ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

Task Extraction in Software Document to Improve Search Process

V.Vidhya[1], T.Priyadharshini[2],

M. E Student, Department of Computer Science Engineering, Anna University(BIT Campus), Trichirapalli, Tamil Nadu, India

Assistant Professor, Department of Computer Science Engineering, Apollo Priyadarshanam Institute of Technology, Chennai, Tamil Nadu, India

ABSTRACT: Now-a-Days knowledge management and Artificial Intelligence are the vital concerns in all kinds of software firms. Simultaneously a significant part of the specialized learning can be caught in documentation, there frequently exists a crevice between the data needs of programming designers what's more, the documentation structure. To offer program engineers or developers, some assistance with navigating documentation, we built up a procedure for naturally extricating undertakings from conceptualizing so as to programmer documentation errands as particular programming activities that have been portrayed in the documentation. More than 70 percent of the assignments we extricated from the documentation of two undertakings were judged significant by at minimum one of two designers. We exhibit TaskNavigator, a client interface for pursuit inquiries that proposes errands removed with our strategy in an auto-complete rundown alongside ideas, code components, and segment headers. We directed a field study in which six proficient designers utilized TaskNavigator for two weeks as a major aspect of their continuous work. We discovered query items recognized through removed errands to be more useful to designers than those found through ideas, code components, and area headers. The outcomes show that errand portrayals can be successfully removed from programming documentation, and that they cross over any barrier between documentation structure and the data needs of programming engineers.

KEYWORDS: Task Extraction, Document Analysis, Link Navigation, Page Estimation, Page Ranking.

I. INTRODUCTION

In software documentation, the knowledge needed by software developers is captured in many forms of documentation, typically written by different individuals. Software documentation is an important part of software engineering. It describes user how to operate and how to use the system. Documentation is the information and describes the product to its user. It consists of product technical manuals and online information. Software documentation is derived from idea programmers and engineers “document” in formal writing. It should provide for communication between the team members and also provides the enough information to management to allow them to perform all program management related activities.

In software documentation, gap between the information needs of software developers and the documentation structure. Developers struggle to find the right information in the right form at the right time .The above issues motivated to me to choose this project. The main goal of software documentation users learns the knowledge work and users learn the use menus of computer program.

II. TYPES OF SOFTWARE DOCUMENTATION

Software documentation also referred as source code documentation and it also explains how to use the software. Software documentation can be classified into:

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

User Documentation

User documentation aims to help to use software properly. It considers the end users and also known as software manuals. It usually uses a guides the user step by step and uses a technical approaches. User documentation are also available for an online version and pdf format. It also accompanied by additional documentation such as video tutorials, videos and knowledge based articles etc.

Requirements Documentation

Requirement documentations also referred as requirements and also explains the software does and able to do. It consists of several types of requirements which may or may not be included documentation, depending on the purpose and complexity of the system. Those that can affect human safety and created to be used over a long period of time. The need of requirement documentation related to complexity of the product, impact of the product and life expectancy of the software.

Architecture Documentation

Architecture documentation is also referred as software architecture description. It either analyses the software architecture or communicates results to the work product. It provides the technical issues including online marketing and services. Architectural documentation is usually arranged into architectural models and closely related to comparison document. It addresses the current situation and proposes alternative solutions with an aim to identify the best outcome solution. It also designs documentation which includes the conceptual, logical and physical design elements.

Technical Documentation

Technical documentation is very important part of software documentation and many programmers used several type of documentation. It describes the codes, but also addresses the algorithm, interfaces and other technical aspects of software development and application. Technical documentation is usually created by the programmers with the aid of auto generating tools.

Task Extraction

In software documentation, gap between the information needs of software developers and the documentation structure. Developers struggle to find the right information in the right form at the right time .The above issues motivated to me to choose this project. The Scope of this project is to improve the precision of the task extraction. It easier for developers to navigate the documentation by automatically associating the tasks with each paragraph. To find the task description can be effectively extracted from software documentation. In software documentation, the knowledge needed by software developers is captured in many forms of documentation, typically written by different individuals.
Extracting task from software documentation by conceptualizing tasks as specific programming actions that have been described in the documentation. Task navigator, a user interface search queries that extracted the tasks in our technique. Many software development organizations and open-source projects, challenge by creating web pages that collect the most important information. Our goal is to detect the task from software documentation. In this task-oriented view, product information can be divided into these task categories are evaluating, planning for, setting up or installing, customizing, administering, using, and maintaining the product. Documentation is now often built directly into the product as part of the user interface and in help pages. Printed technical manuals are increasingly available at company Web sites in the form of Adobe Acrobat Portable Document Format (PDF) files or in HTML.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

