CLEI ELECTRONIC JOURNAL, VOLUME 18, NUMBER 2, PAPER 2, AUGUST 2015

Applying Information Retrieval Techniques to Detect Duplicates and to Rank References in the Preliminary Phases of Systematic Literature Reviews

Ramon Abilio **Flávio Morais**
IT Department - Federal University of Lavras,
Lavras, MG, Brazil, 37200-000
{ramon.abilio, flavio}@dgti.ufla.br

Gustavo Vale
Department of Computer Science - Federal University of Minas Gerais,
Belo Horizonte, MG, Brazil, 31270-010
gustavovale@dcc.ufmg.br

Claudiane Oliveira **Denilson Pereira** **Heitor Costa**
Department of Computer Science - Federal University of Lavras,
Lavras, MG, Brazil, 37200-000
claudiane@posgrad.ufla.br, {denilsonpereira, heitor}@dcc.ufla.br

Abstract

Systematic Literature Review (SLR) is a means to synthesize relevant and high quality studies related to a specific topic or research questions. In the Primary Selection stage of an SLR, the selection of studies is usually performed manually by reading title, abstract, and keywords of each study. In the last years, the number of published scientific studies has grown increasing the effort to perform this sort of reviews. In this paper, we proposed strategies to detect non-papers and duplicated references in results exported by search engines, and strategies to rank the references in decreasing order of importance for an SLR, regarding the terms in the search string. These strategies are based on Information Retrieval techniques. We implemented the strategies and carried out an experimental evaluation of their applicability using two real datasets. As results, the strategy to detect non-papers presented 100% of precision and 50% of recall; the strategy to detect duplicates detected more duplicates than the manual inspection; and one of the strategies to rank relevant references presented 50% of precision and 80% of recall. Therefore, the results show that the proposed strategies can minimize the effort in the Primary Selection stage of an SLR.

Keywords: Systematic Literature Review; Information Retrieval; Vector Model; Primary Selection.

1 Introduction

Systematic Literature Review (SLR) is a means to evaluate and interpret relevant work available in the literature regarding a particular research question, a topic area or a phenomenon of interest [1][2][3][4]. SLR can be performed basically in three phases [3]: i) Planning; ii) Execution; and iii) Analysis of Results. These phases can be subdivided in stages, for example, the Planning phase has the stage of Development of the Protocol used in the SLR, and the Execution phase has the stages of Primary Selection and Secondary Selection of Studies.

In the Planning phase of an SLR, we define terms (words and/or expressions) and their synonyms related to the topic or research question that will be used for searching relevant works. One term and its synonyms are connected by the logical operator OR to create a group of terms of search. Each group is connected by the logical operator AND in order to create the search string. This string can be used in online repositories of scientific studies that provide search engines, aiming at retrieving works whose content has at least one term of each group.

In the Primary Selection stage, we use the string in the search engines, export the results and organize them to allow the exclusion of duplicates and references to materials that are not papers (non-papers), such as table of contents and abstracts of proceedings. After that, the references are manually selected by reading their attributes, such as title, abstract, and keywords. In the Secondary Selection stage, the studies selected in the Primary Selection are fully read. In both selections, we use the inclusion and exclusion criteria, defined in the Planning phase, to select or not a study [3]. The results of the Secondary Selection are the studies considered relevant in the context of the research. One problem in the Primary Selection is the effort necessary to perform the activity due to the increased amount of studies [4]. In order to reduce the bias in the selection of the studies, at least two researchers have to read the title, the abstract, the keywords, and, when necessary, the full text [3].

Aiming to minimize this effort and help the researchers in the Primary Selection, several proposals were presented to rank references (documents) with regarding their relevance, automate the selection, and validate the selected studies [4][5][6][7][8]. In this work, we proposed the use of Information Retrieval techniques to detect duplicated references and to rank references aiming to detect materials that are not papers (non-papers). Information Retrieval techniques provide easy access to the information of interest of the users. Information Retrieval models are created to provide a ranking function that assigns points to documents regarding a given query [9]. One of the classic models of Information Retrieval is the Vector Model, an algebraic model that represents documents and queries as vectors in a t-dimensional space [9].

In this work, we treated each reference obtained from the results exported by search engines as a document. Our proposal covers the three activities of the Primary Selection: detection of non-papers, detection of duplicates, and ranking of references. The detection of non-papers analyzes the title and ranks the references regarding a query that has terms present in references for non-papers. The detection of duplicates is based on the analysis of similarity of references considering title and abstract. The ranking of references is based on their relevance. We proposed two strategies to rank the references considering the search string, title, abstract and keywords. We evaluated the strategies performing an experimental evaluation using two real datasets obtained from two SLRs published in 2012 [10] and 2014 [11]. Both SLRs are related to Software Engineering. The results show that the proposed strategies can minimize the effort in the Primary Selection stage.

This work extends a previous work [12] – which presented the strategies to rank references regarding their relevance considering the search string – by adding strategies to detect duplicated references and non-papers, as well as an experimental evaluation using one more dataset. The main contributions of this work are:

- A strategy to automate the detection of duplicates using the Jaccard Coefficent Similarity [14];
- A strategy to automate the detection of non-papers using the Vector Model;
- Two strategies using the Vector Model to rank the references used in a Systematic Literature Review regarding their relevance considering the terms used in the search string;
- The possibility of using the proposed strategies for assessing the quality of the search string;
- The identification of unexpected behavior of search engines related to the Boolean query, since the references may be returned without considering the union of the search terms with the AND connector;
- An initial collection of terms present in non-papers.

The remainder of this paper is organized as follows. Section 2 briefly discusses some related work. Section 3 presents contextualization of Systematic Literature Review and Information Retrieval. The proposed strategies to detect duplicates and non-papers, and for the ranking of references are described in Section 4. Section 5 presents the experimental evaluation setting and results, and Section 6 discusses the results. Section 7 presents the conclusions and the suggestions of future work.

2 Related Work

In a study that aims to reduce the effort in conducting an SLR in the context of Evidence-based Software Engineering [1], a tool was proposed to act as an interface to perform searches in databases, such as IEEExplore[1], ACM Digital Library[2], Scopus[3] and Science Direct[4]. To obtain relevant studies, the authors proposed to use techniques of information retrieval, such as, text-based techniques for natural language processing and clustering of similar studies by using the clustering techniques K-means or k-NN. To evaluate the proposed techniques, the authors suggested the use of the Recall and Precision measures. The authors did not implement the proposed strategies, but mentioned the implementation as future work [4].

In another study [5], Linked Data, Text Mining and Naïve Bayes techniques were used in an iterative and supervised process used in the Secondary Selection to reduce manual work and the bias of subjectivity. The proposal was to extend and enrich the basic process of the Primary Selection using existing technologies in the area of Semantic Web and Text Mining in the context of Linked Data technique. To evaluate the proposal, an experiment was conducted by using information from an SLR published in 2007. In this case study, manual work in the search and in the Primary Selection of the review was simulated. The relevant studies from the Primary Selection were used to create the prototype and run the proposed technique. The results indicate that the used technique can reduce the effort by 20% with a recall of 100%.

To provide support in verifying the results of the Secondary Selection, the technique Visual Text Mining (VTM) was used [6]. This technique supports tasks that involve interpretation of large amount of textual data and the integration between Text Mining and Information Visualization techniques. The purpose of the study was to modify the execution phase, adding two VTM techniques to support the review of the selected studies (content map and citation map). Case studies were performed to verify the applicability of the technique, and the results indicated that VTM can help in the Primary Selection accelerating the process and increasing the reliability of the results.

In another study [7], a semi automated triage was performed on the process of selecting papers. In this triage, Support Vector Machines (SVM) were used. SVM can be constructed in different spaces, for example, using abstract and title or full-text. To semi automate the triage process is difficult due to the imbalance of irrelevant papers on relevant papers. Therefore, an active learning strategy was developed for imbalanced data sets. The triage was performed in three Systematic Literature Reviews. In two of them, the algorithm was able to reduce the number of manually selected references by 50%, and, in the third study, around 40%. In these studies, there was no compromise in the quality and scope.

In order to reduce the effort in reading the references in an SLR in the healthcare area, a ranking of relevant references was performed in two phases [8]: i) text retrieval; and ii) re-ranking via text classification. In the first step, constraints are added using various controls, for example, .mp control, that look for a particular term in the title, abstract and MeSH heading. In the second step, the information is obtained from the previous step and the references are ordered hierarchically using Support Vector Regression (SVR). In this work, the authors used the machine learning tool Weka[5] to apply the proposed technique in 17 SLRs, and the references considered relevant were ranked among the first 30% of the ranked references.

The aim of this study is to apply Information Retrieval techniques in the Primary Selection stage, as well as, in the three studies [4][7][8], and differently from the two studies [5][6], in which the focus was to provide support on the Secondary Selection stage. Regarding studies focusing on Primary Selection, one of them [4] proposed a tool, in which Information Retrieval techniques would be implemented, but the tool was not developed and evaluated. In this work, we implemented the strategies and evaluated their applicability. In two studies, the SVM [7] and the SVR [8] techniques were used, while in this work, we used the Vector Model.

