Frame
            df_allContext = rbind(df_allContext, data.frame(
              Task1 = dataset$ID[[i]],
              Task2 = dataset$ID[[j]],
              method = "noDescnoTitle",
              similarityindex = similarity,
              dataset$Stack.Overflow[[i]],
              dataset$Stack.Overflow[[j]]
            )
            )
          }
        }
      }


      #4
      #Goes through each line of dataset, comparing and getting similarity index from Dandelion API
      for (i in 1:nrow(dataset))   {
        for (j in 1:nrow(dataset))   {
          #   for (i in 1:5)   {
          #   for (j in 1:5)   {
          #Creating url submission string to Dandelion API of each line of dataset
          #Submit call to Dandelion API and get JSON answer

          #Contex: noProcess
          if (dataset$ID[[i]] != dataset$ID[[j]] ) {
            url_set = gsub(" ", "%20", (paste(url_part1,
                                      dataset$Project[[i]],                          dataset$Title[[i]],
dataset$Description[[i]], dataset$Project.Tags[[i]], dataset$Task.Tags[[i]],
                                      url_part2,
                                      dataset$Project[[j]],                          dataset$Title[[j]],
dataset$Description[[j]], dataset$Project.Tags[[j]], dataset$Task.Tags[[j]],
                                      url_part3)),
                        fixed=TRUE)
            #Extracts similarity index from Json response from Dandelion Server
            data1 = fromJSON(url_set)
            names(data1)
            similarity =  data1$similarity

            # Populates Data Frame
            df_allContext = rbind(df_allContext, data.frame(
              Task1 = dataset$ID[[i]],
              Task2 = dataset$ID[[j]],
              method = "noProcActivity",
              similarityindex = similarity,
              dataset$Stack.Overflow[[i]],
              dataset$Stack.Overflow[[j]]
            )
            )
          }
        }
      }


      #5
      #Goes through each line of dataset, comparing and getting similarity index from Dandelion API
      for (i in 1:nrow(dataset))   {
        for (j in 1:nrow(dataset))   {
          #   for (i in 1:5)   {
          #   for (j in 1:5)   {
          #Creating url submission string to Dandelion API of each line of dataset
          #Submit call to Dandelion API and get JSON answer

          #Contex: No Desc, No Title, no Proc Activity
          if (dataset$ID[[i]] != dataset$ID[[j]] ) {
            url_set = gsub(" ", "%20", (paste(url_part1,
                                      dataset$Project[[i]],                dataset$Project.Tags[[i]], dataset$Task.Tags[[i]],
                                      url_part2,
                                      dataset$Project[[j]],                dataset$Project.Tags[[j]], dataset$Task.Tags[[j]],
                                      url_part3)),
                        fixed=TRUE)
            #Extracts similarity index from Json response from Dandelion Server
            data1 = fromJSON(url_set)
            names(data1)
            similarity =  data1$similarity

            # Populates Data Frame
            df_allContext = rbind(df_allContext, data.frame(
              Task1 = dataset$ID[[i]],
              Task2 = dataset$ID[[j]],
              method = "noProcActNoDescNoTitle",
              similarityindex = similarity,
              dataset$Stack.Overflow[[i]],
              dataset$Stack.Overflow[[j]]
            )
            )
          }
        }
      }

      #6
      #Goes through each line of dataset, comparing and getting similarity index from Dandelion API
      for (i in 1:nrow(dataset))   {
        for (j in 1:nrow(dataset))   {
          #   for (i in 1:5)   {
          #   for (j in 1:5)   {
          #Creating url submission string to Dandelion API of each line of dataset
          #Submit call to Dandelion API and get JSON answer
```

```
          #Contex: No Project Tag
          if (dataset$ID[[i]] != dataset$ID[[j]] ) {
            url_set = gsub(" ", "%20", (paste(url_part1,
                                    dataset$Project[[i]],                    dataset$Title[[i]],
dataset$Description[[i]], dataset$ProcessActivity[[i]], dataset$Task.Tags[[i]],
                                    url_part2,
                                    dataset$Project[[j]],                    dataset$Title[[j]],
dataset$Description[[j]], dataset$ProcessActivity[[j]], dataset$Task.Tags[[j]],
                                    url_part3)),
                        fixed=TRUE)
            #Extracts similarity index from Json response from Dandelion Server
            data1 = fromJSON(url_set)
            names(data1)
            similarity =  data1$similarity

