Issue Tracking System based on Ontology and Semantic Similarity Computation

Habes Alkhraisat
Department of computer science
Al-Balqa Applied University
AL salt, Jordan

***Abstract*—A computer program is never truly finished; changes are a constant feature of computer program development, there are always something need to be added, redone, or fixed. Therefore, issue-tracking systems are widely used on the system development to keep track of reported issues. This paper proposes a new architecture for automated issue tracking system based on ontology and semantic similarity measure. The proposed architecture integrates several natural languages techniques including vector space model, domain ontology, term-weighting, cosine similarity measure, and synonyms for semantic expansion. The proposed system searches for similar issue templates, which are characteristic of certain fields, and identifies similar issues in an automated way, possible experts and responses are extracted finally. The experimental results demonstrated the accuracy of the new architecture, the experiment result indicates that the accuracy reaches to 94%.**

Keywords—issue tracking; ontology; similarity computation; vector space model

I. Introduction

Issue tracking systems are implemented as a part of integrated project management system. Software projects rely on issue tracking systems to direct corrective maintenance activity and to guide the maintenance activities of software developers. Users report symptoms of the issue along with related information, that include short or detailed textual descriptions of the issue, product and component that are affected by the issue, and how to reproduce. Developers then verify and fix the reported issues. There are often many reports that are received and thus developers would need to prioritize which reports are more important than others.

Issue tracking systems looks like a natural language information retrieval system that can be queried with natural language and return knowledge. Therefore the use of semantic knowledge in developing issue tracking systems improve its ability to semantically infer the similar issues. With cumulative information about issues collected by the issue tracking system over a period and with integrated semantic techniques, it is possible to build semantic issue tracking system that semantically searches for similar issues and links the knowledge for each issue.

This paper, proposes issue-tracking system with integrated semantic techniques for transforming issues content information into meaningful knowledge. The proposed system searches the knowledge documented to inferred semantically similar issues and recommended developers and the most similar files related to the reported issue.

The motivation of this work, for inference of the knowledge field and for recommendation of experts and files related to the reported issue, knowledge document includes not only the previously reported issues but also object-oriented mapping ontology, programmer-readable annotation, and developer experience. The main contribution of the paper is that, it proposes an issue tracking system that predicts similar issue in aa semantic way, possible experts, and possible program files related to issue.

II. Related Work

“Who Knows about That Bug? Automatic Bug Report Assignment with a Vocabulary-Based Developer Expertise Model” [1] uses source code vocabulary to find the most applicable expert for a given bug tracking item. This approach takes long time to make proper recommendations.

“Expertise Recommender: A Flexible Recommendation System and Architecture” [2] uses the change history of source code. It describes a general recommendation architecture that is grounded in a field study of expertise locating.

“Expert Recommender Systems in Practice: Evaluating Semi-automatic Profile Generation” [3] uses a client program which examines the documents within a folder and subfolder which was selected by the user and sends these examined word statistics to the server and compares it with other statistics.

[1], [2], and [3] have the problem that they are not sufficiently integrated into the task workflow a bug tracking and project management system.

III. System Architecture

The semantic approach for issue tracking system, proposed in this paper, consists of two main process: frontend process for handling the users’ issues and backend system for processing issues. Fig. 1 illustrates the main components for issues tracking system proposed in this paper. Frontend system handles the submitted issue and deal with the issue real-time processing. The backend is the platform for frontend processing, and mainly processes and maintains the issue database.

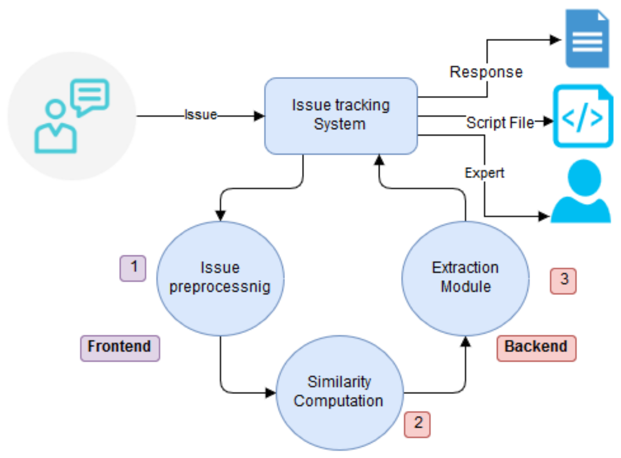

Fig. 1. Overview of issue tracking System Architecture

Fig. 2 shows the global architecture and necessary components to build the proposed system and clarifies the interaction between the system components. The system starts by receiving issue from the user, and finishes by providing the appropriate response for the reported issue, passing through the different phases.