In this paper, we proposed three strategies based on the cosine similarity of the Vector Model and one strategy using Jaccard [14] to help researchers in the Primary Selection stage, reducing their effort in the detection of duplicates

[1] http://ieeexplore.ieee.org/
[2] http://dl.acm.org/
[3] http://www.scopus.com/
[4] http://www.sciencedirect.com/
[5] http://www.cs.waikato.ac.nz/ml/weka

and non-papers, and ranking references regarding the search string terms. As input, these strategies use the data of the SLR protocol and the collection of references generated by the execution of the search string, in other words, the overhead for researchers regarding the application of the techniques is minimal.

3 Background

This section briefly presents the Systematic Literature Review process and Information Retrieval concepts.

3.1 Systematic Literature Review

Systematic Literature Review (SLR) is a component of evidence-based practice [1][3], and it is part of what is called Secondary Studies in systematic research. Performing an SLR, the research, selection, analysis and organization of studies are facilitated due to the pre-defined steps and criteria [4]. With this, the researchers can organize the performing of the technique and, in addition, it is possible to find relevant content on a research topic in an electronic database with lots of information. The execution of an SLR is based on three phases (Figure 1) [2][3][13]:

- **Planning**. The reason to perform the SLR is established in this phase. The stages of this phase are: i) describing the research motivations and objectives; ii) defining research questions related to the motivations and objectives; iii) developing the research protocol that will be applied to the search; and iv) assessing the protocol to verify the SLR applicability;
- **Execution**. A search in the databases defined in the previous phase is performed. The researchers classify the obtained references guided by the inclusion and exclusion criteria. The stages of this phase are: i) searching references in the defined databases and organizing the results; ii) Primary Selection: the researchers perform a first selection of studies reading title, keywords, and abstracts of the references verifying their compliance with the criteria; iii) Secondary Selection: a second selection is performed aiming to eliminate unsuitable studies; and iv) organizing the selected references to be analyzed;
- **Analysis of Results**. The studies referenced by the selected references are read, the data collected, and analyzed. The stages of this phase are: i) collecting and organizing data: the researchers read the studies selected in the Secondary Selection stage to collect and analyze the data that answer the research questions; and ii) assessing the results: the data are reviewed and the report is published.

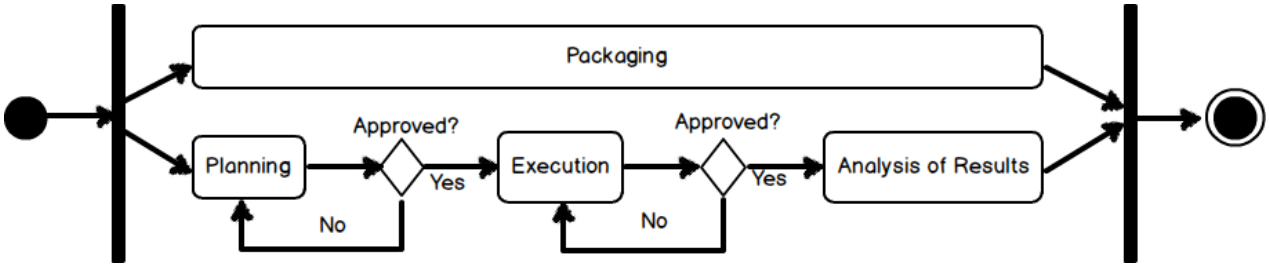

Figure 1: Process of Systematic Literature Review (Adapted from [13])

3.2 Information Retrieval

Information Retrieval deals with the representation, storage, organization, and access to information items, such as documents, web pages, online catalogs, structured and semi-structured records, and multimedia objects [9]. Information Retrieval Modeling is a complex process that produces a ranking function that assigns weights to documents regarding to a given query [9]. The Vector Model is a classic model of Information Retrieval. It is an algebraic model that represents documents and queries as vectors in t-dimensional space, in which t is the number of distinct terms (words or expressions) in a collection [9].

Let D be a collection of N documents (1) and let K be the set of t distinct terms that appear in documents of D (2). For each pair (k_i, d_j), $k_i \in K$ e $d_j \in D$, it is associated a weight $w_{i,j} \geq 0$. A document d_j is represented as a vector of weights (3). Each $w_{i,j}$ represents the importance of the term k_i to describe the semantic content of the document d_j.

$$D = \{d_1, d_2, ..., d_n\} \quad (1)$$

$$K = \{k_1, k_2, ..., k_t\} \quad (2)$$

$$\overrightarrow{d_j} = \{w_{1,j}, w_{2,j}, ..., w_{t,j}\} \quad (3)$$

A scheme used to calculate the weights, known as TF-IDF (Term-Frequency x Inverse Document Frequency), consider the frequency of the term in the document (TF) and the rarity of the term in the collection (IDF). The degree of similarity between a query q and one document d is evaluated as the correlation between the vectors $\overrightarrow{d_j}$ and $\vec{q}$. This correlation can be quantified by the cosine of the angle between those two vectors (4). This similarity measure is known as Cosine Similarity [9].

$$sim(d,q) = \frac{\sum_{i=1}^{t} w_{i,d} \cdot w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,d}^2} \cdot \sqrt{\sum_{i=1}^{t} w_{i,q}^2}} \quad (4)$$

After obtaining the results of the ranking function, it is important to evaluate their quality. Typically, this evaluation is the comparison of the results of the Information Retrieval system with the results proposed by experts that reviewed the collection of documents [9]. Two classical measures to evaluate Information Retrieval systems are Precision and Recall [9]. Formally, these measures are: q a query from a reference collection, R the set of documents judged as relevant to q and, |R| the number of documents in R. By submitting q, we obtain a set of results A (|A| documents). Considering $|R \cap A|$ the number of documents in the intersection between R and A, the precision (5) can be defined as the fraction of retrieved documents that are relevant, and the recall (6) as the fraction of relevant documents retrieved.

$$Precision = \frac{|R \cap A|}{|A|} \quad (5)$$

$$Recall = \frac{|R \cap A|}{|R|} \quad (6)$$

The Jaccard Coefficient Similarity (7) is a function based on words (tokens) used to quantify the similarity and diversity between sets of strings [14]. It is the result of the intersection divided by the result of the union of the sets of evaluated strings.

$$Jaccard_{(A,B)} = \frac{|A \cap B|}{|A \cup B|} \quad (7)$$

4 Strategies to Rank References and to Detect Non-Papers and Duplicates

Figure 2 depicts a Primary Selection stage process. In this process, researchers have to deal with hundreds or thousands of references exported by different search engines. This collection of references has papers and non-papers, such as, Table of Contents and Copyright Notice, and can be added in a database (Sum database). Due to the search in more than one source (or machine), the number of duplicates increases and they have to be removed (Union database). After that, the researchers have to remove references for non-papers obtaining a database composed only by references for papers (Papers database). In this last database, the ranking by relevance performed by the search engines is lost when the results are aggregated due to the joint of the exported references, and the database manipulation for removal of

duplicates and non-papers. This section presents the strategies that we propose to filter only papers for full-text read and to rank the references regarding to the search string.

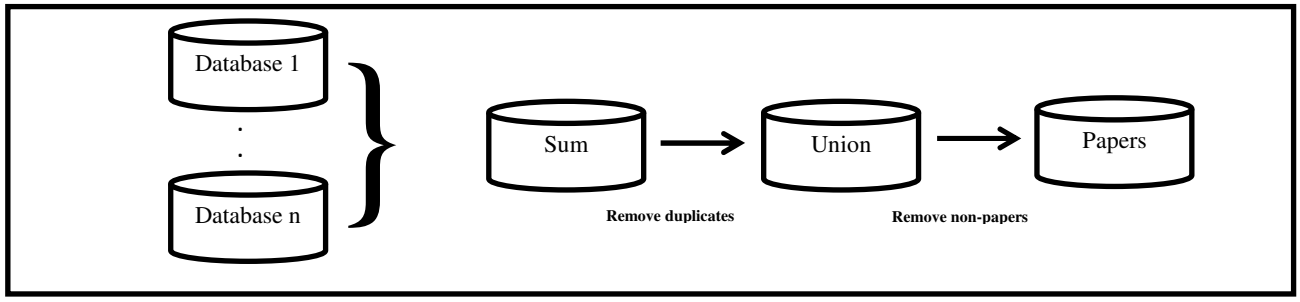

Figure 2: Process of Primary Selection Stage

4.1 Strategy to Detect Duplicates

After exporting and aggregating the results of the searches conducted during the Primary Selection, the removal of duplicated references is necessary. The first strategy to remove duplicates is to sort the references by their titles. However, we had exported references with definite and indefinite articles omitted at the beginning of the titles, some symbols removed, such as colons and dashes, or even extensions of papers that had the original title changed. Therefore, this simple strategy is not efficient.