            # Populates Data Frame
            df_allContext = rbind(df_allContext, data.frame(
              Task1 = dataset$ID[[i]],
              Task2 = dataset$ID[[j]],
              method = "noProjTag",
              similarityindex = similarity,
              dataset$Stack.Overflow[[i]],
              dataset$Stack.Overflow[[j]]
            )
            )
          }
        }
      }

      #7
      #Goes through each line of dataset, comparing and getting similarity index from
Dandelion API
      for (i in 1:nrow(dataset))   {
        for (j in 1:nrow(dataset))   {
          #   for (i in 1:5)   {
          #   for (j in 1:5)   {
          #Creating url submission string to Dandelion API of each line of dataset
          #Submit call to Dandelion API and get JSON answer

          #Contex: No Task Tag
          if (dataset$ID[[i]] != dataset$ID[[j]] ) {
            url_set = gsub(" ", "%20", (paste(url_part1,
                                    dataset$Project[[i]],                    dataset$Title[[i]],
dataset$Description[[i]], dataset$ProcessActivity[[i]], dataset$Project.Tags[[i]],
                                    url_part2,
                                    dataset$Project[[j]],                    dataset$Title[[j]],
dataset$Description[[j]], dataset$ProcessActivity[[j]], dataset$Project.Tags[[j]],
                                    url_part3)),
                        fixed=TRUE)
            #Extracts similarity index from Json response from Dandelion Server
            data1 = fromJSON(url_set)
            names(data1)
            similarity =  data1$similarity

            # Populates Data Frame
            df_allContext = rbind(df_allContext, data.frame(
              Task1 = dataset$ID[[i]],
              Task2 = dataset$ID[[j]],
              method = "noTaskTag",
              similarityindex = similarity,
              dataset$Stack.Overflow[[i]],
              dataset$Stack.Overflow[[j]]
            )
            )
          }
        }
      }

      #8
      #Goes through each line of dataset, comparing and getting similarity index from
Dandelion API
      for (i in 1:nrow(dataset))   {
        for (j in 1:nrow(dataset))   {
          #   for (i in 1:5)   {
          #   for (j in 1:5)   {
          #Creating url submission string to Dandelion API of each line of dataset
          #Submit call to Dandelion API and get JSON answer

          #Contex: no Project
          if (dataset$ID[[i]] != dataset$ID[[j]] ) {
            url_set = gsub(" ", "%20", (paste(url_part1,
                                    dataset$Title[[i]],                    dataset$Description[[i]],
dataset$ProcessActivity[[i]], dataset$Project.Tags[[i]], dataset$Task.Tags[[i]],
                                    url_part2,
                                    dataset$Title[[j]],                    dataset$Description[[j]],
dataset$ProcessActivity[[j]], dataset$Project.Tags[[j]], dataset$Task.Tags[[j]],
                                    url_part3)),
                        fixed=TRUE)
            #Extracts similarity index from Json response from Dandelion Server
            data1 = fromJSON(url_set)
            names(data1)
            similarity =  data1$similarity

            # Populates Data Frame
            df_allContext = rbind(df_allContext, data.frame(
              Task1 = dataset$ID[[i]],
              Task2 = dataset$ID[[j]],
              method = "noProject",
              similarityindex = similarity,
              dataset$Stack.Overflow[[i]],
              dataset$Stack.Overflow[[j]]
            )
            )
          }
        }
      }

      write.csv(df_allContext, file = "df_allContext.csv")
```

Appendix B – Preliminary Assessment Implementation Code

```
public class LeveTestClass {
  /**
   * Calculates the similarity (a number within 0 and 1) between two strings.
   */
  public static double similarity(String s1, String s2) {
    String longer = s1, shorter = s2;
    if (s1.length() < s2.length()) { // longer should always have greater length
      longer = s2; shorter = s1;
    }
    int longerLength = longer.length();
    if (longerLength == 0) { return 1.0; /* both strings are zero length */ }
    return (longerLength - editDistance(longer, shorter)) / (double) longerLength;