The first component handles the job of prepossessing of issues, source code, and system documentations, which includes tokenization, stemming and keyword extraction. In the second component, the system applies the similarity computation, the similarity computation includes the semantic extension, feature vector generation and similarity measures. In the third component, the response to the submitted issues are extracted from the issues database, for response extraction the following are applied: confidence computation, response selection, and automatic return.

The scenario for an issue tracking system include the following:

1) The user reports an issue to the issue tracking system.

2) Next, in the issue Processing Module, the issue is rephrased by expanding the issue and passing it to the Issue Extraction Module

3) The Information Retrieval component is used to retrieve the relevant issues, response, files, and developers based on the important keywords that appear in the issue.

IV. Preprocessing Model

Preprocessing model starts by tokenizing issues dataset, internal and external system documentations. The tokenization splits up the entire issues, user manual, and programmer annotation into a bag of words. For improving the performance of extracting module and to have exactly matching stems, stemming algorithm has been applied to the bag of words generated after the tokenization process [4]. As a final step, the preprocessing model removes stop words from the bag of words generated after the stemming process. English stop word list which is available online[1] is used for the removal of stop words from the stem word.

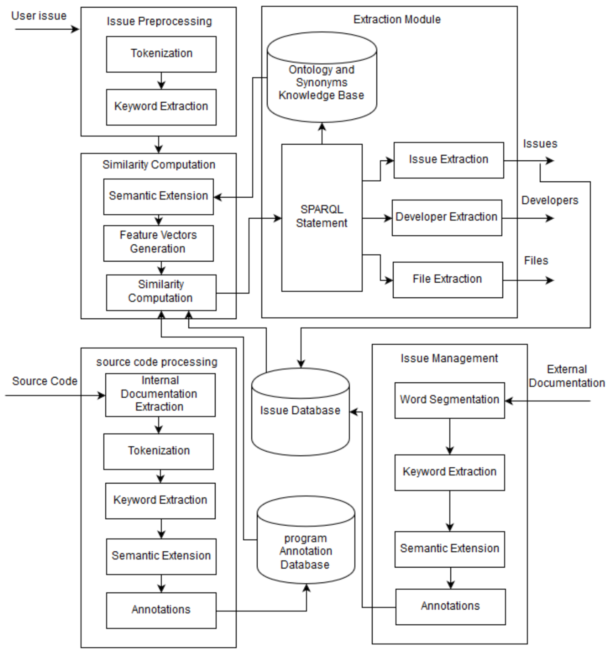

Fig. 2. Global schema of system Architecture

To illustrate the preprocessing model, let's study the following issue:

Issue: There is a problem while uploading the staff personal image.

Tokens: there, is, a, problem, while, uploading, the, staff, personal, images.

Keywords: problem, uploading, staff, personal, image

Stem: problem, upload, staff, person, image

Each keyword is enhanced with the synonyms terms extracted from the ontology. Therefore, our system adds takes the benefits of the shared ontologies and enriches the keyword senses with senses extracted from their synonyms. For semantic extension and keyword enrichment, the synonyms of keywords have been extracted from Macmillan Dictionary[2]. As example, it is possible to enrich the keywords of issue by extraction all synonyms of word "problem" and we get: "problem, difficulty, trouble", and by extraction all synonyms of word "image", and we get: "photo, picture, portrait". The outcome of preprocessing model is a bag of words. The bag of words is then used to represent the issue numerically as vector.

V. Ontology Mapping

Ontologies plays an important role in applications based on the semantic technologies. It consists of concepts, relationships between concepts, restrictions and is described in the ontological languages like Web Ontology Language (OWL).

[1] B. R. Porter M, "The English (Porter2) stemming algorithm," 09 2016. [Online]. Available: http://snowballstem.org.
[2] Princeton University, "WordNet," 09 2016. [Online]. Available: http://www.macmillandictionary.com/.

OWL represents the rich and complex knowledge about things, groups of things, and relations between things.

In some sense, object-oriented representation looks like the ontological representation. In this paper, we used the Semantic framework for mapping object-oriented model to semantic web languages [5]. The relations between object-oriented elements and ontologies are described in table 1.

TABLE I. OBJECT-ORIENTED PARADIGM AND ONTOLOGIES MAPPING

Ontology		**Object Oriented**
Class	↔	Class
Instance object	↔	Object
Property	↔	Attribute
Predicate	↔	Attribute name
Object	↔	Attribute value

The following example illustrates the mapping process between class written in PHP language and OWL.