To help in this process, we proposed an algorithm to detect duplicates based on the Jaccard Similarity Coefficient (Jaccard) (Figure 3). The algorithm receives as input a set of references R and a threshold value, and returns a list of duplicated references L. The algorithm goes through the references in a nested loop, checking the similarity (Jaccard Similarity function) of the current reference with the subsequent. The ObtainTitleAbs function receives a document as parameter and returns its title and abstract concatenated (titleAbsDoc). If the similarity between the references is greater than or equal to the threshold, then the reference and its similarity are inserted in the list L. At the end, the list L with the duplicated references is returned.

Algorithm 1

Require: Set of reference R

Require: $threshold$

Ensure: List of duplicates L

```
1: m ← Length(R)
2: for (i ← 1; i ≤ m; i++) do
3:    titleAbsDoc_i ← ObtainTitleAbs(R[i])
4:    for (j ← i+1; j ≤ m; j++) do
5:       titleAbsDoc_j ← ObtainTitleAbs(R[j])
6:       sim ← JaccardSimilarity (titleAbsDoc_i, titleAbsDoc_j)
7:       if (sim ≥ threshold) then
8:          InsertListDuplicated(R[j], sim, L)
9:       end if
10:   end for
11: end for
12: return L
```

Figure 3: Algorithm to Detect Duplicates

Although Jaccard indicates the degree of similarity returning a number between 0 and 1, we have to define a threshold value to indicate that a reference is duplicated. That is, if a Jaccard value between two references is greater than or equal to the threshold value, one of the references is considered duplicated. From our experiments using one of the SLRs, we obtained a threshold value equals to 0.7 (70%).

The implemented algorithm receives, as input, a Bibtex file (a set of references R) and returns a CSV (Comma Separated Values) file with each reference and its duplicates along with their degree of similarity (list of duplicates L). Figure 4 shows an example of the output of the Strategy to Detect Duplicates. This example depicts a group of references indicated as duplicates, i.e., the algorithm read the reference "Dynamic coupling **measures** for object-oriented software" from the input and indicated the reference "Dynamic coupling **measurement** for object-oriented software" as a duplication with 0.89 of similarity between them.

Title	Similarity
Dynamic coupling measures for object-oriented software	**-**
Dynamic coupling measurement for object-oriented software	**0.89**

Figure 4: Example of Output of the Strategy to Detect Duplicates

4.2 Strategies to Rank References for Full-Text Read

Aiming to reorder the references according to their relevance regarding the search string, we proposed two ranking strategies. In both strategies, we used the Vector Model and the weights of the terms in documents were calculated using only the information present in the input document, i.e., global information was not considered, such as, the IDF that requires data from a collection of documents to be calculated. The weight of a term in a document is defined by its frequency in the document. Therefore, the more times the terms of the query appear in a document, more relevant that document is to the query. To obtain this relevance, the weight $w_{i,j}$ of the term i in document j (8) is calculated, being $f_{i,j}$ the frequency of the term i in document j, and max $f_{l,j}$ the frequency of the term l, where l is the most frequent term in document j. In both strategies, we considered title, abstract, and keywords of the references.

$$w_{i,j} = \frac{f_{i,j}}{max\, f_{l,j}} \qquad (8)$$

4.2.1 Strategy 1

In Strategy 1, we used the traditional form of the Vector Model that considers the partial contribution of each query term. The query terms contained in the search string are treated in the same way, regardless whether they are synonyms or not, ignoring duplicated terms and the connectors AND and OR. To check how similar a document is to the query, the strategy calculates the degree of similarity using Cosine (4). Figure 5 depicts the algorithm to rank references. It receives, as input, a set of references R and a query Q, and returns a list of ranked references L. The algorithm can be divided in two steps: the construction of an inverted index structure [9] and the calculus of the ranking function.

The first step starts inserting distinct tokens from the references into an inverted index structure (Lines 1-6). That structure is composed of key-value pairs. In each pair, the key is a distinct token and the value is a list of occurrences containing, for each document in which the token occurs, its frequency of occurrence and the identification of the document. The construction of this structure is performed by the InsertInvertedIndex function. The Tokenize function splits a text into tokens. Before being tokenized, each string is preprocessed, removing punctuation marks, symbols (such as (), [], {}), and stopwords (articles, prepositions, and conjunctions), and converting letters to lowercase.

In the second step (Lines 7-17), the TokenOcurrIndex function verifies the occurrence of each token of the query in the inverted index, and the CalcWeight function calculates the weight of each token of the query in the document. Next, the similarity of the query with the document is calculated (Line 15). The similarity is obtained by the sum of the weights divided by normDoc. NormDoc is a method for document length normalization, in which the document is

represented by a weight vector of terms, the denominator of (4). After that, the document and the similarity are inserted into a list that is sorted in descending order and returned.

Algorithm 2

Require: Set of reference R

Require: Query Q

Ensure: List of ranked reference L

```
1: for each reference r_j ∈ R do
2:    R_0 ← Tokenize(r_j)
3:    for each token tk ∈ R_0 do
4:       InsertInvertedIndex(tk, j, freq)
5:    end for
6: end for
7: Q_0 ← Tokenize(Q)
8: for each reference r_j ∈ R do
9:    sum ← 0
10:   for each token tk ∈ Q_0 do
11:      if TokenOccurIndex(tk) then
12:         sum ← sum + CalcWeight(tk, j)
13:      end if
14:   end for
15:   sim ← sum/normDoc
16:   InsertDocRankList(j, sim, L)
17: end for
18: Order(L)
19: return L
```

Figure 5: Algorithm of the Strategy 1 to Rank References

4.2.2 Strategy 2

The idea in the Strategy 2 is to define a ranking function that simulates the Boolean expression of the search string. The search terms were organized in groups that aggregates a term and its synonyms using the OR connector. The groups are then connected by the AND connector. To verify the similarity of a document with regard to the query, we calculate the similarity (9), considering g_i as the i^{th} group of terms, sim(g_i) the maximum similarity obtained in the group (10), k_p as the term in the position p between n terms of the group i, and sim(d, k_p) the similarity obtained by the cosine similarity (4).

$$sim(d,q) = \sum_{i=1}^{m} sim(g_i) \qquad (9)$$

$$sim(g_i) = max\left(sim(d,k_p)\right), 1 \le p \le n \qquad (10)$$

The algorithm, depicted by Figure 6, shows the steps of the Strategy 2 to rank the references. It receives as input a set of references R and a query Q, and returns a list of ranked references L. This algorithm is similar to the algorithm of the Strategy 1 regarding to its division in 2 steps, the use of the inverted index structure, and the tokenization process. The first step is the same of the Strategy 1: tokenizing each reference and inserting it into the inverted index.

Their differences are in the second step because Strategy 1 considers the partial contribution of each query term (Lines 7-17) whilst the Strategy 2 verifies the tokens of each group of terms present in the query (Lines 7-23). That is, the ObtainGroupQuery function returns the groups of the tokens presented in query file. The TokenOcurrIndex function verifies the occurrence for each token of the group in the inverted index, created in the step 1. CalcWeight function calculates the weight of each token of the query in the document. In sequence, the InsertWeightTk function inserts the weight and the token in W's structure. Then, the maximum weight of the group (the group's similarity) is calculated by the ObtainWeightMax function. Next, the similarity of the query with the document is calculated (Line 21). The similarity is obtained by the sum of the weights divided by normDoc. NormDoc is a method for document length normalization, in which the document is represented by a weight vector of terms, the denominator of (4). After that, the document and the similarity are inserted into a list that is sorted in descending order and returned.

Algorithm 3

Require: Set of reference R

Require: Query Q

Ensure: List of ranked reference L

1: **for each** reference $r_j \in R$ **do**
2: $R_0 \leftarrow$ Tokenize(r_j)
3: **for each** token $tk \in R_0$ **do**
4: InsertInvertedIndex(tk, j, $freq$)
5: **end for**
6: **end for**
7: $G_0 \leftarrow$ ObtainGroupQuery(Q)
8: **for each** reference $r_j \in R$ **do**
9: $sum \leftarrow 0$
10: **for each** group $g_i \in G_0$ **do**
11: $tkG_0 \leftarrow$ Tokenize(g_i)
12: **for each** token $tk \in tkG_0$ **do**
13: **if** TokenOccurIndex(tk) **then**
14: $weight \leftarrow$ CalcWeight(tk, j)
15: InsertWeightTk($tk, weight, W$)
16: **end if**
17: **end for**
18: $weightMaxGroup \leftarrow$ ObtainWeightMax(W)
19: $sum \leftarrow sum + weightMaxGroup$
20: **end for**
21: $sim \leftarrow sum/normDoc$
22: InsertDocRankList(j, sim, L)
23: **end for**
24: Order(L)
25: return L

Figure 6: Algorithm of the Strategy 2 to Rank References

4.3 Strategy to Detect Non-Papers

The strategy to detect non-papers is based on the Strategy 1. That is, this strategy uses Vector Model to rank the references according to their relevance considering a search string. However, in this strategy, the search string is

composed by terms related to non-papers, such as: Table of Contents and Title Page. The difference is in the return. Using the Strategy 1 to detect non-papers, the return is a collection of references with non-papers closer to the top of the rank instead of relevant references for full-text read.