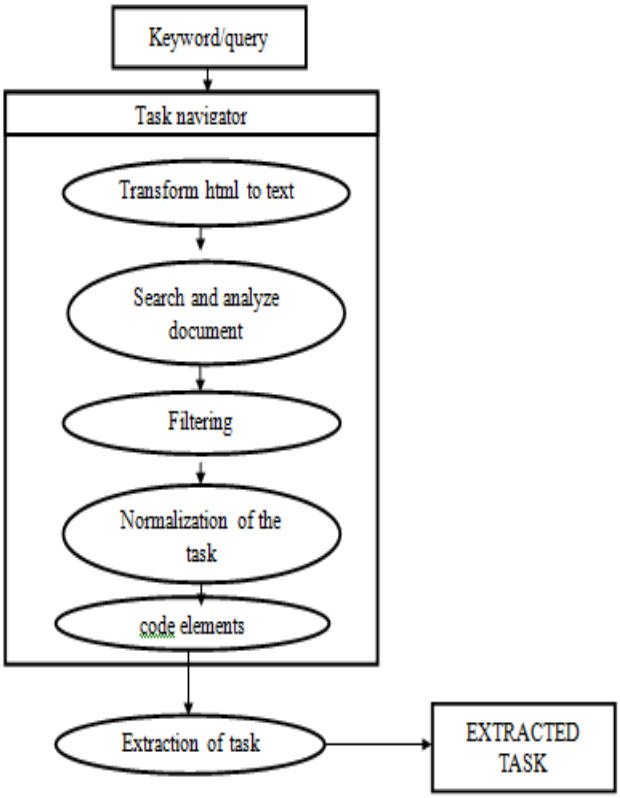

Fig.1. System Architecture Design

Pre-Processing:

In this step, preprocessed by transforming html files into text files and recognized that the textual documents had to be further analyzed and processed. Therefore, I have developed and adapted several techniques to preprocess source code from web. The preprocessing of source code in our code warehouse is partitioned in some phases as described below:

- ✓ ***Cleaning Normalization***
- ✓ ***Transformation***
- ✓ ***Feature extraction and selection, etc***

First, parse the textual data to identify tokens and obtain a stream of these tokens that are processed by the following filters. The first filter decomposes identifier tokens java.sql.ResultSet into their sub tokens java, sql, and ResultSet before returning them to the next filter. The next Filter splits camel cased tokens like Result Set into the sub tokens Result and Set as well as IOException into IO and Exception. The next Filter splits camel cased tokens like Result Set into the sub tokens Result and Set as well as IOException into IO and Exception.

Primitives of Task Extraction:

The technique used in this module for automatically extracting task descriptions from software documentation. Our tool, called TaskNav, suggests these task descriptions in an auto-complete search interface for software documentation along with concepts, code elements, and section headers. Itusestream Mining Techniques to detect every passage in a documentation corpus that describes how to accomplish some task. The core of the task extraction process is the use of grammatical dependencies identified by the Stanford parser to detect every instance of a programming action described in a documentation corpus. Different dependencies are used to account for different grammatical structures (e.g., “returning an iterator”, “return iterator”, “iterator returned”, and “iterator is returned”).

TaskNav can automatically analyze and index any documentation corpus based on a starting URL and some configuration parameters, such as which HTML tags should be ignored. Documentation users can benefit from TaskNav by taking advantage of the task-based navigation offered by the auto-complete search interface.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

- ✓ *Dependency extraction*
- ✓ *Task identification*
- ✓ *Context resolution*
- ✓ *Task refinement*
- ✓ *Task filtering*
- ✓ *Task normalization*

Filtering:
In this module Collocations are detected by finding sequences of words that co-occur more often than they would be expected to by chance. Extract concepts from a documentation corpus by the stream mining for identifying collocations. To filter out meaningless collocations such as noun, verb, adjective, etc. collocations are then filtered using pearson's chi square test.

Code Elements:
In this module the developer will extract Code elements from the documentation using recognized techniques, as well as the original titles found in the documentation**.** The code elements are detected through a set of regular expressions. Code elements: Subscription, ALLOW_URL_REBILL, and CRON_KEY.

Code Elements:
In this module the developer will extract Code elements from the documentation using recognized techniques, as well as the original titles found in the documentation**.** The code elements are detected through a set of regular expressions. Code elements: Subscription, ALLOW_URL_REBILL, and CRON_KEY.

III. EXPERIMENTAL RESULTS

In this system, we can extract the document from defined formatted document/power point files. The structure should be valid and once it is uploaded the verifier has to approve the uploaded extracted content. Once it is correct it gets approved by the verifier, otherwise it will be rejected.