  }

  // Implementation of the Levenshtein Edit Distance
  public static int editDistance(String s1, String s2) {
    s1 = s1.toLowerCase();
    s2 = s2.toLowerCase();

    int[] costs = new int[s2.length() + 1];
    for (int i = 0; i <= s1.length(); i++) {
      int lastValue = i;
      for (int j = 0; j <= s2.length(); j++) {
        if (i == 0)
          costs[j] = j;
        else {
          if (j > 0) {
            int newValue = costs[j - 1];
            if (s1.charAt(i - 1) != s2.charAt(j - 1))
              newValue = Math.min(Math.min(newValue, lastValue),
                  costs[j]) + 1;
            costs[j - 1] = lastValue;
            lastValue = newValue;
          }
        }
      }
      if (i > 0)
        costs[s2.length()] = lastValue;
    }
    return costs[s2.length()];
  }

  public static void printSimilarity(String s, String t) {
```

```
            if (similarity(s, t) > 0.0) {

             System.out.println(String.format(

      "%.3f is the similar \"%s\" and \"%s\"", similarity(s, t), s, t));
            }
           }
  public static void main(String[] args) {
             printSimilarity("[2.10.13.1] [Recepção] [Cheio] Sistema não registra ocorrência
de envio de email ao criar pedido Construir e Testar Código","[HOMOL][Corretiva]
Filtros fixos estão editáveis para os perfis TRANSPORTADOR, DESPACHANTE,
ARMADOR e CLIENTE Construir e testar código");
            //...
            // For space sake, the lines were removed.
              }
}
```

Appendix C – Dataset Sample

ID	Stack Overflow	Project	Title	Description	ProcessActivity	Project Tags	Task Tags
19506	25636091	PTVV	[2.10.13.1] [Recepção] [Cheio] Sistema não registra ocorrência de envio de email ao criar pedido	Ao criar um pedido recepção cheio o sistema envia email com PDF mas não registra ocorrência do envio. Basta criar um pedido recepção cheio sem IMO e sem VGM, colocar horário e finalizar.	Error	sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport	detachedEntity
18984	25996758	PTVV	[2.10.2] Erro ao subir aplicação em homol após inclusão do campo listaEmailNFSe		Error	sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport	LazyInitializationException
18619	29174164	PTVV	[2.10.2] Evolutiva 39 - Inclusão de combo no campo E-mail envio NFS-e	Possibilitar informar mais de um e-mail no campo E-mail envio NFS-e ao criar pedido de retirada de DTC. Sugestão de inclusão de combo selecionável com os e-mails cadastradados nos registros da empresa.	Implement Code	sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport	combo; multi-selection
16565	29174164	PTVV	[2.10.6] [CRUD EMPRESA] Combo Tipo E-mail	A combo de tipo de e-mail não está funcionando no Internet Explorer (vide anexo)	Implement Code	sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport	combo; multi-selection
14446	29174164	PTVV	[2.9] [Presença de Carga] Lacre	Na tela de inclusão de presença de carga, a combo de lacres já deve trazer o lacre preenchido se só houver 1 lacre disponível para seleção. Existindo mais de 1 lacre, manter o comportamento atual (combo para seleção).	Implement Code	sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport	combo; multi-selection
10831	29174164	PTVV	[Termo de Entrega] A combo de despachante deve exibir os usuários que tiverem perfil tanto de despachante quanto de cliente, mostrando primeiro os que tenham perfil de despachante.		Implement Code	sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport	combo;
11607	25636091	PTVV	[Enviar Pendencia] Exception ao enviar pendencia	criou pedido com horario mandou special stow mandou appointment mandou bloqueio - ("Sucesso no Envio de bloqueio Manual RFB para o Navis") - cadeado mandou desbloqueio - ("Sucesso no Envio de desbloqueio Manual RFB para o Navis") enviou pendencia - (sistema exibe mensagem de erro abaixo)	Error	sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport	detachedEntity
4964	25636091	PTVV	[Reprogramar] - Erro ao reprogramar mais de uma vez um pedido	apos reprogramacao, tentar reprogramar novamente. provocando a excecao "detached persist"	Error	sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport	detachedEntity
19369	25996758	PTVV	[2.10.13] [Retirada Importação] Sistema quebra após enviar email na tela de detalhes	Ocorre erro de LazyInitializationException e o sistema quebra. Basta realizar a operação em um pedido retirada importação. O erro ocorreu em um pedido DI. Os testes foram feitos com o perfil adm_sistemas.	Error	sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport	LazyInitializationException

16575 25996758 PTVV [2.10.6] [Retirada Importação] [Reprogramar] - Erro ao reprogramar pedido importação Ocorre LazyInitializationException ao reprogramar pedido retirada importação Error sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport LazyInitializationException
15046 21168521 PTVV [CONTROLE DE CARGA] Cabeçalho das colunas devem ficar bloqueadas caso o usuário role ambos os grids. Quando o usuario fizer scroll para consultar os documentos ou cargas, o cabeçalho deve acompanhar os dados. tal qual ocorre no excel. Implement Code sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport
19649 2607289 PTVV [2.11.0] [Configuração Parâmetro Sistema] Corrigir erro ao não encontrar TipoParametroSistema para um valor no banco de dados Quando um novo parâmetro é criado no banco de dados mas não é inserido no ENUM, ao entrar na tela de Configuração Parâmetros Sistemas o sistema quebra. Alterar o sistema para exibir um alerta de que existem parâmetros "desconhecidos" no banco mas permitir o funcionamento normal da tela. Error sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport
20243 11345926 PTVV [2.10.15] - Corrigir erro de timeout ao tentar buscar arquivos edi no ftp Ler documentação do camel ftp para tentar encontrar uma solução. *OBS (DSV): Adicionei um parâmetro na conexão do FTPConsomeEdiCoparnRouter passando a conexão para o modo passivo.* *https://stackoverflow.com/questions/11345926/apache-camel-failing-ftp-component* *http://slacksite.com/other/ftp.html* Error sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport
21146 25472378 Sisccon Legado Alterar a validação do formato das OS e Booking no Ecargo. Houve uma mudança no formato das OS e Bookings no ECargo e é preciso alterar a validação que é feita SISCCON. Implement Code oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport
20810 28117615 Visão RFB Resolver problemas de integração de dados GTVV Error java; mongodb; oracle; angular; spring; hql; apache-camel; activemq; oracle; sybase
21220 13944222 Visão RFB Incidente 207052 – Data/Hora de escaneamento e Gate In... Incidente 207052 – Data/Hora de escaneamento e Gate In estão iguais e não deveriam pois o escaneamento não acontece no momento do Gate In. Os campos são AcessoVeiculo.Data de Entrada e Imagem.Data de Captura. Implement Code java; mongodb; oracle; angular; spring; hql; apache-camel; activemq; oracle; sybase
