Analyzing Discussion Forums Threads About Java Programming Language Usage

Valeria Zoratto$^+$, Gabriela N. Aranda$^+$, Sandra Roger*, Alejandra Cechich$^+$

Grupo GIISCo$^+$, Grupo GILIA*
Facultad de Informática, Universidad Nacional del Comahue
Buenos Aires 1400 (8300) Neuquén, Argentina
{vzoratto|gabriela.aranda|roger}@fi.uncoma.edu.ar

Abstract. The information generated by means of users interactions in specialized discussion forums can be very useful for other users with similar problems. Our proposal is a process to capture, maintain and analyze existing discussion threads from technical forums. This process can be used to recommend a series of possible solutions in fewer attempts than using traditional multi-purpose Web search engines. This paper presents two case studies that focused on real technical discussion forum threads about problems using the Java programming language. Also, some strategies to classify and relate discussion threads to their corresponding Java document classes are proposed.

1 Introduction

During the last decades, Web has been transformed by the appearance of many platforms for opinion and recommendation exchange (such as wikis and weblogs), and others that already existed (as discussion forums) have consolidated, conforming the Web 2.0 [1]. Particularly, discussion forums are public spaces where messages exchanged among users remain open so that others can contribute to the discussion. According to the classification proposed by Ellis et al. [2], discussion forums are software to support collaborative work that correspond to the messaging systems category (based on text messages) and asynchronous (meaning that participants do not need to be connected simultaneously to interact). Discussion forums available on the Web contain a broad knowledge about recurring problems related to software development and maintenance. According to Gottipati et al. [3], analyzing such information is something desirable and valuable. Software developers regularly employ multipurpose search engines to access such an information. Doing so, they usually go through several pages until they find a problem similar to the one they have. In this way, users might visit many pages before they find a possible solution. Moreover, sometimes they need to try several solutions until they find the most suitable. In order to help software developers to find a fitting solution in a shorter time (and an easier way), in this article we present the basis of a process that will be able to offer a ranking of solutions more likely to be successful. To do so, a preliminary classification of forum discussion threads according to a set of predefined categories is needed. Figure 1 shows such a classification process, where threads are related to a set of more recognizable entities, which are called *reference documents*. These reference documents represent the categories of the desired classification, and they can be selected according to the area of interest of the discussion forum threads under study.

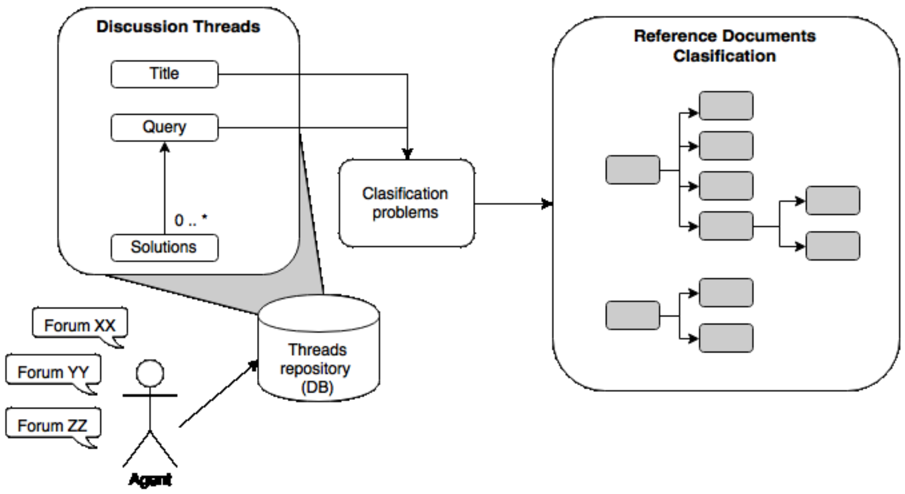

Fig. 1. Thread classification according to reference documents

Having such a goal in mind, in Section 2 we describe, the design of a family of case studies to evaluate strategies for threads classification . Later, in Section 2.1 we introduce, a process to evaluate the proposed assumptions. Following we introduce, the development of two case studies in Section 3 and Section 4, while Section 5 focuses on the comparative analysis of their results. Finally, related work, conclusions and future work are presented in Sections 6 and 7.

2 Design of a Family of Case Studies

Since the focus of our research is analyzing information which is available on the Web, we have designed an empirical strategy based on a series of case studies. Case studies have been chosen since they are flexible design studies, which allow several iterations over the process steps, while experiments and surveys, are fixed design studies that execute the steps once [4].

The proposed family of case studies has been designed following the process defined by Wholin et al. for experimentation in Software Engineering [5], and the GQM method [6] has been applied to define its goals, as it is shown in Table 1.

Table 1. GQM template for case study definition

Analyze	Discussion forums threads available on the Web referring technical problems (*discussion threads*)
For the purpose of	Classifying discussion threads to relate them to recognizable entities (*reference documents*)
With respect to	To determine the level of relationship between different discussion forum threads
From the point of view of	External user of discussion forums (*agent*)
In the context of	Public information available on the Web

In order to conduct a first series of case studies, we have focused our research on forum discussion threads that deal with Java programming language problems, therefore, the chosen set of classification categories or reference documents, are the Oracle Java class specification documents (version 5) [1]. Figure 2 shows an example of both kinds of document structures. For example, a forum discussion thread (a) has a title, a main question and a series of answers. On the other side, an Oracle Java class specification document(b) has a Class Name, a list of Implemented Interfaces, Direct Known Subclasses, etc.

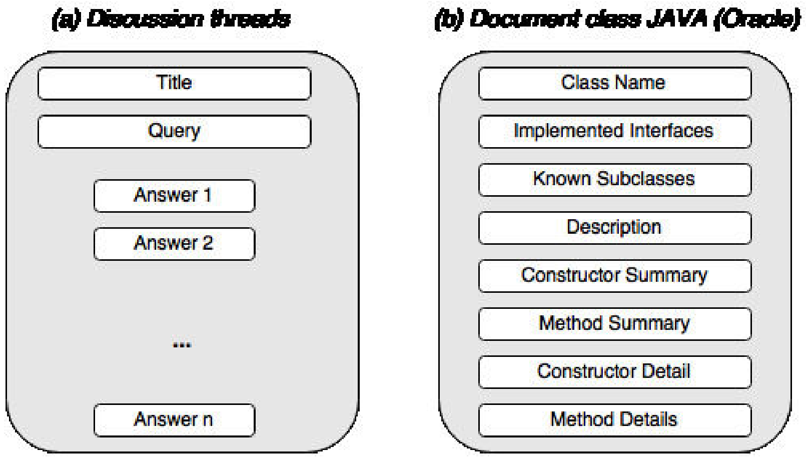

Fig. 2. Structure of documents under study

Considering the kind and amount of information in each type of document involved in this study, we have defined the following assumptions:

[1] http://docs.oracle.com/javase/1.5.0/docs/

- Assumption A: Classifying forum discussion threads into Java classes hierarchy is better (more precise) when more information from Oracle Java class documents is used.
- Assumption B: Classifying forum discussion threads into Java classes hierarchy is better (more precise) when more information about the problem described in each thread is used.