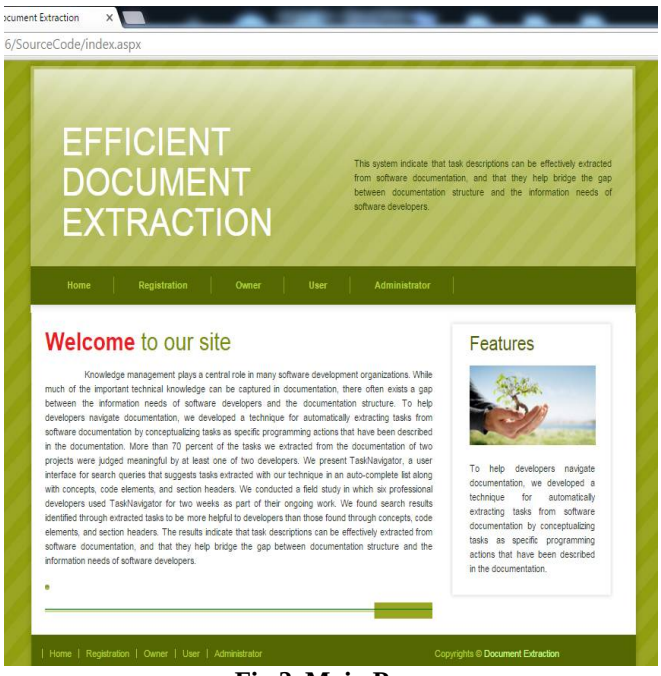

Fig.2. Main Page

In Main Page we have separate menus for administrator to audit the extracted task and approve the links and users to upload the new pages via power points/documents

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

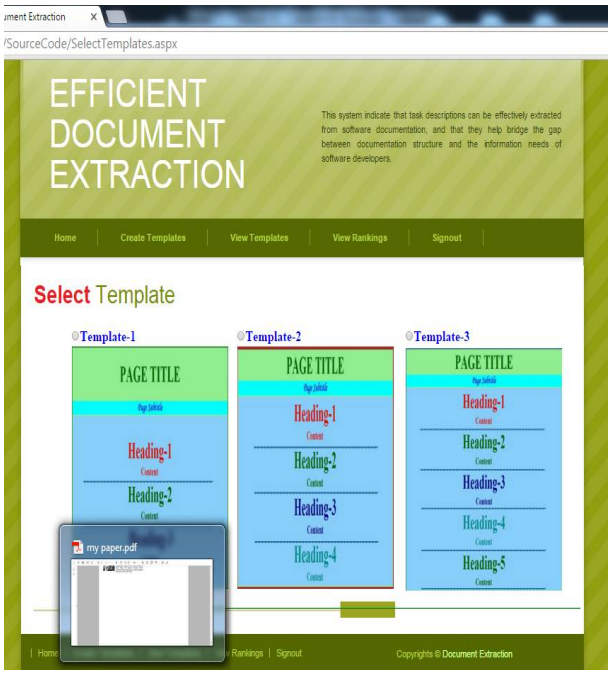

Fig.3. Template Selection

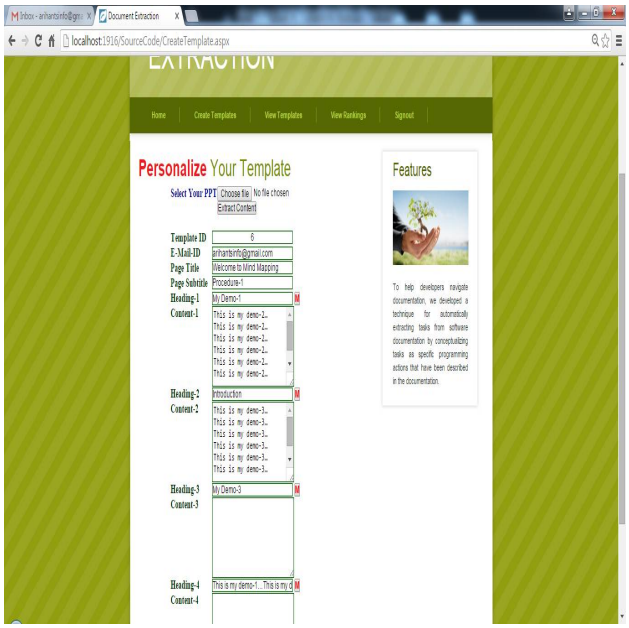

Fig.4. Document Extraction

IV. LITERATURE REVIEW

In "Extracting Structured Data from Natural Language Documents with Island Parsing", the authors A. Bacchelli, A. Cleve, M. Lanza, and A. Mocci, quoted on In this system propose an approach to extract the structured information to be found in documents written in NL software development to allow subsequent data parsing and modeling. The main

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

contributions of this is first A novel, robust and flexible technique that uses island parsing for mining and extracting structured information of Java systems from textual artifacts and second one is an evaluation of the proposed technique for the extraction of the structured information, based on a significant sample of real-world complex textual documents. Evaluated on this paper is a statistically significant set of emails and reached very high accuracy values.