21221 15549931 Visão RFB Incidente 207059 – Quando o campo Ocorrencia.pesoDiferenca é inserido na consulta... Incidente 207059 – Quando o campo Ocorrencia.pesoDiferenca é inserido na consulta o campo Carga.Identificador de Carga passa a ser retornado em branco. Implement Code java; mongodb; oracle; angular; spring; hql; apache-camel; activemq; oracle; sybase
19453 38510879 Arquitetura Instalar e configurar Sonarqube nos projetos da Login Architecture Sonar; sonarqube; devops; jenkins; nexus;
19989 25098307 Migração JBoss Suporte configuração e deploy das integrações no JBoss Manage integrations jboss; apache-camel; log4j
20729 25636091 PTVV [2.10.16] [RECEPÇÃO CHEIO] Erros ao complementar Nota Fiscal) Sistema exibe mensagem de "Pedido Reprogramado com Sucesso" ao inserir notas fiscais e pedido ficar PROGRAMADO 2) Erro de LazyInitializationException ou DetachedEntity ao alterar notas fiscais (associar, incluir, desassociair nota) 3) Erro ao acessar a tela de notas fiscais pela tela de detalhes de um pedido, incluir notas para que o pedido fique PROGRAMADO, retornar para e tela de detalhes e reprogramar o pedido. Error sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport detached entity

19965 25636091 PTVV [2.10.17] Apontamento 23 - Não houve envio de bloqueio manual ao Navis (Retirada de DI) Durante o processo de tratativas do pedido 1720000287 do contêiner ZCSU8598748, o Perfil Fiel depositário utilizou a função de bloqueio manual para este pedido, porém não foi incluído o bloqueio do Navis BLOQUEIO MANUAL DE RETIRADA DE CONTÊINER. Error sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport detached entity
17235 25636091 PTVV [2.10.8] [Recepção Cheio] Erro ao adicionar contêiner além da quantidade disponível no booking O sistema apresentará um erro de nullpointer ao tentar adicionar o contêiner acima da capacidade do booking (no passo 4) e ao finalizar o pedido, apresentará o erro de "detached entity" Error sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport detached entity
10796 25636091 PTVV [Suspensão da Liberação] Ao suspender liberação em lote, o sistema exibe mensagem abaixo, 14:42:19,296 ERROR [retirada] org.hibernate.PersistentObjectException: detached entity passed to persist: br.com.loginlogistica.tvv.entity.OcorrenciaPedido Error sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport detached entity
18769 25996758 PTVV [TRUNK] [Retirada DTC] LazyException ao carregar emails ao Liberar pagamento Erro ao carregar destinatarios para envio de confirmacao de liberacao de pagamento Error sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport LazyInitializationException
11512 25996758 PTVV [Reprogramar Pedido DTC Manual] - Erro ao reprogramar pedido. 14:58:46,363 ERROR [org.hibernate.LazyInitializationException] could not initialize proxy - no Session Error sybase; oracle; jenkins; hibernate; java; seam; jsf; jsf-1.2; richfaces; jpa; poi; jasperreport LazyInitializationException

Appendix D – SMS exclusion form

Year	Authors	Title	Exclusion Reason
2018	Sirres R., Bissyand T.F., Kim D., Lo D., Klein J., Kim K., Traon Y.L.	Augmenting and structuring user queries to support efficient free-form code search	Subject differs from dissertation goal. Deals with vocabulary for search string formation
2018	Wei Q., Liu J., Chen J.	A method for recommending bug fixer using community Q&A information	Subject differs from dissertation goal. Measures expertise of developers.
2018	Etemadi V., Bushehrian O., Akbari R.	Association rule mining for finding usability problem patterns: A case study on StackOverflow	Subject differs from dissertation goal. Discover problem patterns in tools through usability issues.
2018	Gao S., Xing Z., Ma Y., Ye D., Lin S.-W.	Enhancing Knowledge Sharing in Stack Overflow via Automatic External Web Resources Linking	Subject differs from dissertation goal. Uses links from Stack Overflow Posts to reference official documentation of products.
2017	Liu X., Shen B., Zhong H., Zhu J.	EXPSOL: Recommending online threads for exception-related bug reports	Used a model trained by support vector machines.
2017	Fumin S., Xu W., Hailong S., Xudong L.	Recommendflow: Use topic model to automatically recommend stack Overflow Q&A in IDE	It is unclear if the solution suggests a query based on code context or threads.
2016	Sahu T.P., Nagwani N.K., Verma S.	An empirical analysis on reducing open source software development tasks using stack overflow	Although the proposal links bugs and posts, the goal is to compare de average bug fix time of posted bugs in SO.
2015	Nagy C., Cleve A.	Mining Stack Overflow for discovering error patterns in SQL queries	Subject differs from dissertation goal. Use Stack Overflow database to identify error-prone patterns in SQL queries
2015	Zheng X.-L., Chen C.-C., Hung J.-L., He W., Hong F.-X., Lin Z.	A Hybrid Trust-Based Recommender System for Online Communities of Practice	Subject differs from dissertation goal. E-Leaning paper that uses Stack Overflow to experiment.
2015	Amintabar V., Heydarnoori A., Ghafari M.	ExceptionTracer: A Solution Recommender for Exceptions in an Integrated Development Environment	No evaluation performed or metric presented.
2015	Wang W., Malik H., Godfrey M.W.	Recommending posts concerning API issues in developer Q&A sites	Subject differs from dissertation goal. Analyze Stack Overflow posts rather than based in other context

			content.
2016	Ponzanelli L., Bavota G., Di Penta M., Oliveto R., Lanza M.	Prompter: Turning the IDE into a self-confident programming assistant	Uses Stack Overflow API. The mechanism and search algorithm from this API is unknown.
2014	Ponzanelli L., Bavota G., Di Penta M., Oliveto R., Lanza M.	Mining stackoverflow to turn the IDE into a self-confident programming Prompter	Uses Stack Overflow API. The mechanism and search algorithm from this API is unknown.
2014	Ponzanelli L., Bavota G., Di Penta M., Oliveto R., Lanza M.	Prompter: A self-confident recommender system	Uses Stack Overflow API. The mechanism and search algorithm from this API is unknown.
2013	Rahman M.M., Yeasmin S., Roy C.K.	An IDE-based context-Aware meta search engine (SurfClipse)	Uses Stack Overflow API. The mechanism and search algorithm from this API is unknown.
2013	Ponzanelli L., Bacchelli A., Lanza M.	Seahawk: Stack overflow in the IDE	Same study conducted in paper S8
2012	Bacchelli A., Ponzanelli L., Lanza M.	Harnessing Stack Overflow for the IDE (Seahawk)	Same study conducted in paper S8
2012	Zagalsky A., Barzilay O., Yehudai A.	Example overflow: Using social media for code recommendation	No evaluation performed or metric presented.

Appendix E – *RapidMiner* Process XML