5 Evaluation

This section presents the experiments performed to evaluate the proposed strategies. It is divided in two subsections: i) Study Settings: it describes the implemented prototype, the datasets, and the methodological procedures; and ii) Results and Discussion: it shows and discusses the results of each strategy.

5.1 Experimental Evaluation Settings

This subsection presents the implemented prototype, the datasets, and the methodological procedures used to perform the experiments.

5.1.1 Prototype

We implemented the strategies as a prototype, in Java, to evaluate their applicability. We used a computer with a CPU Intel Core i5-3570 3.40GHz Quad-Core and 8GB of RAM memory, Java Development Kit 1.7, and NetBeans 8.0.1. The implemented prototype does not have a graphical interface and we defined its inputs in the source code.

The prototype receives as input: one text file with a collection of references in Bibtex format (Figure 7); one text file containing terms present in non-papers (Figure 8a); and one text file containing terms present in the search string (Figure 8b). As output, it returns the ranked references with non-papers closer to the top, references detected as duplicates with the degree of similarity (value between 0 and 1), ranked references, and recall and precision of the non-papers and of the ranking strategies.

Figure **7** depicts an example of a reference for a paper (@ARTICLE) in Bibtex format. Among the attributes of this reference, we have author, title, abstract, and keywords. The strategies use the last three attributes: title, abstract, and keywords. The files with the terms of non-papers (Figure 8a) and of the search string (Figure 8b) are very similar. Each line has a group of terms. A term is a single word or an expression, which is a set of words indicated between quotation marks. Regarding the search string file, each line has a term that will be connected by a logical operator OR. The groups will be connected by a logical operator AND.

```
@ARTICLE{Alderson1998,
  author = {Alderson, A. and Hull, M.E.C. and Jackson, K. and Griffiths, L.E.},
  title = {Method engineering for industrial real-time and embedded systems},
  journal = {Information and Software Technology},
  year = {1998},
  volume = {40},
  pages = {443--454},
  number = {8},
  month = aug,
  abstract = {Real-time and embedded systems have proved troublesome to produce,
    with all the difficulties of the other kinds of software-based systems
    together with a number of specific additional problems. This paper
    reports on a systematic approach to engineering methods and processes,
    including technical management, for the complete lifecycle of real-time
    and embedded systems in an industrial context. An information framework
    was developed as the base for defining appropriate methods and processes
    supported by tools. Existing methods and tools were utilised as the
    development base. MetaCASE technology enabled iterative enhancement
    under the guidance of case studies.},
  doi = {10.1016/S0950-5849(98)00073-1},
  issn = {0950-5849},
  keywords = {Real-time and embedded systems, Development method, System development,
    MetaCASE, CASE},
  owner = {elsevier},
  timestamp = {2012.05.03},
  url = {http://www.sciencedirect.com/science/article/pii/S0950584998000731}
}
```

Figure 7: Example of Reference in Bibtex Format

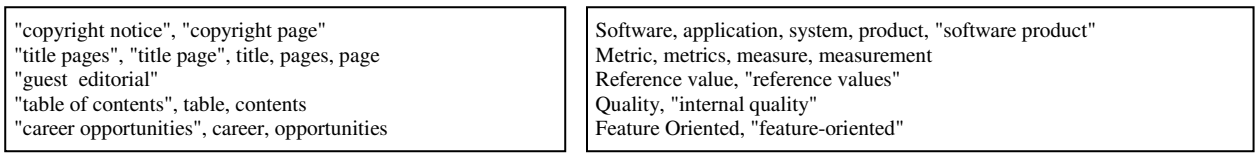

Figure 8: Example of Text File with (a) Terms of Non-Papers and (b) Terms of Search String

5.1.2 Datasets

We used two datasets from two SLRs published in 2012 (SLR-1) [10] and 2014 (SLR-2) [11]. Each dataset is composed by a Bibtex file with the results exported by the search engines, a Bibtex file without references for non-papers, a Bibtex file without duplicated references, a text file with the terms of the search string used in the SLR, and a text file with the titles of the selected studies for full-text read.

Table 1 and Table 2 present the terms used in the search strings in the SLR-1 and SLR-2, respectively. Each line of these tables represents a group of terms (term and synonyms) from the text file that has the search string. We can notice that SLR-1 search string has 11 groups with 33 terms (Table 1), and SLR-2 has 2 groups with 21 terms (Table 2).

Table 1: Terms used in SLR-1 Search String

#	Groups of terms
1	software, application, system, program, product, “software product”, “software system”
2	metric, metrics, measure, measurement
3	reference value, “reference values”
4	quality, “internal quality”
5	feature Oriented, “feature-oriented”
6	aspect Oriented, “aspect-oriented”
7	product line, “product family”, “product-family”, “family based”, “family-based”
8	modularization, modularity, modularisation
9	maintenance, maintainability
10	concern, aspect
11	crosscutting, “cross-cutting”

Table 2: Terms used in SLR-2 Search String

#	Groups of terms
1	"Bad Smell", "Code Smell", "code anomaly", "variability anomaly", "variability smell"
2	"Software Product Line", "software product lines", "software product-line", "software product-lines", "software product family", "software product families", "software product-family", "software product-families", "software family based", "software family-based", "software variability", "software mass customization", "software mass customization production line", "software mass customization production lines", "software-intensive system", "software-intensive systems"

Using those terms (Table 1), the authors of the SLR-1 elaborated the following search string. They used it in the IEEEXplore, Scopus, and Science Direct search engines and obtained 672 references (Table 3).

```
(software OR application OR system OR program OR product OR software product” OR “software system”) AND (metric OR metrics OR measure OR measurement) AND (quality OR “internal quality”) AND (modularization OR modularisation OR modularity) AND (maintenance OR maintainability) AND ((“feature oriented” OR “feature-oriented” OR “product line” OR “product family” OR “family based” OR “product-family” OR “family-based”) OR (“reference value” OR “reference values”) AND (“aspect oriented” OR “aspect-oriented”) AND (concern OR aspect) AND (crosscutting OR “cross-cutting”)))
```

The authors of the SLR-2 elaborated the following search string using the terms presented in Table 2 and used it in the IEEEXplore, Scopus, Elsevier Science Direct, El Compendex, Web of Science, ACM Digital Library, DBLP Computer Science Bibliography, and Computer Science Bibliographies. They obtained 165 references (Table 3).

```
("bad smell" OR "code smell" OR "code anomaly" OR "variability anomaly" OR "variability smell") AND ("software product line" OR "software product-line" OR "software product family" OR "software product-family" OR "software family based" OR "software family-based" OR "software variability" OR "software mass customization" OR "software mass customization production lines" OR "software-intensive system")
```

Table 3 summarizes the number of references in SLR-1 and SLR-2. For example, in SLR-1, 672 references were exported by the search engines. In this collection of references, 20 non-papers and 16 duplicates were detected. At the end, only 13 references were selected for full-text read (Table 4).

Table 3: Datasets used

SLR	# of references	# of non-papers	# of duplicates	# of selected studies
SLR-1	**672**	**20**	**16**	**13**
SLR-2	**165**	**7**	**29**	**44**

Table 4: References selected as relevant in SLR-1

#	References
1	M. S. Ali, M. Ali Babar, L. Chen, and K-J Stol. "A Systematic Review of Comparative Evidence of Aspect-Oriented Programming". Journal of Information and Software Technology, v. 52, n. 9, pp. 871-887, 2010.
2	P. Arpaia, M. L. Bernardi, G. Di Lucca, V. Inglese, and G. Spiezia. "An Aspect-Oriented Programming-Based Approach to Software Development for Fault Detection in Measurement Systems". Journal of Computer Standards & Interfaces, vol. 32, n. 3, pp. 141-152, 2010.
3	E. K. Piveta, A. Moreira, M. S. Pimenta, J. Araújo, P. Guerreiro, R. T. Price. "An Empirical Study of Aspect-Oriented Metrics". Science of Computer Programming, 2012.
4	E. Figueiredo, C. Sant'Anna, A. Garcia, and C. Lucena. "Applying and Evaluating Concern - Sensitive Design Heuristics". Journal of Systems and Software, vol. 85, n. 2, pp. 227-243, 2012.
5	R. Brcina, M. Riebisch. "Architecting for evolvability by means of traceability and features". In: 23rd IEEE/ACM International Conference on Automated Software Engineering, pp. 72-81, 2008.
6	A. L. Medeiros, E. Figueiredo, I. Galvão, A. Garcia, T. Batista, and C. Sant'Anna. "Concern-Based Assessment of Architectural Stability: A Comparative Study". In: 4th Brazilian Symposium on Software Components, Architectures and Reuse, pp. 27-29, 2010.
7	R. Burrows, A. Garcia, F. Taiani. "Coupling Metrics for Aspect-Oriented Programming: A Systematic Review of Maintainability Studies". In: 4th International Conference on Evaluation of Novel Approaches to Software Engineering, 2009.
8	M. Couto, M. Valente, and E. Figueiredo. "Extracting Software Product Lines: A Case Study Using Conditional Compilation". In: European Conference on Software Maintenance and Reengineering, pp. 1-4, 2011.
9	J. M. Conejero, J. Hernández, E. Jurado, K. van den Berg. "Mining early aspects based on syntactical and dependency analyses". Science of Computer Programming, vol.75, n.11, pp. 1113-1141, 2010.
10	F. d'Amorim and P. Borba. "Modularity Analysis of Use Case Implementations". Journal of System and Software, vol. 85, n. 4. pp. 1012-1027, 2012.
11	E. Figueiredo, C. Sant'Anna, A. Garcia, T. T. Bartolomei, W. Cazzola, and A. Marchetto. "On the Maintainability of Aspect-Oriented Software: A Concern-Oriented Measurement Framework". In: 12th European Conference on Software Maintenance and Reengineering, pp. 183-192, 2008.
12	J. M. Conejero, E. Figueiredo, A. Garcia, J. Hernández, E. Jurado. "On the relationship of concern metrics and requirements maintainability". Information and Software Technology, vol. 54, n. 2, pp. 212-238, 2012.
13	M. Revelle, M. Gethers, and D. Poshyvanyk. "Using Structural and Textual Information to Capture Feature Coupling in Object-Oriented Software". Journal of Empirical Software Engineering, vol. 16, n. 6, pp. 773-811, 2011.