PHP class		**OWL**

class employee
{
private id;
private name;
private personal_image;
}
	→	<owl:Class rdf:id="staff">
		<owl:DataTypePropety rdf:id="id">
		<rdfs:range rdf:resource="integer">
		<rdfs:domain rdf:resource="employee">
		</owl:DataTypePropety >
		<owl:DataTypePropety rdf:id="name">
		<rdfs:range rdf:resource="string">
		<rdfs:domain rdf:resource="employee">
		</owl:DataTypePropety >
		<owl:DataTypePropety rdf:id="personal_image">
		<rdfs:range rdf:resource="string">
		<rdfs:domain rdf:resource="employee">
		</owl:DataTypePropety >

VI. SIMILARITY COMPUTATION

Computing the similarity between user's issues with both the issues and program annotations databases plays an important role in the automated issue tracking system.

Issue similarity refers to the similarity between the keyword set of given issue annotated by ontology and the pattern keyword set of issues, and instance keywords have been replaced with class keywords in the keyword set of ontology.

There are many computational models for text similarity such as, support vector machines (SVMs), neural network (NN), machine learning, K-Nearest Neighbor (KNN), and so on. In this paper, vector space model (VSM) has been applied for implementing the propose issue tracking system. The VSM was developed for the SMART information retrieval system [6]. VSMs perform well on tasks that involve measuring the similarity of meaning between words, phrases, and documents [7].

The idea of the VSM is to represent large collection of documents as a vector in a vector space. Using VSM the set of issues and queries are represented as m-dimensional vectors of identifiers in a common vector space, and the vectors are organized into a matrix. The row vectors of the matrix correspond to words and the column vectors correspond to issues. Suppose the issue collection contains n issues and m unique terms. The vector space will then have m rows and n columns. The element $w_{i,j}$ in document vector space represents a non-binary weight of the i^{th} term k_i in the j^{th} issue I_j. Let the weight $w_{i,j}$ associated with a pair (k_i, I_j) is positive and non-binary, then the issue I_i and query Q are represented as vectors:

$$\vec{I_i} = (w_{i,1}, w_{i,2}, \cdots, w_{i,m})$$
$$\vec{Q} = (w_{q,1}, w_{q,2}, \cdots, w_{q,m})$$

where m is the number of feature terms.

The relevance of an issue to a query is given by the similarity of their vectors. The weight for terms in queries and issues are used in the computing degree of similarity. The most popular way to measure the similarity of two vectors is to compute their cosine. The cosine of the angle between issue vector $\vec{I_i}$ and query vector $\vec{Q}$ in vector space models can be measured as follows:

$$sim(\vec{I_i}, \vec{Q}) = \cos(\theta) = \frac{\vec{I_i} \cdot \vec{Q}}{\|\vec{I_i}\| \times \|\vec{Q}\|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sum_{i=1}^{t} w_{i,j}^2 \times \sum_{i=1}^{t} w_{i,q}^2} \quad (1)$$

$sim(\vec{I_i}, \vec{Q})$ varies from 0 to +1, the vector model ranks the documents according to their degree of similarity to the query, the $sim(I_i, Q)$ = 1 when $I_i = Q$, and $sim(I_i, Q) = 0$ when I_i shares no terms with Q.

VII. TERM-WEIGHTING SCHEME

The success of vector space model lies in term-weighting scheme. It assigns more weight to surprising events and less weight to expected events. The term weighting for the vector space model has entirely been based on single term statistics. There are three main factors: term frequency factor, collection frequency factor and length normalization factor. All three factor are multiplied together to make the resulting term weight.

In VSM a weight is assigned to each term in a document depends on the number of occurrences of the term in the issue. The frequency of a term k_i inside a document I_j referred to as the term frequency tf factor and is given by:

$$tf(k_i) = \frac{freq_{i,j}}{max_j\, freq_{i,j}} \quad (2)$$

Furthermore, the inverse of the frequency of a term k_i among the documents in the collection referred to as the inverse document frequency idf. idf measures of rareness of a term across all documents. Assume there are N documents in the collection, and that term k_i occurs in df_i of them. Then idf factor of term k_i is essentially

$$idf(k_i) = \log(\frac{N}{df_i}) \quad (3)$$

Thus, the idf of a rare term is high, whereas the idf of a frequent term is likely to be low. The tf and idf are combined, to produce a composite weight schema for each term in each document, the resulted weight schema is called a Term frequency–inverse document frequency $tf - idf$ scheme

[7][8]. The $tf - idf$ weighting scheme assigns to term k_i a weight in document d given by:

$$tf - idf(k_i) = tf(k_i) \times idf(k_i) \quad (4)$$

VIII. Response Extraction and Updating Issues Database

Once ontology classes and properties from keyword set are well fixed, the appropriate SPARQL query to retrieve the response from issue database is built[3]. The query is composed from resources and ontology classes determined by the keyword set. The class name in query pattern will be replaced with corresponding instance name in user issue, based on the query patterns and replace-pair in the most similar issue. The response to the issue is parsed to get extracted the response from the result of SPARQL statement.