```
<?xml version="1.0" encoding="UTF-8"?><process version="8.2.001">
  <context>
    <input/>
    <output/>
    <macros/>
  </context>
  <operator activated="true" class="process" compatibility="8.2.001"
expanded="true" name="Process">
    <parameter key="logverbosity" value="init"/>
    <parameter key="random_seed" value="2001"/>
    <parameter key="send_mail" value="never"/>
    <parameter key="notification_email" value=""/>
    <parameter key="process_duration_for_mail" value="30"/>
    <parameter key="encoding" value="SYSTEM"/>
    <process expanded="true">
      <operator activated="true" class="retrieve" compatibility="8.2.001"
expanded="true" height="68" name="Retrieve dataset8Translations" width="90" x="112"
y="187">
        <parameter key="repository_entry" value="dataset8Translations"/>
      </operator>
      <operator activated="true" class="select_attributes" compatibility="8.2.001"
expanded="true" height="82" name="Select Attributes" width="90" x="246" y="187">
        <parameter key="attribute_filter_type" value="subset"/>
        <parameter key="attribute" value=""/>
        <parameter key="attributes"
value="Description|ProcessActivity|Project|Project Tags|Stack Overflow|Task
Tags|Title|Category"/>
        <parameter key="use_except_expression" value="false"/>
        <parameter key="value_type" value="attribute_value"/>
        <parameter key="use_value_type_exception" value="false"/>
        <parameter key="except_value_type" value="time"/>
        <parameter key="block_type" value="attribute_block"/>
        <parameter key="use_block_type_exception" value="false"/>
        <parameter key="except_block_type" value="value_matrix_row_start"/>
        <parameter key="invert_selection" value="false"/>
        <parameter key="include_special_attributes" value="false"/>
      </operator>
      <operator activated="true" class="text:process_document_from_data"
compatibility="8.1.000" expanded="true" height="82" name="Process Documents from
Data" width="90" x="380" y="187">
        <parameter key="create_word_vector" value="true"/>
        <parameter key="vector_creation" value="TF-IDF"/>
        <parameter key="add_meta_information" value="true"/>
        <parameter key="keep_text" value="true"/>
        <parameter key="prune_method" value="none"/>
        <parameter key="prune_below_percent" value="3.0"/>
        <parameter key="prune_above_percent" value="30.0"/>
        <parameter key="prune_below_rank" value="0.05"/>
        <parameter key="prune_above_rank" value="0.95"/>
```