2.1 Evaluation Process

Based on the design explained above, we have defined a series of steps to test hypotheses proposed by case studies. Figure 3 shows the phases of this process, as well as their inputs and outputs. Though the phases have been numbered and presented as a sequence, some of them can be performed in concurrently. For example, Phase 2 (Expert classification) can be made at the same time than Phase 3, 4 or 5, since its development and results do not depend on the other phases or their intermediate results. The following sections will explain the most important features of each phase and their implementation in the case studies that are shown in this article.

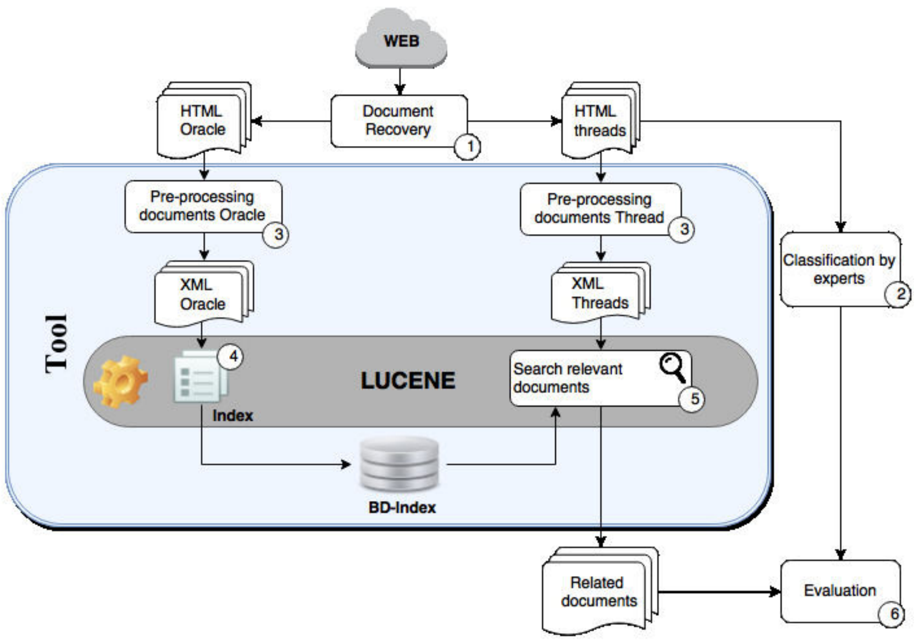
WEB
Document Recovery 1
HTML Oracle
HTML threads
Tool
Pre-processing documents Oracle 3
Pre-processing documents Thread 3
Classification by experts 2
XML Oracle
XML Threads
4
LUCENE
Index
Search relevant documents 5
BD-Index
Related documents
Evaluation 6

Fig. 3. Case study process

Phase 1: Documents Recovery. The main goal of this phase is to capture the documents that will be the basis of the case studies to be developed. Since discussion threads will be related to questions about Java programming language usage, it was determined that a suitable set of reference documents would be the Oracle specification for Java classes, Version 1.5.0. Table 2 presents the template model to document the case study characteristics.

For retrieving both kinds of documents (Oracle specifications and forum threads), we selected GNU *wget*[2] free software package . It is a non-interactive command-line tool, so it may easily be called from scripts.

Table 2. Template for technical characteristics of the case study

	Reference documents	**Discussion forum threads**
Site	Oracle	
Language	English	
URL Site	http://docs.oracle.com/javase/1.5.0/docs/	
Retrieval date	27/11/2014	
Selection criteria	-	
Retrieved documents	2953 (all the available for 1.5.0 version)	

Phase 2: Expert Classification. In this phase, the threads recovered in the previous phase are analyzed to identify those Java classes mentioned in the thread that are more related to the problem. In order to obtain an impartial qualification, this analysis must be agreed by multiple experts. Since threads are written in natural language, the information contains a high level of ambiguity. For this reason, a five-level Likert scale has been suggested: `<very high, high, medium, low, very low>`. The previous scale has been used to determine which Java classes are more related to the discussion thread and in which level they are considered to be related.

For example, given the discussion thread shown in Table 3, expert classification (Table 4) has determined that it is highly related to the Double Class (`very high`), and the Integer Class (`high`), but poorly related to the String Class (`low`), and the JOptionPane Class `(very low)`.

Phase 3: Document Pre-Processing. The goal of this phase is to prepare the documents downloaded from the Web (in HTML format) for further analysis with information retrieval techniques. To do so, new versions of these documents are generated in XML format, discarding irrelevant HTML code (for example, style tags, paragraph breaks, etc.), and adding labels to differentiate sections of interest in this study – as depicted in Figure 2. Also, to avoid unnecessary information, other sections with irrelevant content (e.g. advertising banners) are discarded.

[2] https://www.gnu.org/software/wget/

Table 3. Example of a discussion thread

URL	stackoverflow.com/questions/2240851/how-do-i-read-in-a-double-in-java
Title	How do I read in a double in Java?
Main Question	I am trying to get a decimal input from the keyboard, and it is just not working. First I tried double d = Integer.parseInt(JOptionPane.showInputDialog("Please enter a number between 0 and 1:")); and that obviously didn't work very well. I am used to just parsing int's as they come in from the keyboard right into a variable, but I don't know what I am supposed to do for decimals! I need to be able to take a decimal like .9 straight from the keyboard and be able to have it in a variable I can do calculations with. I know this is a basic question, but I need some help. Thanks!
Ans 1	double d = new Double(JOptionPane.showInputDialog("Please enter a number between 0 and 1:"));
Ans 2	I would use the Double class rather than the Integer class :) double d = Double.parseDouble(JOptionPane.showInputDialog("Please enter a number between 0 and 1:"));
Ans 3	Have you tried substituting Double for Integer ? So Double.parseDouble(JOptionPane.showInputDialog("Please enter a double:")); Or Double.valueOf(JOptionPane.showInputDialog("Please enter a double:"))
Ans 4	You should use Double.parseDouble(String). Another option is Double.valueOf(String) and I prefer the later because it's more change tolerant (if the parameter is changed for something else than a String, you may not have to modify your code), even if it creates an unnecessary Double that you don't need in your example.

Table 4. Example of expert classification for thread shown in Table 3

Very High	High	Medium	Low	Very Low
Double	Integer		String	JOptionPane

Finally, in order to provide information to evaluate the assumptions, the following three different versions of Oracle documents are generated:

- O_a: Only the class name.
- O_b: Class name and method names.
- O_c: All sections of the document (except the "Details constructor" and "Detailed methods" sections).

Similarly, forum threads are processed to obtain the following three different versions:

- F_1: Only the thread title.
- F_2: The title of the thread and the main question.
- F_3: The full text of the thread (title, main questions and answers).

Given the different versions of both kinds of documents, nine combinations are settled, which form the basis of the case study family, as it is shown in Table 5.

Table 5. Combinations of the types of documents to be analyzed

	Only Title (F_1)	Title + Query (F_2)	Full Text (F_3)
Only the name of the class (O_a)	O_aF_1	O_aF_2	O_aF_3
The name of the class and the names of all methods (O_b)	O_bF_1	O_bF_2	O_bF_3
Full Text (O_c)	O_cF_1	O_cF_2	O_cF_3

Phase 4: Reference Document Indexing. In this phase, reference documents (previously recovered and pre-processed) are indexed automatically using Lucene 4.9.0 [7].