5.1.3 Methodological Procedures

The methodological procedures were: i) we received three Bibtex files from the authors of the SLRs containing all references exported by the search engines, references without duplicates, one file without references for non-papers; ii) we edited a text file with terms of the search string used in each SLR and a text file with the titles of the selected studies for full-text read; and iii) tested and analyzed the strategies.

We used all references of SLR-1 to test the algorithm to detect duplicates and define the threshold value. After that, we used the algorithm to detect duplicates in the SLR-2 references. In both detections, we manually analyzed the indications of duplicates and the Jaccard values. In addition, we compared the references indicated by the strategy as duplicates with the duplicated references removed manually by the authors of the SLRs.

To identify common terms in titles of references for non-papers, we performed a search on IEEEXplore, Science Direct and Scopus, and, from the results, built a search string. After that, we used the algorithm with the search string to rank the references of SLR-1 and SLR-2. We (i) analyzed the results by checking recall and precision, and (ii) verified the positions of the references for non-papers in the ranking.

We used each file with only references for papers to perform the ranking by using each strategy. After that, we (i) analyzed the results by checking recall and precision, and (ii) verified the positions of the references in the ranking and the amount of terms of the string found in each reference. In the ranking of the references of SLR-1, we identified 4 references without abstract and 81 references without keywords, while in SLR-2, we identified 1 reference without abstract and 23 references without keywords. This occurred when the results were exported by search engines. We did not index the references without abstracts because they would also be excluded in manual analysis.

5.2 Results

5.2.1 Strategy to Detect Duplicates

The complete base of SLR-1 has 672 references, from which 636 were chosen manually in the Primary Selection stage. Thereby, 36 references were removed because they were considered duplicates or non-papers. Comparing the files, we concluded that, from the 36 removed references, 16 were considered duplicates and 20 were considered non-papers.

We executed the duplicates detection algorithm on the Complete Base, and, as result, the algorithm indicated 16 groups of duplicates with a total of 35 references. From the 16 groups, 13 had 2 references each, and the other 3 had 3 references each. Summarizing, the algorithm indicated 19 duplicates, 3 duplicates more than the manual process, and all 16 duplicates manually detected were also indicated by the algorithm.

The algorithm detected 3 references with the title “Modularity analysis of use case implementations” (Table 5). Taking the first reference and comparing the second and the third to it, the second reference has 100% of similarity and the third has 76% of similarity to the first. Analyzing the title and the abstracts of the first and of the third references, we noticed that the abstract of the third reference has the differences highlighted in Figure 9. For example, the abstract of the first reference begins with “A” and the abstract of the second begins with “Component”.

Table 5: Duplicated References that have the same Title

#	Title	Similarity (%)
1	**Modularity analysis of use case implementations**	-
2	**Modularity analysis of use case implementations**	100
3	**Modularity analysis of use case implementations**	76

Reference #1	Reference #3
A component-based decomposition can result in implementations having use cases code tangled with other concerns and scattered across components.	**Component-based decomposition can result in implementations with use cases code tangled and scattered across components.**

Figure 9: Differences between Abstracts of References that have the same Titles

All references manually removed were indicated by the algorithm with 100% similarity between the duplicates. Table 6 presents the similarity among the references for non-papers indicated by the algorithm. However, these references were manually removed because they were considered non-papers, not duplicates.

Table 6: Duplicated References for Non-papers

#	Title	Similarity (%)
1	**IEEE Recommended Practice for Powering and Grounding Electronic Equipment**	**-**
2	**IEEE Recommended Practice for Powering and Grounding Electronic Equipment - Redline**	**75**
3	**IEEE Recommended Practice for Powering and Grounding Electronic Equipment - Redline**	**78**

Table 7 shows references that have different titles indicated as duplicates. The title of the third reference is different from the title of the first and second references: "Towards A Process-Oriented Software Architecture Reconstruction Taxonomy" and "Software Architecture Reconstruction: A Process-Oriented Taxonomy". The manual process detected the similarity between the first and the second references, but the third reference was not detected. Analyzing why the algorithm indicated that similarity, we notice that first and second references may be an extension of the third because they have 19 pages and were published in a journal, in 2009, by two of the authors of the third reference, while, the third reference has 12 pages and was published in a conference in 2007. The manual process missed this similarity due to differences in the titles.

Table 7: Duplicated References that have Different Titles

#	Title	Similarity (%)
1	**Software Architecture Reconstruction: A Process-Oriented Taxonomy**	**-**
2	**Software Architecture Reconstruction: A Process-Oriented Taxonomy**	**100**
3	**Towards A Process-Oriented Software Architecture Reconstruction Taxonomy**	**79**

5.2.2 Strategy to Detect Non-Papers

To perform the detection of non-papers, we had to perform a search to find common terms that appears in the title of the references for non-papers. We built the following search string with the terms that we found. For example, the terms "IEEE Standard" and "IEEE Std" commonly appeared in the title of the IEEE Standards, and the terms "International" and "Proceedings" commonly appeared in the titles of conference, symposium, and workshop proceedings.

```
("ieee standard", "ieee std", "title pages", "title page", "title", "pages", "page",
"guest editorial", "booklet", "career opportunities", "career", "opportunities",
"draft", "international", "chapter", "annotated", "bibliography", "table of contents",
"table", "contents", "index", "conclusion", "schedule", "tutorial", "papers",
"presented", "journal", "abstracts", "proceedings", "conference", "symposium",
"workshop", "annual index", "proceedings of the", "book of abstracts", "book",
"copyright notice", "copyright page", "copyright", "notice", "front matter", "front
cover", "frontmatter", "frontcover", "front", "cover", "art", "detailed program")
```

As a result in ranking the 632 references of the SLR-1, the algorithm retrieved 40 references because they had, at least, one term of the search string. From these 40 ranked references, 17 were non-papers and they were positioned as follows: 1 to 10, 13, 26, 27, 29, 32, 33 and 34. As a result in ranking the 165 references of the SLR-2, the algorithm retrieved 9 references. From these 9 references, 6 were non-papers and they were positioned on the first 6 positions of the rank.

The 3 following non-papers were not ranked in the SLR-1: "Systems and software engineering – Vocabulary", "An assessment of systems and software engineering scholars and institutions (2003–2007 and 2004–2008)" and "Guide to the Software Engineering Body of Knowledge 2004 Version", and, in the SLR-2, only 1 non-paper was not ranked: "21st century processes for acquiring 21st century software intensive systems of systems". Those 4 references were not ranked because they do not have any term of the search string.

Table 8 shows the precision for the interpolated values in the 11 standard recall levels of the strategy to rank non-papers. For example, when 50% of the references are retrieved (recall), the precision is 100%. Our strategy (described in Section 4.3) retrieved 85% of the non-papers of SLR-1 and SLR-2.

Table 8: Non-Papers - Recall versus Precision

Recall (%)	Precision (%)	
	SLR-1	SLR-2
0	100.0	100.0
10	100.0	100.0
20	100.0	100.0
30	100.0	100.0
40	100.0	100.0
50	100.0	100.0
60	48.4	85.7
70	48.4	0
80	48.4	0
90	0	0
100	0	0

5.2.3 Strategies to Rank References

Strategy 1 and Strategy 2 retrieved 600 references ranking the 632 references of the SLR-1. Table 10 presents the position of each selected reference of the SLR-1. For example, in the Strategy 1, the reference 12 is in position 1, and, in the Strategy 2, this reference is in the position 2. We also notice that Strategy 2 positioned the 13 references closer to top of the rank.

Table 9: Position of References of SLR-1 - Strategy 1 and Strategy 2

Strategy 1		Strategy 2	
Position	Reference	Position	Reference
1	12	1	3
2	3	2	12
6	11	5	2
11	4	8	4
16	10	12	11
22	2	13	10
27	9	15	9
28	7	16	7
51	13	61	6
56	6	64	13
64	8	67	8
106	1	93	5
113	5	97	1

Figure 10 graphically presents the variation of the references positions regarding the applied strategies. Analyzing Figure 10, we can notice different positions for the same reference in the both strategies. For example, Strategy ranked the reference 5 in the position 113 whilst the Strategy 2 ranked it in the position 93. That is, reference 5 was better positioned by Strategy 2 because its position is closer to the top of the rank (position 1).