```
          <parameter key="datamanagement" value="double_sparse_array"/>
          <parameter key="data_management" value="auto"/>
          <parameter key="select_attributes_and_weights" value="false"/>
          <list key="specify_weights"/>
          <process expanded="true">
            <operator activated="true" class="text:transform_cases"
compatibility="8.1.000" expanded="true" height="68" name="Transform Cases"
width="90" x="112" y="34">
              <parameter key="transform_to" value="lower case"/>
            </operator>
            <operator activated="true" class="text:tokenize" compatibility="8.1.000"
expanded="true" height="68" name="Tokenize" width="90" x="246" y="34">
              <parameter key="mode" value="non letters"/>
              <parameter key="characters" value=".:"/>
              <parameter key="language" value="English"/>
              <parameter key="max_token_length" value="3"/>
            </operator>
            <operator activated="true" class="text:filter_stopwords_dictionary"
compatibility="8.1.000" expanded="true" height="82" name="Filter Stopwords
(Dictionary)" width="90" x="380" y="34">
              <parameter key="file"
value="/Users/glaucia/Dropbox/mestrado/dissertation/stopwords.txt"/>
              <parameter key="case_sensitive" value="false"/>
              <parameter key="encoding" value="x-MacRomania"/>
            </operator>
            <operator activated="true" class="text:stem_snowball"
compatibility="8.1.000" expanded="true" height="68" name="Stem (Snowball)"
width="90" x="514" y="34">
              <parameter key="language" value="Portuguese"/>
            </operator>
            <connect from_port="document" to_op="Transform Cases"
to_port="document"/>
            <connect from_op="Transform Cases" from_port="document"
to_op="Tokenize" to_port="document"/>
            <connect from_op="Tokenize" from_port="document" to_op="Filter
Stopwords (Dictionary)" to_port="document"/>
            <connect from_op="Filter Stopwords (Dictionary)" from_port="document"
to_op="Stem (Snowball)" to_port="document"/>
            <connect from_op="Stem (Snowball)" from_port="document"
to_port="document 1"/>
            <portSpacing port="source_document" spacing="0"/>
            <portSpacing port="sink_document 1" spacing="0"/>
            <portSpacing port="sink_document 2" spacing="0"/>
          </process>
        </operator>
        <operator activated="true" class="set_role" compatibility="8.2.001"
expanded="true" height="82" name="Set Role" width="90" x="514" y="187">
          <parameter key="attribute_name" value="Stack Overflow"/>
          <parameter key="target_role" value="label"/>
          <list key="set_additional_roles">
            <parameter key="Stack Overflow" value="id"/>
          </list>
        </operator>
```