Limitation:

- ✓ *Except java systems, it can be applicable with limited efforts*
- ✓ *Low performance*

In "Natural language parsing of program element names for concept extraction", the authors S. L. Abebe and P. Tonella, quoted on, this system propose natural language parsing (NLP) techniques to extract domain concepts and relations from program identifiers. To achieve this, sentences are constructed from list of terms that compose an identifier and parsed using NLP. The linguistic dependencies produced after parsing are then used to recognize concepts and relationships that form our automatically extracted ontology. It is achieved at negligible extra costs, since the ontology extraction technique is automated. The results obtained from the case study are encouraging and indicate that ontology concepts can contribute to a substantial reduction of the concept location effort, improved precision of the queries which include ontology concepts.

Limitation:

- ✓ Limited structural and programming language specification.
- ✓ It does not consider structural relationships such as extends and implements

V. CONCLUSION

Software documentation, to help bridge the gap between the information needs of software developers and the structure of existing documentation. In this paper the task based navigation technique is used. It improves precision of the task extraction. Task navigator, automatically analyze a documentation corpus and detect every passage that describes how to accomplish some task. The obtained search results for development tasks are more helpful to developers that generate the concepts, code elements, and task extraction. These results indicate that development task can be extracted from software documentation automatically, and they can help bridge the gap between software documentation and the information needs of software developers. In further process the task extraction can be done in software document without the use of web development projects.

REFERENCES

[1] S. L. Abebe et al(2010), "Natural language parsing of program element names for concept extraction," IEEE Int.Conf. Program Comprehension, pp. 156–159.
[2] A. Bacchelli, A. Cleve, M. Lanza, and A. Mocci (2011), "Extracting structured data from natural language documents with island parsing," Int. Conf. Automated Soft. Eng., pp. 476–479.
[3] M. Barouni-Ebrahimi and A. A. Ghorbani (2007), "On query completionin web search engines based on query stream mining," IEEE/WIC/ACM Int. Conf. Web Intell,pp.317–320.
[4]J.-R. Falleri, M. Huchard, M. Lafourcade, C. Nebut, V. Prince, andM. Dao(2010), "Automatic extraction of a WordNet-like identifier networkfrom software," IEEE Int. Conf. Program Comprehension, pp.4–13.
[5] G. Gay, S. Haiduc, A. Marcus, and T. Menzies (2009) "On the use of relevance feedback in IR-based concept location," IEEE Int. Conf. Softw. Maintenance, pp. 351 360.
[6] S. Gupta, S. Malik, L. Pollock, and K. Vijay-Shanker(2013), "Part-ofspeech tagging of program identifiers for improved text-based software engineering tools," IEEE Int. Conf. Program Comprehension, pp. 3–12.
[7] E. Hill, L. Pollock, and K. Vijay-Shanker (2009) "Automatically capturingsource code context of NL-queries for software maintenanceand reuse," Int. Conf. Soft. Eng., pp. 232–242.
[8] T. C. Lethbridge, J. Singer, and A. Forward (2003) "How software engineers use documentation: The state of the practice," IEEE Soft., vol. 20, no. 6, pp. 35–39.
[7] K. Pearson (1900), "On a criterion that a given system of deviations from the probable in the case of correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," vol. 50, no. 5, pp. 157–175.
[9] M. Petrenko, V. Rajlich, and R. Vanciu(2008), "Partial domain comprehension in software evolution and maintenance," IEEE Int. Conf. Program Comprehension, pp. 13–22.
[10] D. Poshyvanyk and A. Marcus(2007) "Combining formal concept analysiswith information retrieval for concept location in source code," IEEE Int. Conf. Program Comprehension, pp. 37–48.
[11] Y. Tian, D. Lo, and J. Lawall(2014), "Automated construction of a software-specific word similarity database," Softw. Maintenance, Reengineering Reverse Eng., pp. 44–53.
[12] C. Treude and M.-A. Storey (2012), "Work item tagging: Communicating concerns in collaborative software development," IEEE Trans. Soft. Eng., vol. 38, no. 1, pp. 19–3