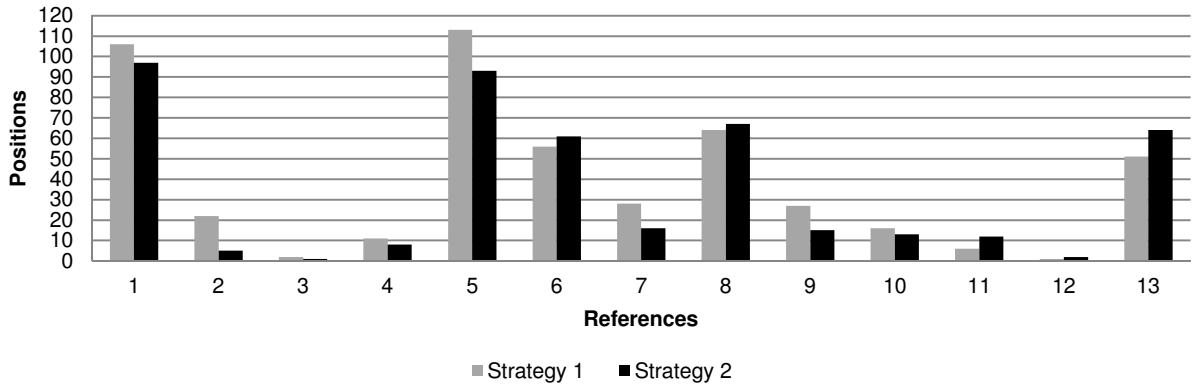

Figure 10: Variation of Positions of References of SLR-1 – Strategy 1 and Strategy 2

Strategy 1 and Strategy 2 retrieved 126 references from the 165 references of the SLR-2. Table 10 and Figure 11 present the variation of the SLR-2 references position. For example, in the first position of the Strategy 1 and the Strategy 2, we have the references 57 and 52, respectively.

Table 10: Position of References of SLR-2 - Strategy 1 and Strategy 2

Strategy 1		Strategy 2	
Position	**Reference**	**Position**	**Reference**
1	57	1	52
3	63	3	114
5	80	5	125
6	81	6	65
7	68	7	41
9	114	9	91
11	89	14	98
12	44	18	96
13	43	20	103
14	123	21	42
16	87	22	104
17	98	23	81
18	96	26	46
19	104	27	44
22	125	28	7
23	65	30	57
24	42	31	30
25	41	32	45
26	62	34	20
29	103	38	6
31	91	39	85
34	25	42	25

Strategy 1		Strategy 2	
Position	**Reference**	**Position**	**Reference**
35	46	43	123
36	52	46	59
39	117	47	89
41	102	50	43
44	45	51	53
45	93	52	93
46	30	54	86
47	85	56	80
49	53	59	68
50	20	60	102
53	86	64	117
57	7	65	111
59	6	69	107
63	111	71	63
64	37	72	87
69	116	74	116
70	59	76	64
79	107	83	62
85	64	87	28
93	70	92	37
97	28	93	70
124	1	124	1

Observing Figure 11, we can notice that the references 1, 70, and 96 were ranked in the same position by both strategies. Conversely, the reference 52 was moved from the position 52 (Strategy 1) to the position 1 (Strategy 2), and the reference 63 was moved from the position 3 (Strategy 1) to the position 71 (Strategy 2).

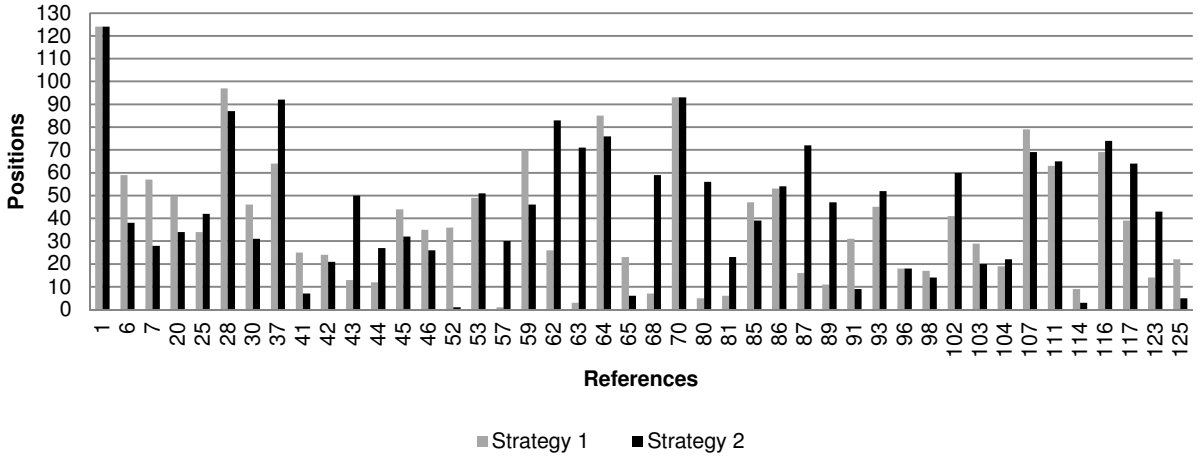

Figure 11: Variation of Positions of the References of the SLR-2 – Strategy 1 and Strategy 2

We manually counted the terms of the references positioned in the first and in the last position by both strategies applied in both SLRs. Table 11 and Table 12 present the counting of terms of the SLR-1, and Table 13 and Table 14 present the counting of terms of the SLR-2. For example, in SLR-1/Strategy 1 (Table 11), the term *software* appears 18 times in the reference 12 (position 1), and 9 times in the reference 5 (position 113).

In the counting of terms regarding the Strategy 2 (Table 12 and Table 14), the references were grouped and the higher occurrence in each group is highlighted. For example, in SLR-1/Strategy 2 (Table 12), we have 11 groups, and, in the group 1, the term *software* appears 5 times in the reference 52 (position 1) and 3 times in the reference 1 (position 124).

Table 11: Count of Terms of the SLR-1 - Strategy 1

#	Terms	Reference / Position	
		12 / 1	5 / 113
1	software	18	9
2	application	0	0
3	system	1	0
4	program	0	0
5	product	5	0
6	software product	2	0
7	software system	0	0
8	metric	0	0
9	metrics	5	2
10	measure	0	0
11	measurement	0	0
12	reference	0	0
13	value	0	0
14	values	0	0
15	reference value	0	0
16	reference values	0	0
17	quality	5	3
18	internal	0	0
19	internal quality	0	0
20	feature	0	2
21	oriented	0	0
22	feature oriented	0	0
23	feature-oriented	0	0
24	aspect	0	0
25	aspect oriented	0	0
26	aspect-oriented	0	0
27	line	2	0
28	family	0	0
29	product line	2	0
30	product family	0	0
31	product-family	0	0
32	family based	0	0
33	family-based	0	0
34	based	0	1
35	modularization	0	0
36	modularity	2	0
37	modularisation	0	0
38	maintenance	3	0
39	maintainability	11	0
40	concern	7	0
41	crosscutting	11	0
42	cross-cutting	0	0
43	cross	0	0
44	cutting	0	0
	Total	74	17

Table 12: Count of Terms of the SLR-1 - Strategy 2

Group	Terms	Reference / Position			
		3 / 1	Higher Occurrence	1 / 97	Higher Occurrence
1	software	5	5	3	3
	application	0		0	
	system	0		1	
	program	0		0	
	product	0		0	
	software product	0		0	
	software system	0		0	
2	metric	0	7	0	0
	metrics	7		0	
	measure	0		0	
	measurement	0		0	
3	reference	0	1	0	0
	value	1		0	
	values	1		0	
	reference value	0		0	
	reference values	0		0	
4	quality	0	0	1	1
	internal	0		0	
	internal quality	0		0	
5	feature	0	6	0	4
	oriented	6		4	
	feature oriented	0		0	
	feature-oriented	0		0	
6	aspect	7	7	3	4
	oriented	6		4	
	aspect oriented	0		0	
	aspect-oriented	6		3	
7	product	0	0	0	3
	line	0		0	
	family	0		0	
	product line	0		0	
	product family	0		0	
	product-family	0		0	
	family based	0		0	
	family-based	0		0	
	based	0		3	
8	modularization	0	1	1	2
	modularity	0		2	
	modularisation	1		0	
9	maintenance	0	0	0	0
	maintainability	0		0	
10	concern	0	7	0	3
	aspect	7		3	
11	crosscutting	2	2	0	0
	cross-cutting	0		0	
	cross	0		0	
	cutting	0		0	
	Total	49		28	