```
<operator activated="true" class="data_to_similarity_data" compatibility="8.2.001" expanded="true" height="68" name="Data to Similarity Data" width="90" x="447" y="34">
  <parameter key="measure_types" value="NominalMeasures"/>
  <parameter key="mixed_measure" value="MixedEuclideanDistance"/>
  <parameter key="nominal_measure" value="JaccardSimilarity"/>
  <parameter key="numerical_measure" value="CosineSimilarity"/>
  <parameter key="divergence" value="GeneralizedIDivergence"/>
  <parameter key="kernel_type" value="radial"/>
  <parameter key="kernel_gamma" value="1.0"/>
  <parameter key="kernel_sigma1" value="1.0"/>
  <parameter key="kernel_sigma2" value="0.0"/>
  <parameter key="kernel_sigma3" value="2.0"/>
  <parameter key="kernel_degree" value="3.0"/>
  <parameter key="kernel_shift" value="1.0"/>
  <parameter key="kernel_a" value="1.0"/>
  <parameter key="kernel_b" value="0.0"/>
</operator>
<operator activated="true" class="set_role" compatibility="8.2.001" expanded="true" height="82" name="Set Role (2)" width="90" x="581" y="34">
  <parameter key="attribute_name" value="SECOND_ID"/>
  <parameter key="target_role" value="prediction"/>
  <list key="set_additional_roles">
    <parameter key="FIRST_ID" value="label"/>
    <parameter key="SECOND_ID" value="prediction"/>
  </list>
</operator>
<operator activated="true" breakpoints="after" class="filter_examples" compatibility="8.2.001" expanded="true" height="103" name="Filter Examples" width="90" x="715" y="34">
  <parameter key="parameter_expression" value=""/>
  <parameter key="condition_class" value="custom_filters"/>
  <parameter key="invert_filter" value="false"/>
  <list key="filters_list">
    <parameter key="filters_entry_key" value="SIMILARITY.ge.0\.5"/>
  </list>
  <parameter key="filters_logic_and" value="true"/>
  <parameter key="filters_check_metadata" value="true"/>
</operator>
<operator activated="true" class="performance_classification" compatibility="8.2.001" expanded="true" height="82" name="Performance" width="90" x="849" y="34">
  <parameter key="main_criterion" value="first"/>
  <parameter key="accuracy" value="true"/>
  <parameter key="classification_error" value="false"/>
  <parameter key="kappa" value="false"/>
  <parameter key="weighted_mean_recall" value="false"/>
  <parameter key="weighted_mean_precision" value="true"/>
  <parameter key="spearman_rho" value="false"/>
  <parameter key="kendall_tau" value="false"/>
  <parameter key="absolute_error" value="false"/>
  <parameter key="relative_error" value="false"/>
  <parameter key="relative_error_lenient" value="false"/>
  <parameter key="relative_error_strict" value="false"/>
  <parameter key="normalized_absolute_error" value="false"/>
```