To do so, the original stopwords list of Lucene has been modified to remove the Java reserved words (`for, then, if, this`) and add other words that were not representative in that context.

Phase 5: Searching relevant documents. By combining the 3 versions of Oracle documents (O_a, O_b, O_c) and threads documents (F_1, F_2, F_3), a series of 9 tests has been defined for this phase, which is shown in Table 6.

Table 6. Description of the case study tests

#	Test	Description
1	O_aF_1	Oracle documents containing only the class name and Forum thread documents containing only the title.
2	O_aF_2	Oracle documents containing only the class name and Forum thread documents containing the title and the main question.
3	O_aF_3	Oracle documents containing only the class name and full text Forum thread documents.
4	O_bF_1	Oracle documents containing the class and method names and Forum thread documents containing only the title.
5	O_bF_2	Oracle documents containing the class and method names and Forum thread documents containing the title and the main question.
6	O_bF_3	Oracle documents containing the class and method names and full text Forum thread documents.
7	O_cF_1	Full text Oracle documents and Forum thread documents containing only the title.
8	O_cF_2	Full text Oracle documents and Forum thread documents containing the title and the main question.
9	O_cF_3	Full text Oracle documents and full text Forum thread documents.

Following, using the index created in Section 2.1 (Phase 4), a series of searches using the content of the pre-processed threads is carried out. Thus, a ranking of relevant Oracle documents is obtained for each discussion thread, sorted according to the *tf/idf* formula [8] implemented by Lucene.

Phase 6: Evaluation of the Performance of the Results. In this phase, the results obtained in the previous phase are contrasted with the classification made by the experts. To evaluate and analyze these results, a series of measures widely used in information retrieval techniques has been selected [9].

Let D_{rlv} all relevant documents in a search and $|D_{rlv}|$ the number of documents in this set. Assuming that the recovery strategy being evaluated produces a set of documents responses D_{rcp}. . Be $|D_{rcp}|$ the number of documents in this set, the following measures are defined:

Precision it is the fraction of retrieved documents (D_{rcp}) that are relevant to the user request.

$$P = \frac{|D_{rlv} \bigcap D_{rcp}|}{|D_{rcp}|}$$

Recall it is the fraction of relevant documents (D_{rlv}) that are successfully retrieved.

$$R = \frac{|D_{rlv} \bigcap D_{rcp}|}{|D_{rlv}|}$$

F-Measure is an assessment measure that combines the above measures (precision and recall) in a single value.

$$F = \frac{2*P*R}{P+R}$$

Regarding to the measures listed above, precision provides information on how many valid (or relevant) documents are retrieved, but not how many valid documents are not recovered. For example, if there are 10 relevant documents and the system retrieves one of them, then a precision of P = 1 is obtained, i.e. 100% of the retrieved documents are valid. However, this measure does not provide information on the number of not recovered valid documents (9).

On the other hand, the recall measure returns the fraction of relevant recovered documents out of the total of recovered documents. In the above example, the recall is R = 0.10 due to a single relevant document is recovered of the 10 expected.

For the example above, a value $F = \frac{2*P*R}{P+R} = \frac{2*1*0.1}{1+0.1}$=0,18 is obtained, which is a more realistic value than the previous measures.

These measures are evaluated using a determined series of *cutoff* values. That is, for each cutoff value N, precision, recall and F-measure are evaluated considering only the first N recovered documents.

3 Case Study 1: *Class Integer*

For these case studies, we have focused on the *Stack Overflow*[3] discussion forum, which is widely used by Java programmers community. Using the search functionality of the site, we have introduced the string <"*Integer class" AND Java*> and downloaded the first 20 retrieved threads. On the other side, we have downloaded all the Java classes specifications from the Oracle site for version 5. Table 7 shows a summary of such information.

Table 7. Implementation characteristics of Case Study 1

	Reference documents	**Discussion forum threads**
Site	Oracle	Stack Overflow
Language	English	English
URL Site	http://docs.oracle.com/javase/1.5.0/docs/	http://stackoverflow.com/
Retrieval date	27/11/2014	27/02/2015
Selection criteria	-	"Integer class" AND Java
Retrieved documents	2953 (all the available for 1.5.0 version)	20

Once retrieved, forum discussion threads have been analyzed in order to identify, among all the Java classes that were mentioned in each thread, which of them were related to the problem the thread deal with. Table 8 shows the result of expert classification using the Likert scale presented above. For example, the thread 8 has been classified in a `high` level with classes *Integer* and *Character*, low level with class *String*, and very low level with *BigInteger* Class.

Due to classes with `low` and `very low` levels of relationship have not been considered relevant for thread classifications, the following evaluation has been focused on threads that had at least one Java class in the three higher levels (`medium, high, very high`). Since thread 16 does not fulfill this condition, the latter phases have been narrowed to the 19 remaining threads. Once the 9 tests presented in Table 5 had been executed, we have noticed that some Oracle documents appeared highly related to most of the threads. Analyzing this phenomena, we have detected that this was because some class names were also common words in the Java programmers vocabulary . For example, the word "*class*" is mentioned in many different contexts and most of the times it is not referred to the Java class named *"Class"*. Something similar happens with *Parameter*, *Error*, etc. The list of such a classes, that we have called *Stopword Classes*, is presented in Table 9.

Once this validity threat has been detected, the Phase 4 has been repeated without including such classes in Oracle document sets (O_a, O_b, O_c), and 9

[3] http://stackoverflow.com/

Table 8. Expert classification for threads used in Case Study 1

Thread	Very high	High	Medium	Low	Very low
3	Integer JTable			String, Object Component	JScrollPane JCombobox JFrame
6	Integer		Long, Boolean Character Double, Short Float, Byte		
8		Integer, Character		String	BigInteger
12	Integer	Long	String		
16					String, Integer
18	String	Integer			
19			Integer		Float, Double

Table 9. Stopword Classes list

Stopword Class Names							
Any	Byte	Class	Doc	Error	Exception	HTML	Key
Method	Object	Operation	Option	Package	Parameter	Point	Result
Set	Source	System	Text	Time	Type	Types	View
Void							

new tests has been carried out (10 to 18). For this reason, Table 6, have been redefined to add tests 10 to 18, as it is shown in Table 10. The new tests have been named adding the postfix "*SW*". For example, the test 10 (O_aF_1*SW)* is the replication of test 1 *(*O_aF_1) without considering the Stopword Classes.

3.1 Analysis of Case Study 1 Results

Aforementioned, measures evaluation must be done considering cutoff values. To settle such values, first we have calculated the media of related classes for all the threads and obtained the value 1.94; so we have determined N=2 as the initial cutoff value. Following, we have analyzed Table 4, where it can be seen that thread 6 is the one with more related Java classes (8). Therefore, even when in IR literature the normal cutoff scale is 5, 10, 20, 30, 50, 100, etc., because of the characteristics of this case study, we have considered a more appropriated scale (2, 3, 4, 5, 10), which allows a better evaluation for the first N results returned by Lucene.

Following, in Table 11 the results for the 18 tests are shown, comparing the automatic versus expert classification, with cutoff values N=2, 3, 4, 5 and 10. As it can be seen, tests 8, 17 and 18 have obtained a value F = 0 when cutoff is 2, which means that non relevant documents were retrieved in the first and second place of Lucene ranking.