Finally, the system updates the issues for constant learning and answering new issues. All issues with a cosine similarity measure higher than a defined cut-off threshold are considered similar and added to the feature set, weighted by its similarity value. On the other hand, if the a cosine similarity measure is less than the predefined threshold, then there is no corresponding query pattern in the issue database, and the issue will be added to issue database after annotating. In this paper, all issues with a cosine similarity value higher than 0.17, which has a 76% accuracy, are considered similar and added to the feature set, weighted by its similarity value.

IX. System Implementation

Issue tracking system architecture described previously has been implemented using PHP. It is semantic-based issue response that returns similar issues and recommends experts for the submitted issue.

X. Experiments and Evaluation

For the evaluation purpose, the proposed issue tracking system has been installed for employees working at IT department of an Institute of Family Health (IFH)[4]. At the time of system evaluation set of 240 issues have been evaluated collected from 100 employees working at IFH and 5 experts working at IT department have been given. The Ontology database includes 20 classes and 100 properties.

For evaluation purpose, accuracy and Recall Rate has been applied [9]. Accuracy and recall for the proposed issue tracking system are shown in Table 2.

[3] https://www.w3.org/TR/rdf-sparql-query/
[4] The Institute for Family Health (IFH) is a regional model providing comprehensive family healthcare services and training for professionals and caretakers in the fields of family healthcare. http://www.ifh-jo.org/index.php?language_id=1

XI. Conclucsion

The main contribution of this paper is that it proposes an ontological semantic based issue tracking system. To achieve the system goals, the proposed system combines the Vector Space Models with domain ontology representing the issue and code vocabulary. To improve the semantic similarity and accuracy of system, each word is enhanced using the synonyms terms extracted from the ontology pool and keywords synonyms database. Therefore, the system takes advantage of the shared ontologies available on the Web and semantically enriches the keyword senses with senses extracted from their synonyms.

The experimental results indicate that the system reaches an accuracy of 94% based on test set of 240 issues and 5 experts. In future extensions, the accuracy of the proposed system would be compared with Jira bug tracking [10].

TABLE II. Experiment Result

	Issue Response	**File recommender**	**Experts recommender**
Recall	95%	98%	92%
Accuracy	94%	96%	93.5%

References

[1] M. Dominique , K. Adrian and N. Oscar, "Assigning Bug Reports using a Vocabulary-Based Expertise Model of Developers," 6th IEEE International Working Conference on Mining Software Repositories, pp. 131-140, 2009.

[2] M. W. David and A. S. Mark, "Expertise Recommender: A Flexible Recommendation System and Architecture," in Proceedings of the 2000 ACM conference on Computer supported cooperative work, New York, 2000.

[3] T. Reichling and V. Wulf, "Expert recommender systems in practice: evaluating semiautomatic profile generation," in SIGCHI Conference on Human Factors in, New York, 2009.

[4] M. Porter, "An Algorithm for Suffix Stripping," Program electronic library and information systems, vol. 14, no. 3, pp. 130-137, July 1980.

[5] M. R. Ježek P., Semantic framework for mapping object-oriented model to semantic web languages., vol. 9, Front. Neuroinform, 2015.

[6] G. Salton, A. Wong and S. C. Yang, "A Vector Space Model for Automatic Indexing," Communications of the ACM, vol. 18, no. 11, pp. 613-620, Nov. 1975.

[7] D. M. Christopher, R. Prabhakar and S. Hinrich, Introduction to Information Retrieval, New York: Cambridge University Press, 2009.

[8] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Information Processing and Management, vol. 24, no. 5, pp. 513-523, 1988.

[9] S. Blair, "A Guide to Evaluating a Bug Tracking System.," 2004.

[10] V. Heyn and P. Adrian, "Semantic Jira - Semantic Expert Finder in the Bug Tracking Tool Jira," in 9th International Workshop on Semantic Web Enabled Software Engineering, Berlin, 2013.