Table 13: Count of Terms of the SLR-2 - Strategy 1

#	Terms	Reference / Position	
		57 / 1	1 / 124
1	Bad Smell	0	0
2	Code Smell	0	0
3	code anomaly	0	0
4	variability anomaly	0	0
5	variability smell	0	0
6	Bad	0	0
7	Smell	0	0
8	Code	5	1
9	anomaly	0	0
10	variability	4	0
11	Software Product Line	0	0
12	software product lines	1	0
13	software product-line	0	0
14	software product-lines	0	0
15	software product family	0	0
16	software product families	0	0
17	software product-family	0	0
18	software product-families	0	0
19	software family based	0	0
20	software family-based	0	0
21	software variability	0	0
22	software mass customization	0	0
23	software mass customization production line	0	0
24	software mass customization production lines	0	0
25	software-intensive system	0	0
26	software-intensive systems	0	0
27	software	4	0
28	product	6	0
29	line	5	0
30	lines	1	0
31	Product-line	0	0
32	Product-lines	0	0
33	family	0	0
34	families	0	0
35	based	2	0
36	Family-based	0	0
37	mass	0	0
38	customization	0	0
39	production	0	0
40	Software-intensive	0	0
41	intensive	0	0
42	system	0	0
43	systems	2	0
	Total	30	1

Table 14: Count of Terms of the SLR-2 - Strategy 2

Group	Terms	Reference / Position			
		52 / 1	Higher Occurrence	1 / 124	Higher Occurrence
1	Bad Smell	0	12	0	1
	Code Smell	0		0	
	code anomaly	0		0	
	variability anomaly	0		0	
	variability smell	0		0	
	Bad	0		0	
	Smell	0		0	
	Code	2		1	
	anomaly	0		0	
	variability	12		0	
2	Software Product Line	0	12	0	0
	software product lines	0		0	
	software product-line	0		0	
	software product-lines	0		0	
	software product family	0		0	
	software product families	0		0	
	software product-family	0		0	
	software product-families	0		0	

Table 14: Count of Terms of the SLR-2 - Strategy 2 (cont.)

Group	Terms	Reference / Position			
		52 / 1	Higher Occurrence	1 / 124	Higher Occurrence
2	software family based	0		0	
	software family-based	0		0	
	software variability	1		0	
	software mass customization	0		0	
	software mass customization production line	0		0	
	software mass customization production lines	0		0	
	software-intensive system	0		0	
	software-intensive systems	0		0	
	software	3		0	
	product	0		0	
	line	0		0	
	lines	0		0	
	Product-line	0		0	
	Product-lines	0		0	
	family	0		0	
	families	0		0	
	based	0		0	
	Family-based	0		0	
	variability	12		0	
	mass	0		0	
	customization	0		0	
	production	0		0	
	Software-intensive	0		0	
	intensive	0		0	
	system	1		0	
	systems	1		0	
	Total	32		1	

Table 15 and Figure 12 show the recall and precision of the strategies to rank references. For example, in SLR-1/Strategy 2, when 80% of the selected references are retrieved (recall), the precision is 50%. We can note that Strategy 2 was equal to or better than Strategy 1, i.e., it ranked the selected references closer to the top of the rank. When applied in the SLR-2, Strategy 2 was better than Strategy 1 in the first two recall levels obtaining 100% of precision. However, Strategy 1 obtained higher precision in the other recall levels.

Table 15: Ranking References - Recall versus Precision

Recall (%)	Precision (%)			
	SLR-1		SLR-2	
	Strategy 1	Strategy 2	Strategy 1	Strategy 2
0	100.0	100.0	73.7	100.0
10	100.0	100.0	73.7	100.0
20	50.0	100.0	73.7	66.7
30	36.4	60.0	73.7	66.7
40	28.6	50.0	72.0	66.7
50	28.6	50.0	64.7	53.5
60	28.6	50.0	63.3	53.5
70	17.9	50.0	63.3	53.2
80	17.2	50.0	57.1	53.2
90	11.5	14.8	50.6	52.0
100	11.5	13.4	35.5	35.5

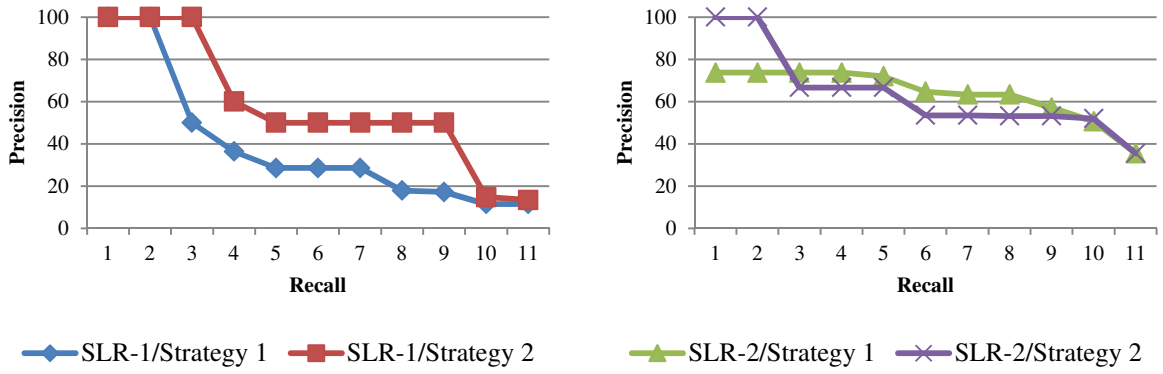

Figure 12: Recall versus Precision SLR-1 and SLR-2 – Strategy 1 and Strategy 2

6 Discussion

In our experimental evaluations, the results of the strategy to detect duplicated references show that it can minimize the effort to detect duplicates because it considers similarities in the title or in the abstract of references. To verify the similarity, we combined the title and the first 100 characters of the abstract. This was sufficient to detect all duplicates that were also manually removed in SLRs and to detect extensions of papers.

The use of more characters of the abstract increases the algorithm execution time and the results are not better. The threshold value (70%) and the use of the 100 first characters of the abstract may be not optimal values because they were defined using only one SLR. Despite this, we used them to detect duplicates in SLR-2 and the algorithm detected all 29 references manually detected plus 2 duplicated references that have different titles. Hence, the strategy to detect duplicated references can be used before the detection of non-papers minimizing the effort to detect them.

In the strategy to detect non-papers, we used only the title because some references for non-papers do not have abstract or keywords. We tested the use of the abstract for ranking and more non-papers were retrieved. However, the recall and the precision were worse due to the large number of references that were also retrieved. Despite the good results, the search string has to be improved with more or differently terms aiming to retrieve more non-papers, and, at the same time, avoiding the retrieve of more references for papers.

Regarding the strategies to rank references, with the ranking generated by Strategy 1, researchers could find the relevant references in the first 113 (17.7%) ranked references of the SLR-1, and, in the first 97 (75.2%) ranked references of the SLR-2. Strategy 2 ranked the selected references of SLR-1 in the first 97 (15.3%) positions, while it ranked the references of SLR-2 in the first 92 (71.3%) positions.

When we analyzed the number of common terms between ranked references and the search strings, we noticed that, in the SLR-1, for example: i) the reference in the position 1 of Strategy 1 has a relatively small number of terms (30%), however it stayed in the first position because it has a total of 74 occurrences of the terms; ii) no more than 73% of the groups have the search string terms; iii) 5% of the references do not have any terms of search string; and iv) 13% of the references exported by the search engines do not have keyword field.

The relatively low amount of terms between the search string and the references indicates that 70% of them may have interfered negatively in a search result, because irrelevant references may be present in search results. Therefore, we can use the proposed strategies to verify if relevant references are closer to the top of the rank in a pilot search. If not, we can choose better terms and improve the results. For example, the reference in position 31 (Strategy 2/SLR-2) has the term “software anomalies” in its title and this term is not present in the search string. If it was present, perhaps that reference would be better ranked.

The existence of references that do not have terms of the string in the title, abstract or keywords can happen due to how the engines perform the search. The documentation of Scopus and Science Direct engines indicates that, when informing an expression in quotes, they do similarity search, i.e., the engines return the same result for the terms "heart-attack" and "heart attack", for example. However, the use of quotation marks in the IEEExplore search engine indicates an exact search, although return the same result for the terms "solid-state" and "solid state".

These search engines use the technique of stemming, that is, a search for the term “metrics” generates results with the term “metric”, inclusive. The IEEExplore performs stemming even among British and American English, but its documentation indicates that some technical terms may not be in stemming dictionary. In IEEExplore, this technique is only applied if the search is not exact, i.e., if the term is not in quotes.

Another feature available on these search engines is the ability to specify which fields the researchers want to perform the search. In IEEExplore, there is an option called “metadata” that includes, for example: abstract, index terms, document title, publication title, and authors. In the SLRs used, the authors performed the searches in IEEEXplore using this option, which may have generated 5% of references that do not have the string terms in title, abstract, and keywords.

In the Primary selection stage, after the execution of the search string, the result must be exported to be analyzed but some search engines do not provide appropriate mechanisms to this aim. For example, in the ACM Digital Library, we can export the references of the search results, but we have to export each reference individually. In IEEExplore, Scopus, and Science Direct, we can export sets of references, but in IEEExplore, the results can be exported in blocks with a maximum of 100 references (if the results have 1000 references, the export must be performed in 10 steps). On the other hand, in the Science Direct and Scopus engines, we can export all references of the search result in a single step if the result has no more than 1000 references.