Table 10. Tests set for Case Study 1

#	Test	Filtering SW	Description
1	O_aF_1	NO	Oracle documents containing only the class name and Forum thread documents containing only the title, without filtering *Stopword Classes*
2	O_aF_2	NO	Oracle documents containing only the class name and Forum thread documents containing the title and the main question, without filtering *Stopword Classes*
3	O_aF_3	NO	Oracle documents containing only the class name and full text Forum thread documents, without filtering *Stopword Classes*
4	O_bF_1	NO	Oracle documents containing the class and method names and Forum thread documents containing only the title, without filtering *Stopword Classes*
5	O_bF_2	NO	Oracle documents containing the class and method names and Forum thread documents containing the title and the main question, without filtering *Stopword Classes*
6	O_bF_3	NO	Oracle documents containing the class and method names and full text Forum thread documents, without filtering *Stopword Classes*
7	O_cF_1	NO	Full text Oracle documents and Forum thread documents containing only the title, without filtering *Stopword Classes*
8	O_cF_2	NO	Full text Oracle documents and Forum thread documents containing the title and the main question, without filtering *Stopword Classes*
9	O_cF_3	NO	Full text Oracle documents and full text Forum thread documents, without filtering *Stopword Classes*
10	O_aF_1SW	YES	Oracle documents containing only the class name and Forum thread documents containing only the title, filtering *Stopword Classes*
11	O_aF_2SW	YES	Oracle documents containing only the class name and Forum thread documents containing the title and the main question, filtering *Stopword Classes*
12	O_aF_3SW	YES	Oracle documents containing only the class name and full text Forum thread documents, filtering *Stopword Classes*
13	O_bF_1SW	YES	Oracle documents containing the class and method names and Forum thread documents containing only the title, filtering *Stopword Classes*
14	O_bF_2SW	YES	Oracle documents containing the class and method names and Forum thread documents containing the title and the main question, filtering *Stopword Classes*
15	O_bF_3SW	YES	Oracle documents containing the class and method names and full text Forum thread documents, filtering *Stopword Classes*
16	O_cF_1SW	YES	Full text Oracle documents and Forum thread documents containing only the title, filtering *Stopword Classes*
17	O_cF_2SW	YES	Full text Oracle documents and Forum thread documents containing the title and the main question, filtering *Stopword Classes*
18	O_cF_3SW	YES	Full text Oracle documents and full text Forum thread documents, filtering *Stopword Classes*

Table 11. Precision (P), *Recall* (R) and F-measure (F) for each test (Case Study 1)

		cutoff 2			cutoff 3			cutoff 4			cutoff 5			cutoff 10		
#	Test	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
1	Oa-F1	0,316	0,395	0,351	0,412	0,612	0,493	0,430	0,664	0,522	0,432	0,682	0,529	0,440	0,682	0,535
2	Oa-F2	0,368	0,515	0,430	0,316	0,647	0,424	0,285	0,726	0,409	0,246	0,726	0,367	0,218	0,805	0,343
3	Oa-F3	0,500	0,693	0,560	0,386	0,752	0,510	0,329	0,822	0,470	0,305	0,928	0,459	0,179	0,954	0,301
4	Ob-F1	0,316	0,401	0,353	0,237	0,401	0,298	0,276	0,559	0,370	0,253	0,559	0,348	0,244	0,682	0,359
5	Ob-F2	0,237	0,360	0,286	0,193	0,439	0,268	0,158	0,465	0,236	0,168	0,542	0,257	0,105	0,700	0,183
6	Ob-F3	0,263	0,377	0,310	0,246	0,482	0,326	0,211	0,535	0,302	0,200	0,632	0,304	0,111	0,691	0,191
7	Oc-F1	0,079	0,112	0,093	0,070	0,138	0,093	0,053	0,138	0,076	0,042	0,138	0,065	0,026	0,191	0,046
8	Oc-F2	0,000	0,000	0,000	0,018	0,026	0,021	0,013	0,026	0,018	0,011	0,026	0,015	0,005	0,026	0,009
9	Oc-F3	0,026	0,026	0,026	0,035	0,053	0,042	0,026	0,053	0,035	0,021	0,053	0,030	0,016	0,070	0,026
10	Oa-F1 SW	**0,816**	**0,638**	**0,716**	**0,842**	**0,682**	**0,754**	**0,842**	**0,682**	**0,754**	**0,842**	**0,682**	**0,754**	**0,842**	**0,682**	**0,754**
11	Oa-F2 SW	0,474	0,594	0,527	0,474	0,700	0,565	0,447	0,726	0,554	0,434	0,726	0,543	0,425	0,752	0,543
12	Oa-F3 SW	0,526	0,700	0,601	0,456	0,829	0,588	0,368	0,836	0,511	0,337	0,868	0,485	0,255	0,901	0,397
13	Ob-F1 SW	0,421	0,454	0,437	0,386	0,507	0,438	0,425	0,559	0,483	0,425	0,559	0,483	0,435	0,682	0,531
14	Ob-F2 SW	0,237	0,360	0,286	0,193	0,439	0,268	0,184	0,489	0,268	0,158	0,515	0,242	0,102	0,673	0,178
15	Ob-F3 SW	0,289	0,404	0,337	0,246	0,482	0,326	0,211	0,535	0,302	0,211	0,638	0,317	0,111	0,691	0,191
16	Oc-F1 SW	0,079	0,112	0,093	0,070	0,138	0,093	0,053	0,138	0,076	0,042	0,138	0,065	0,026	0,191	0,046
17	Oc-F2 SW	0,000	0,000	0,000	0,018	0,026	0,021	0,013	0,026	0,018	0,011	0,026	0,015	0,005	0,026	0,009
18	Oc-F3 SW	0,000	0,000	0,000	0,018	0,026	0,021	0,026	0,053	0,035	0,021	0,053	0,030	0,011	0,053	0,018

In Figure 4 these results are presented in a bar graphic. As it can be seen, test 10 has achieved the highest performance for cutoff N=3, obtaining a precision of 84% and F-measure of 75% (as visible in Table 11).

Analyzing F-measure average (Figure 5), it is clear that performance is higher for all the tests when Stopword Classes have been filtered (tests 10 to 18) in contrast to tests that have not ((1 to 9). This improvement is due to the fact that the ambiguity of Stopword Classes names makes irrelevant documents to be retrieved as relevant. Also, it is visible that tests 10, 11 and 12 (which are the tests O_a) have gotten the highest F-measure average, being test 10 the greatest one. In addition, according to these tests, performance tends to be lower as long as more information from discussion threads is used.