```
          <parameter key="root_mean_squared_error" value="false"/>
          <parameter key="root_relative_squared_error" value="false"/>
          <parameter key="squared_error" value="false"/>
          <parameter key="correlation" value="false"/>
          <parameter key="squared_correlation" value="false"/>
          <parameter key="cross-entropy" value="false"/>
          <parameter key="margin" value="false"/>
          <parameter key="soft_margin_loss" value="false"/>
          <parameter key="logistic_loss" value="false"/>
          <parameter key="skip_undefined_labels" value="true"/>
          <parameter key="use_example_weights" value="true"/>
          <list key="class_weights"/>
        </operator>
        <connect from_op="Retrieve dataset8Translations" from_port="output"
to_op="Select Attributes" to_port="example set input"/>
        <connect from_op="Select Attributes" from_port="example set output"
to_op="Process Documents from Data" to_port="example set"/>
        <connect from_op="Process Documents from Data" from_port="example set"
to_op="Set Role" to_port="example set input"/>
        <connect from_op="Set Role" from_port="example set output" to_op="Data
to Similarity Data" to_port="example set"/>
        <connect from_op="Data to Similarity Data" from_port="similarity example
set" to_op="Set Role (2)" to_port="example set input"/>
        <connect from_op="Set Role (2)" from_port="example set output"
to_op="Filter Examples" to_port="example set input"/>
        <connect from_op="Filter Examples" from_port="example set output"
to_op="Performance" to_port="labelled data"/>
        <connect from_op="Performance" from_port="performance" to_port="result
1"/>
        <portSpacing port="source_input 1" spacing="0"/>
        <portSpacing port="sink_result 1" spacing="0"/>
        <portSpacing port="sink_result 2" spacing="0"/>
      </process>
    </operator>
  </process>
```

Appendix F – Combinations on project task context elements

Selected Attributes		Precision	Accuracy	Attribute(s) change
Category Description Process Project	Project Tags Task Tags	40.87%	36.57%	Removed Title
Category Process Project	Project Tags Task Tags Title	45.64%	37.12%	Removed Description
Category Description Process Project	Project Tags Title	54.76%	38.28%	Removed TaskTags
Category Description Process Project	Task Tags Title	61.46%	74.47%	Removed ProjectTags
Description Process Project	Project Tags Task Tags Title	49.01%	41.18%	Removed Category
Category Description Process	Project Tags Task Tags Title	61.46%	74.47%	Removed Project
Category Description Process	Task Tags Title	77.78%	85%	Removed Project Project Tags
Title Description		71.67%	54%	Removed Category Process Project Project Tags Task Tags
Project Tags Task Tags		13.51%	22.69%	Removed Category Process Project Title Description