Additionally to those drawbacks, some references are exported with incomplete data, e.g., abstract and keywords. We executed the SLR-1 search string in the ACM Digital Library and selected a reference. The reference data (e.g., title, authors, abstract and keywords) were presented in the screen, but when we exported the reference, the abstract was not exported. The absence of any information can affect the ranking because fewer terms are indexed, which can result in a lower ranking.

Analyzing the strategies evaluation, we noticed an unexpected result in the Strategy 2 results. We observed, in SLR-1, that only eight groups have, at least, one term in common with the references. However, we expected, at least, one term from each group (11 groups) due to the search string organization. To verify if the Strategy 2 was wrong, we executed the following SLR-1 search string part:

```
(software OR application OR system OR program OR product OR "software product" OR
"software system") AND (metric OR metrics OR measure OR measurement) AND (quality OR
"internal quality") AND (modularization OR modularisation OR modularity)AND(maintenance
OR maintainability) AND...
```

Observing the search string above, we can note 5 groups (terms connected by the OR operator) connected by the AND operator, i.e., the result should have at least one term from each group. By analyzing the term count, we noticed that references 1 and 3 (Strategy 2) do not meet this restriction. Both references were exported from Science Direct. Therefore, we simulated a search on Science Direct using only this piece of string and limiting the search by: i) fields: title, abstract and keywords; ii) area: Computer Science; and iii) period: 1990 to 2012. That search engine sorts the results by relevance and, in the first position, we had reference 12, as well as in Strategy 1. Performing the count of terms per group (Strategy 2), we found that the first three references do not follow the restriction of the search string. This indicates a fault in Boolean searching or the usage of a strategy not mentioned in the search engine documentation, because if the groups were connected with AND, at least one term from each group should exist in the selected references returned by the search engine. This fault may generate results that do not meet the search string increasing the effort to export the search results and to select relevant references.

The selection of relevant references typically involves "random" reading, since the result may not be sorted by relevance when the results of different search engines are combined in only one repository. A subsequent ranking of the

references using the same terms as the search string or other terms may reduce the effort in Primary Selection, because the reading will be guided by the ranking, in which the most relevant search results will be at the top.

One issue that must be considered is the subjectivity present in the texts, because the context may indicate the relevance of the study to a research question, not only the presence of one or more terms and their synonyms. For example, in the abstract of a reference from SLR-1, it is mentioned that: i) the aspect-oriented (AOP) promises to enhance programming software modularization and providing better separation of concerns; ii) an SLR was performed to find empirical studies that presented comparisons between AOP and non-POA approaches; and iii) the studies analyzed showed positive or negative effects related to performance, code size, modularity and evolution. Therefore, the context indicates the use of measures related to AOP and their use to evaluate software quality, because empirical studies were surveyed.

For example, in SLR-1, reference 1 was selected by subjectivity and was retrieved in Strategy 1 because it has 5 terms of the search string with a total of 17 occurrences. The reference 5 was also selected by subjectivity and occupies position 97 in Strategy 2 with 28 terms of 7 groups. Regarding the SLR-2, reference 1 was positioned in the last position by both strategies. Reading its abstract, we noticed the presence of the term “feature-oriented programming”, which is associated to Software Product Lines. However, it lacks other terms related to the search string indicating the selection by the subjectivity.

We noticed that the subjectivity affected the Recall and Precision values (Figure 12) because references of this type, in general, has a low number of terms, and, therefore, they are not closer to the top of the rank. The low presence of the search string terms in the reference 52 (first position - Table 14) of the SLR-2 indicates that the majority of references was selected by subjectivity reflecting the low precision.

7 Conclusion

A Systematic Literature Review is one of the main techniques for synthesizing relevant work related to a particular topic or issue. In the Primary Selection stage, title, abstract, and keywords of each reference returned in the initial search should be read to identify relevant studies to answer research questions. The Primary Selection stage demands effort of researchers due to increased amount of published work. Aiming to reduce this effort, in this paper, we proposed the use of Information Retrieval techniques to detect duplicated references, non-papers, and to rank references according to their relevance regarding a search string.

The strategy to detect duplicates is based on Jaccard Similarity Coefficient and the strategies to rank the references are based on Vector Model. We used the Vector Model in a strategy to position non-papers at the top of the rank and in another two strategies to rank relevant references at the top of the rank. These two strategies are: i) Strategy 1: uses the traditional Vector Model; and ii) Strategy 2: simulates the Boolean expression of the search string.

The evaluation of these strategies was conducted using two real datasets. The results of the strategy to detect duplicates indicate that it can be used in the place of manual detection, because it detected all duplicates that were manually detected and, in addition, it detected references for extensions of studies. The results of the strategy for non-papers show that the strategy can help detecting non-papers, despite it missed some papers due to the search string limitation. Both strategies to rank relevant references can be used to reduce the manual effort, because, in the results, relevant documents were positioned, for example, between 15% to 20% of the references of the top ranking. The strategies do not indicate a percentage of references to be read in advance, but the ranking of references enables reading more adherent to the elaborated search string.

Some limitations of this study include the need to (i) evaluate more datasets to further confirm the results, and (ii) consider the subjectivity of the texts analyzed. Still, the study proved to be promising since the goal is to use the proposed strategies as an aid in the selection process and not to replace human labor. We implemented the strategies as a prototype only to evaluate their applicability and it does not have a graphical interface. In spite of that, the proposed strategies are simple to implement and showed relevant results.

As future work, we intend to implement the strategies in a Web tool and improve the strategies to detect non-papers and rank relevant references. Finally, it is suggested to evaluate the use of the strategies to analyze the full-text of papers aiming to improve the ranking.

Acknowledgements

This work was partially supported by Capes, CNPq, and Fapemig.

References

[1] T. Dyba, B. A. Kitchenham, M. Jorgensen. "Evidence-based Software Engineering for Practioners". *IEEE Software*, vol.22, n.1, pp.58-65. 2005

[2] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman. "Systematic Literature Reviews in Software Engineering - A Systematic Literature Review". *Information and Software Technology*, vol.51, n.1, pp. 7-15, 2009.

[3] B. Kitchenham. "Guidelines for Performing Systematic Literature Review in Software Engineering". EBSE Technical Report, 2.3, Keele University, 2007.

[4] H. Ramampiaro, D. Cruzes, R. Conradi, M. Mendona. "Supporting Evidence-based Software Engineering with Collaborative Information Retrieval". In: International Conference on Collaborative Computing: Networking, Applications and Worksharing, pp.1-5, 2010.

[5] F. Tomassetti, G. Rizzo, A. Vetro, L. Ardito, M. Torchiano, M. Morisio. "Linked Data Approach for Selection Process Automation in Systematic Reviews". In: 15th Annual Conference on Evaluation and Assessment in Software Engineering, pp. 31-35, 2011.

[6] K. R. Felizardo, G. F. Andery, F. V. Paulovich, R. Minghim, J. C. Maldonado. "A Visual Analysis Approach to Validate the Selection Review of Primary Studies in Systematic Reviews". *Information and Software Technology*, vol.54, n.10, pp. 1079-1091, 2012.

[7] B. C. Wallace, T. A. Trikalinos, J. Lau, C. E. Brodley and C. H. Schimid. "Semi-automated Screening of Biomedical Citations for Systematic Reviews". *BMC Bioinformatics*, vol.11, n.1, pp. 55-67, 2010.

[8] D. Martinez, S. Karimi, L. Cavedon, T. Baldwin. "Facilitating Biomedical Systematic Reviews Using Ranked Text Retrieval and Classification". In: 13th Australasian Document Computing Symposium, pp. 53-60, 2008.

[9] R. Baeza-Yates, B. Ribeiro-Neto. *Modern Information Retrieval*. 2nd edition, 2011.

[10] R. Abilio, P. Teles, H. Costa, E. Figueiredo. "A Systematic Review of Contemporary Metrics for Software Maintainability". In: 6th Brazilian Symposium on Software Components Architectures and Reuse (SBCARS), pp.130-139, 2012.

[11] G. Vale, E. Figueiredo, R. Abilio, H. Costa. "Bad Smells in Software Product Lines: A Systematic Review". In: 7th Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS), pp. 84-93, 2014.

[12] R. Abilio; Oliveira, C.; Vale, G.; Morais, F.; Pereira, D.; Costa, H. "Systematic Literature Review Supported by Information Retrieval Techniques: A case study". In: XL Latin American Computing Conference (CLEI), pp. 1-11, 2014.

[13] G. Travassos, J. Biolchini. "Revisões Sistemáticas Aplicadas a Engenharia de Software". In: 21th Brazilian Symposium on Software Engineering (SBES), 2007.

[14] P. Jaccard. "Etude comparative de la distribution florale dans une portion des Alpes et des Jura". *Bulletin del la Societe Vaudoise des Sciences Naturelles*, v. 37, pp. 547-579, 1901.