As we have noticed a better performance for tests that filtered Stopword Classes, the further analysis has been focused on tests 10 to 18. Doing

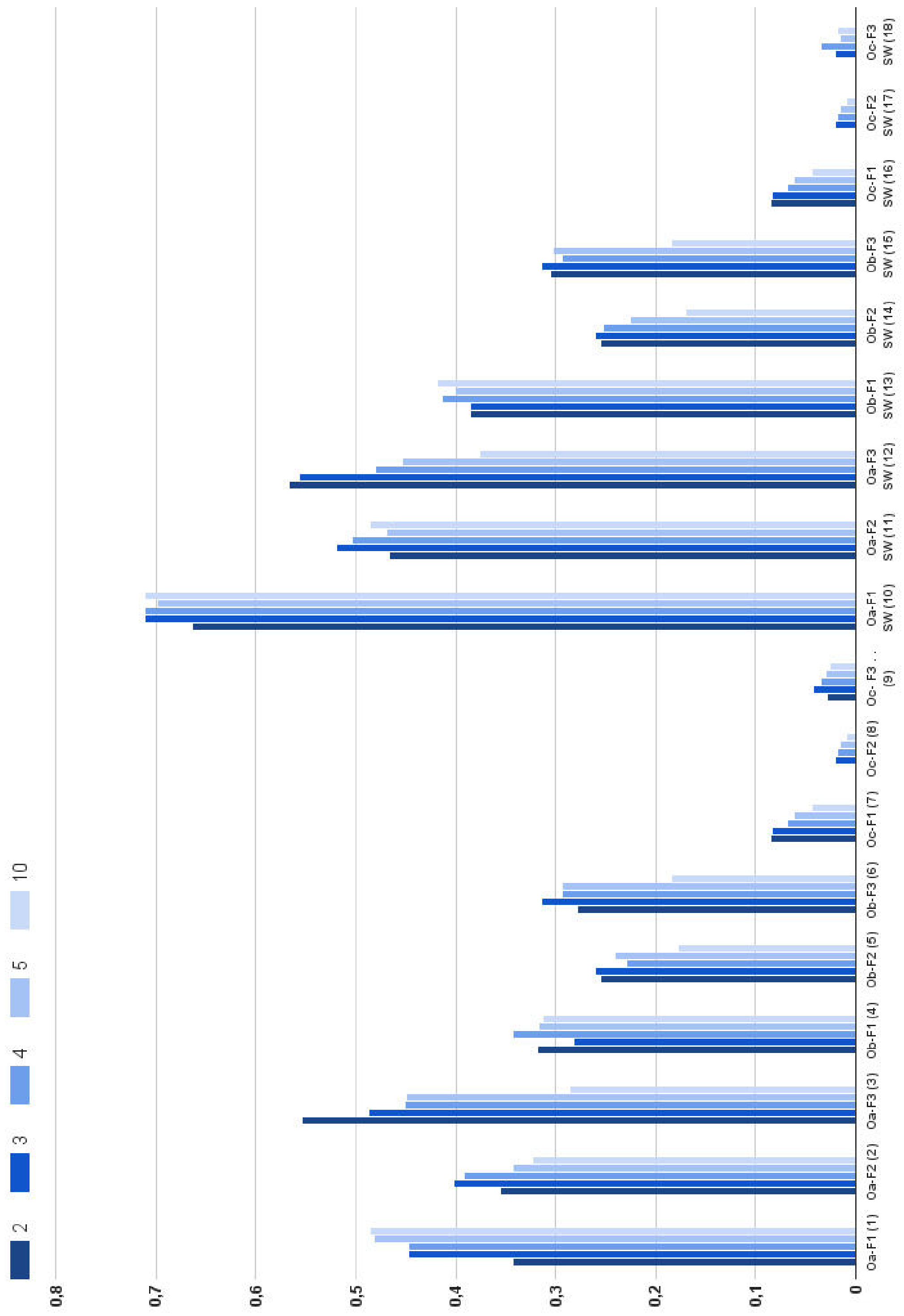

Fig. 4. F-measure by cutoff (Case Study 1)

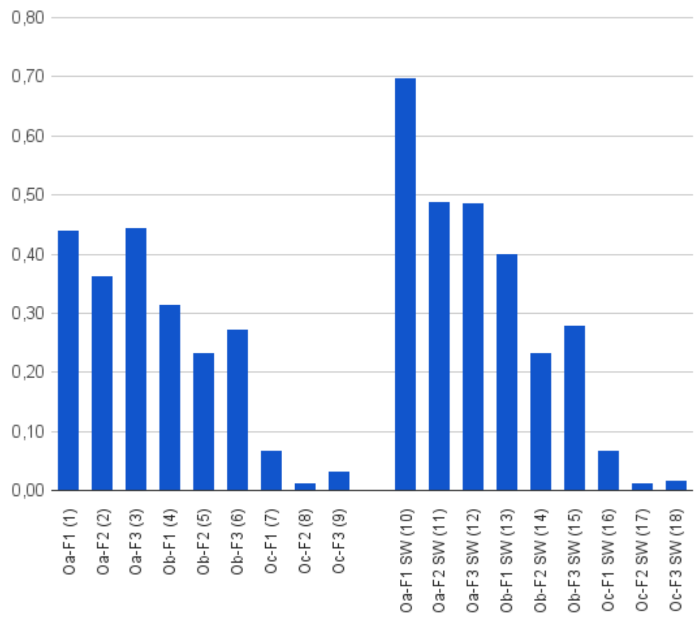

Fig. 5. F-measure media for Case Study 1

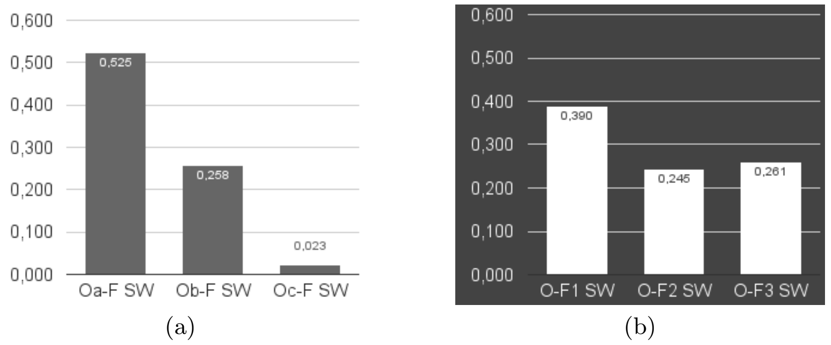

Fig. 6. Performance analysis: (a) group by amount of information in Oracle documents; (b) group by amount of information in forum discussion threads (Case Study 1)

so, Figure 6 shows the performance trend when tests are grouped by the amount of information used in (a) Oracle documents and (b) forum discussion threads. Analyzing such graphics, it can be seen that tests that considered more information from Oracle documents (tests O_c) have a considerable lower performance than tests that considered less information (tests O_a and O_b) – Figure 6 (a). Furthermore, performance of the tests that have used only the class name (tests O_a) is higher than performance of tests that have considered information from other document sections (tests O_b and O_c). This could happen because the name of related classes is mentioned more frequently than method names or other information from the description of the Oracle documents.

On the other hand, even when tests considering only the thread title (tests F_1) have obtained a better performance, it can be seen that using the full text of threads (tests F_3) has got better performance than using the title and main question (tests F_2) – Figure 6 (b).

4 Case Study 2: Tag Java

In order to evaluate the proposed process in a more general context about Java related problems (instead of a particular class) we have extended the study to more threads

Again,the selected discussion forum has been Stack Overflow. In this case, instead of performing a search for a particular class, the tags filter of the site has been used to select the initial set of threads. A *tag* is a keyword that groups related threads within the site Stack Overflow. These tags can be selected from an existing set or added by the user who creates the thread. Since the site offers this facility, we have performed the Case Study 2 focusing on the threads with the tag ”java”. Table 12 summarizes the characteristics of the realization of this phase, indicating the download date, number of retrieved documents for each type, etc., as described previously.

Table 12. Technical characteristics for Case Study 2

	Reference documents	**Discussion forum threads**
Site	Oracle	Stack Overflow
Language	English	English
URL Site	http://docs.oracle.com/javase/1.5.0/docs/	http://stackoverflow.com/
Retrieval date	27/11/2014	27/11/2015
Selection criteria	-	Tag: java
Retrieved documents	2928 (all the available for version 1.5.0 without Stopword Classes)	140

During Phase 2, the same strategy of expert classification has been used. From this analysis, 90 of the 140 retrieved threads have been discarded, because they were not directly related to problems using the Java classes. This happened because, as aforementioned, users choose the tags related to their question, and usually they opt for the label "java" when they use the Java language in a particular environment (Android, JSP, etc.), or programming platforms (NetBeans, Eclipse, etc.). An example of such a classification is shown in Table 13. For instance, the thread STBJava1-14 (Tabla 14) does not expose a problem about the Java language usage, but on which programming platform Java language behave better.

Table 13. Example of expert classification of generic threads for Case Study 2

Threads	**Very High**	**High**	**Medium**	**Low**	**Very Low**	**Related to Java Classes**
STBJava1-1			Arrays, Array	Random		yes
STBJava1-15	LinkedList	ArrayList				yes
STBJava1-25	String	Arrays	Set, HashSet, List			yes
STBJava1-44	Random		String Secure-Random	BigInteger		yes
STBJava1-14						no (Efficiency Netbeans vs Eclipse)
STBJava2-2	String	HashMap, Map			Integer, String	yes
STBJava2-24						no (Android)
STBJava3-12		Pattern, Matcher	String			yes

Similarly to Case Study 1, it has been verified which threads contain at least one Java class in a very high, high or medium levels. Since all the threads met this condition, non threads have been discarded in this phase.

As aforementioned, Case Study 1 best results have been obtained when filtering Stopword Classes. Therefore, in Case Study 2 the analysis has been restricted to the 9 tests that correspond to tests 10 to 18 in Table 10.

4.1 Analysis of the Results of the Case Study 2

In order to establish cutoff values for this case study, as the maximum number of Java classes related to a thread is 5, whereby it has been determined N = 5 as

Table 14. Sample discussion thread to be classified(STBJava1-14)

Link	http://stackoverflow.com/questions/21947452/why-is-printing-b-dramatically-slower-than-printing
Title	Why is printing “B” dramatically slower than printing “#”?
Question	I generated two matrices of 1000 x 1000: First Matrix: O and #. Second Matrix: O and B. Using the following code, the first matrix took 8.52 seconds to complete: `Random r = new Random(); for (int i = 0; i < 1000; i++) { for (int j = 0; j < 1000; j++) { if(r.nextInt(4) == 0) { System.out.print("O"); } else { System.out.print("#"); } } System.out.println(""); }` With this code, the second matrix took 259.152 seconds to complete: `Random r = new Random(); for (int i = 0; i < 1000; i++) { for (int j = 0; j < 1000; j++) { if(r.nextInt(4) == 0) { System.out.print("O"); } else { System.out.print("B"); //only line changed } } System.out.println(""); }` What is the reason behind the dramatically different run times? As suggested in the comments, printing only System.out.print("#"); takes 7.8871 seconds, whereas System.out.print("B"); gives still printing.... As others who pointed out that it works for them normally, I tried Ideone.com for instance, and both pieces of code execute at the same speed. Test Conditions: I ran this test from Netbeans 7.2, with the output into its console I used System.nanoTime() for measurements
Answer	I performed tests on Eclipse vs Netbeans 8.0.2, both with Java version 1.8; I used System.nanoTime() for measurements. Eclipse: I got the same time on both cases - around 1.564 seconds. Netbeans: Using "#": 1.536 seconds Using "B": 44.164 seconds So, it looks like Netbeans has bad performance on print to console.

the maximum cutoff value. Also, the average of valid responses for each thread has been calculated, obtaining a value of 2.34, so as N = 2 has been settle as the initial cutoff. Therefore, the cutoff values for Case Study 2 has been defined as 2, 3, 4 and 5.

Table 15 shows the results of the 9 tests of this case study and their evaluation for the defined cutoff values. As it can be seen, there are no tests where the F value is 0, as in the Case Study 1 for cutoff N = 2. That means that it has been recovered at least one relevant Java class for each cutoff.

Table 15. Precision (P), *Recall* (R) and F-measure (F) for each test (Case Study 2)

		cutoff 2			cutoff 3			cutoff 4			cutoff 5		
#	Test	P	R	F	P	R	F	P	R	F	P	R	F
1	O_aF_1SW	0,650	0,420	0,489	0,653	0,427	0,495	0,653	0,427	0,495	0,653	0,427	0,495
2	O_aF_2SW	**0,660**	**0,538**	**0,571**	**0,593**	**0,584**	**0,564**	**0,558**	**0,619**	**0,556**	**0,543**	**0,637**	**0,550**
3	O_aF_3SW	0,580	0,555	0,541	0,467	0,639	0,516	0,385	0,684	0,473	0,340	0,741	0,450
4	O_bF_1SW	0,430	0,357	0,365	0,400	0,393	0,363	0,388	0,417	0,360	0,382	0,430	0,357
5	O_bF_2SW	0,520	0,486	0,481	0,413	0,542	0,447	0,352	0,582	0,416	0,315	0,612	0,392
6	O_bF_3SW	0,540	0,506	0,498	0,393	0,541	0,435	0,335	0,604	0,414	0,284	0,646	0,380
7	O_cF_1SW	0,230	0,215	0,213	0,167	0,245	0,190	0,150	0,283	0,189	0,136	0,333	0,187
8	O_cF_2SW	0,190	0,173	0,174	0,153	0,203	0,169	0,135	0,233	0,166	0,120	0,253	0,159
9	O_cF_3SW	0,220	0,223	0,211	0,200	0,280	0,223	0,180	0,334	0,224	0,160	0,365	0,214

Figure7 shows the obtained F-measure values for each cutoff while Figure 8 presents the F-measure average.

As it can be seen, the best performance has been obtained when considering only the class name in Oracle documents (O_a) and the title and main question in threads (F_2). This is true even when considering the worst cutoff for this case study (N = 5).

In addition, by analyzing the trend of the values in Figure 7, it can be seen that when full text of Oracle documents has been considered (O_c), the performance remains below the values obtained when less information is used (O_a, O_b). Also, it is important to mention that tests considering the class name or class and method names (O_a, O_b), show the best performance when more information from threads is used (F_3). This performance improves as long as the cutoff is smaller.

Figure 9 shows the performance trends in this case study, grouped by the amount of information contained in (a) Oracle documents and (b) discussion threads. By analyzing these figures it can be seen that as long as the information contained in the documents Oracle is higher, the performance tends to decrease slightly – Figure 9 (a).

On the other hand, considering the amount of information contained in discussion threads – Figure 9 (b) – it can be seen that the performance is lower when considering just the title of the thread (cases F_1) than when more information is considered (cases F_2 and F_3). This may happen because, when

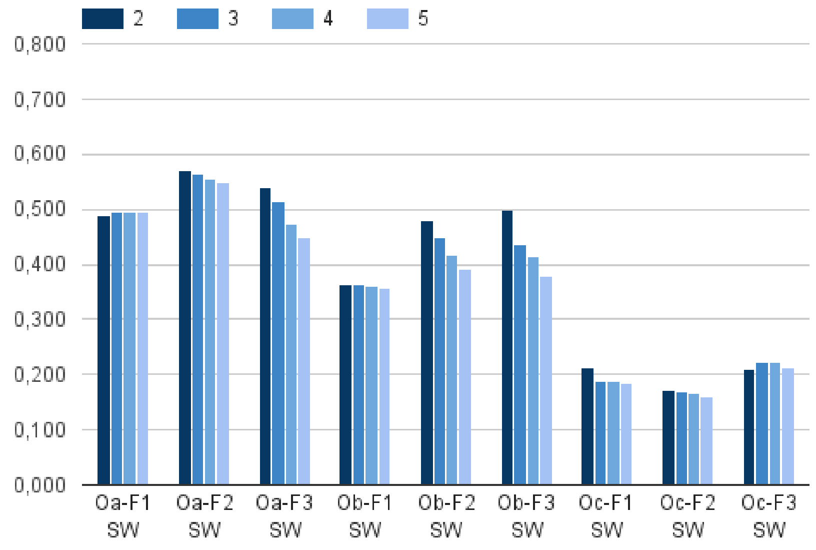

Fig. 7. F-measure by cutoff (Case Study 2)

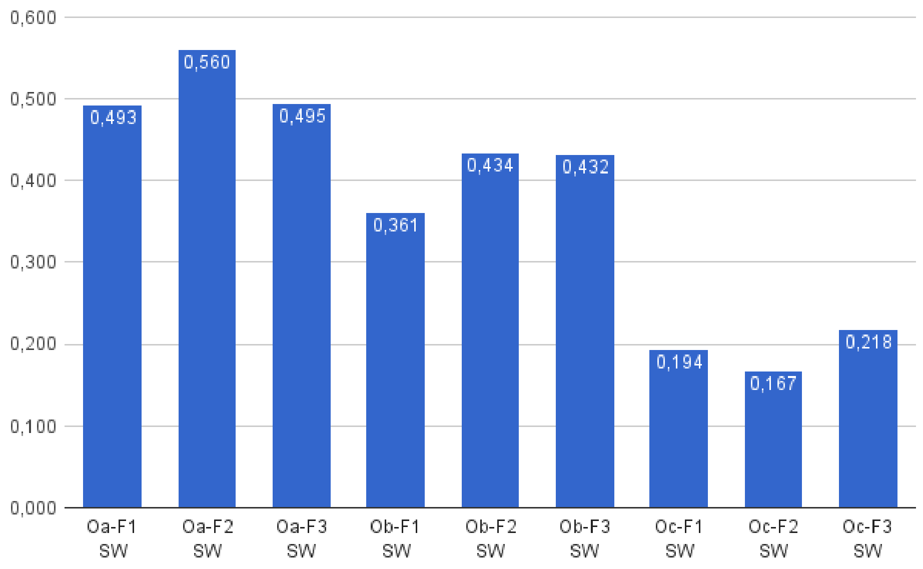

Fig. 8. F-measure media for Case Study 2

threads refer to general Java problems (as opposite to Case Study 1 that explicitly mention the Integer Class), the class names usually do not appear in the title thereof. In addition, it can be seen that the difference between the performance for tests F_2 and F_3 (0.387 and 0.382 respectively), is too narrow to reach a conclusion about it.

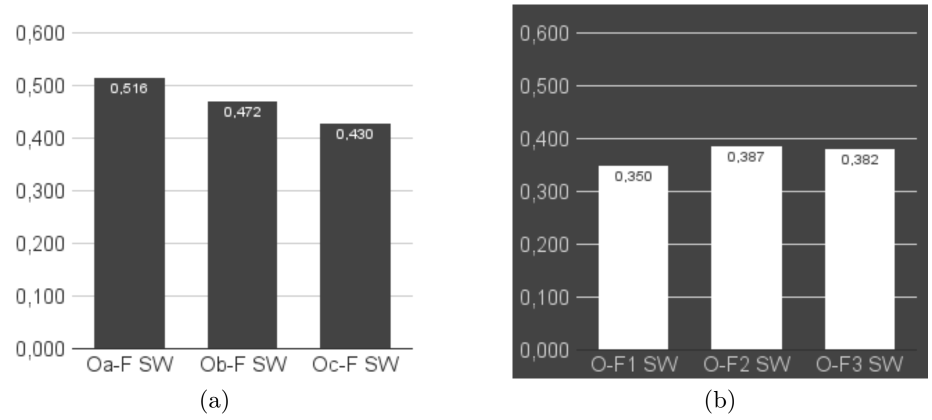

Fig. 9. Performance analysis: (a) group by amount of information in Oracle documents; (b) group by amount of information in forum discussion threads (Case Study 2)

5 Comparative Analysis of the Results

Following, a comparative analysis of the results of the case studies is shown. Then, considering the first assumption:

– Assumption A: Classifying forum discussion threads into Java classes hierarchy is better (more precise) when more information from Oracle Java class documents is used.

Figure 10 shows the results of both case studies grouped by the amount of information in the Oracle documents. In general, it can be seen that in both cases, increasing the amount of information in these documents decreases the performance of information retrieval. By analyzing the difference between the results of both case studies, a significant difference between tests O_b (0.258 in Case Study 1 and 0.472 in Case Study 2) is observed. This difference is even more remarkable for tests O_c (0.023 in Case Study 1 and 0.430 in Case Study 2). However, this difference is negligible for tests O_a (0.525 in Case Study 1 and 0.516 in Case Study 2), which are the tests that have obtained the best performance in both case studies.

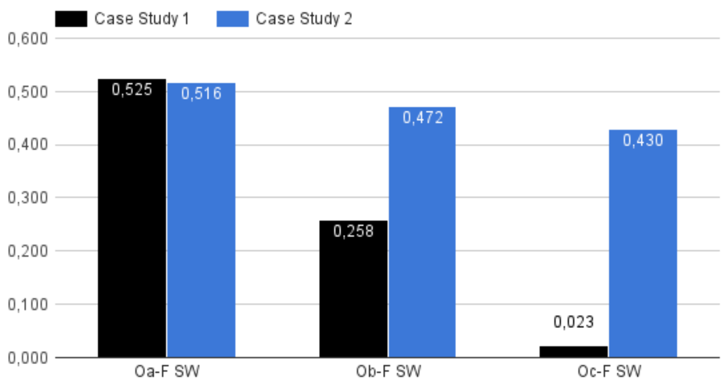

Fig. 10. Performance group by amount of information in Oracle documents

The trend in both case studies contradicts the Assumption A, indicating that Oracle documents containing only the class name allow a better performance when they are used as reference documents. In addition, as more information is considered from the Oracle documents, performance tends to be lower, which would prove the opposite of the assumption, that is, the greater the information is the lower performance is obtained. This could happen because the name of the related classes is often directly and more frequently mentioned in discussion threads, than the methods or other information expressed in the class description.

Regarding the second assumption:

– Assumption B: Classifying forum discussion threads into Java classes hierarchy is better (more precise) when more information about the problem described in each thread is used.

Figure 11 shows the results obtained in both case studies grouped by the amount of information considered from the discussion threads. Comparing the results, it is observed that in Case Study 1 the best performance is obtained when considering only the title of the thread (F_1 tests). By contrast, in Case Study 2 this happens when the title and the main question are used (tests $F_2 = 0.387$). Furthermore, in Case Study 1, the performance of tests F_1 (0.390) shows a marked difference with the other tests ($F_2 = 0.245$ and $F_3 = 0.261$), while for Case Study 2, the best performance ($F_2 = 0.387$) has a little difference with respect to the others ($F_1 = 0.350$, $F_3 = 0.382$).

On the other hand, when analyzing the absolute difference between the performance of tests F_2 and F_3, for both case studies, (Case Study 1: $|$ 0245-0261 $| = 0.016$) and Case Study 2: $|$ 0387-0382 $| = 0.005$), it can be seen that it is maintained at a low level. This difference is slightly higher for tests F_1 ($|$ 0390-0350 $| = 0.040$) but also remains being low.

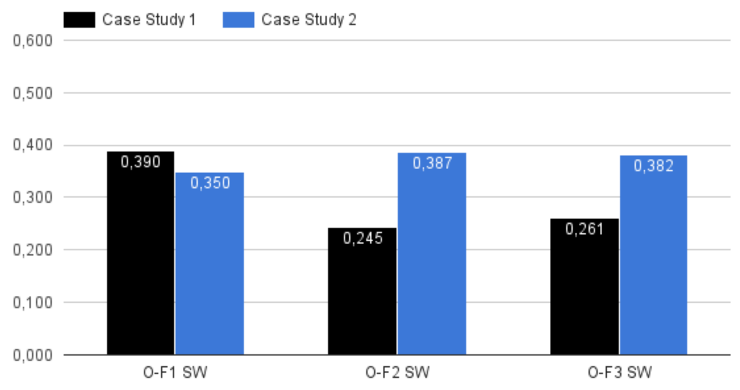

Fig. 11. Performance comparison group by amount of information in discussion threads

From the above, it is not possible to draw any conclusion to accept or reject the raised Assumption B or its inverse. When comparing both case studies, it is striking that tests F_1 have obtained the best performance for Case Study 1 and the worst performance for Case Study 2. This may happen because when looking for a particular class (Integer) it is highly probable that the name of the class would be included in the title of the thread; while when asking a more general question, the related classes are more likely to be referred to in the main question or the answers of the thread.

6 Related Work

There are several proposals for knowledge reuse in discussion forums. For example, Chen et al. proposal [10] automatically analyzes the threads of a discussion forum of an Artificial Intelligence course and recommends a list of related threads written by former students. On the other hand, Helic and Scerbakov proposal [11] classifies messages in a discussion forum according to a predefined topics hierarchy. Both proposals are intended for a collaborative learning domain. In contrast, our recommender is focused on a wider context, involving users with different background about the topic. Finally, these studies used an unique forum, the standard format of information can be ensured, whereas our proposal aims to collect information from several forums, therefore heterogeneity format is an extra challenge.

Considering discussion forum classification using information retrieval tools, Nicoletti et al. [12] focuses on classifying each message in a discussion thread according to a topic hierarchy extracted from the Wikipedia. In this case, this work differs from our proposal since it focuses at a message level instead of threads.

7 Conclusion and Future Work

In this paper, we have presented two case studies in order explore strategies to classify technical discussion forum threads into a set of reference documents. The thread analysis was restricted to Java programming language queries within the Stack Overflow discussion forum. Different strategies to pick up threads were followed: In Case Study 1, the threads were recovered as they contained the string "Integer Class AND Java", whereas in Case Study 2, the threads contain a "java" tag chosen by forum users. In both cases, threads were classified into the same set of reference documents: the Oracle specification documents for Java classes.

The classification was done considering the amount of information contained in the different sections of both types of documents. In the second part of Case Study 1 and Case Study 2 Java classes whose names are common vocabulary of Java programmers were filtered as they make results confusing.

The results of both case studies would indicate that, unlike the proposed assumptions, classification would be more precise when Oracle documents contain only the class name. On the other hand, results are not conclusive according to assumption B, since Case Study 1 worked better for documents containing just the thread title and Case Study 2 did it best with documents containing the title and the main question, with a slight difference over the other two versions. Replication of both case studies is needed in order to to verify the observed trend.

Since these results come from a set of threads which is restricted to a single forum, future work will focus on replicate them using a wider set of threads from other forums as well as other techniques to improve the information retrieval process, to ensure the generality of these results.

Also, the measures used to evaluate the results in this paper consider the retrieved documents as a set, regardless their order of importance. In the future it is planned to extend the analysis to other types of metrics to consider the ranking of relevant documents returned by Lucene.

Acknowledgements

This work is supported by the sub-project "*Knowledge reuse in technical discussion forums*", of 04/F001 Research Program "*Reuse oriented development*", and the project 04/F006 "*Intelligent agents in dynamics environments*", at Comahue National University (Neuquén, Argentina).

References

1. J. Dorn, "Social Software (and Web 2.0)," in *Social Computing: Concepts, Methodologies, Tools, and Applications* (S. Dasgupta, ed.), pp. 305–311, PA: Information Science Reference, 2010.

2. C. A. Ellis, S. J. Gibbs, and G. L. Rein, "Groupware: Some Issues and Experiences," *Communications of ACM*, vol. 34, no. 1, pp. 38–58, 1991.
3. S. Gottipati, D. Lo, and J. Jiang, "Finding relevant answers in software forums," in *26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, KS, USA, November 6-10, 2011*, pp. 323–332, 2011.
4. M. V. Zelkowitz, D. R. Wallace, and D. W. Binkley, "Experimental validation of new software technology," in *Lecture Notes on Empirical Software Engineering* (N. Juristo and A. M. Moreno, eds.), pp. 229–263, River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2003.
5. C. Wohlin, P. Runeson, M. Hšt, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering.* Computer Science, Springer, 2012.
6. R. van Solingen and E. Berghout, *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development.* McGraw-Hill, 1999.
7. C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing.* Cambridge, MA, USA: MIT Press, 1999.
8. T. Rölleke and J. Wang, "TF-IDF Uncovered: A Study of Theories and Probabilities," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, (New York, NY, USA), pp. 435–442, ACM, 2008.
9. R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.
10. W. Chen and R. Persen, "A recommender system for collaborative knowledge," in *2009 Conference on Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modelling*, (Amsterdam, The Netherlands, The Netherlands), pp. 309–316, IOS Press, 2009.
11. D.Helic and N. Scerbakov, "Reusing Discussion Forums as Learning Resources in WBT Systems," in *IASTED International Conference Computers and Advanced Technology in Education*, (Rhodes, Greece), pp. 223 – 228, 2003.
12. M. Nicoletti, S. Schiaffino, and D. Godoy, "Mining Interests for User Profiling in Electronic Conversations," *Expert Syst. Appl.*, vol. 40, pp. 638–645, Feb. 2013.