Linking E-Mails and Source Code Using BM25F

Raffaele Branda, Anna Tolve, Licio Mazzeo, and Giuseppe Scanniello
University of Basilicata, Potenza, ITALY
Email: raf.bran@gmail.com, anna.tolve@tiscali.it, licio.mazzeo@gmail.com, giuseppe.scanniello@unibas.it

***Abstract*—Existing approaches to recover links between e-mails and software artifacts are based on text search or text retrieval and reformulate link recovery as a document retrieval problem. We refine and improve such solutions by leveraging the parts of which an e-mail is composed of: header, current message, and previous messages. The relevance of these parts is weighted by a probabilistic approach based on text retrieval. We implemented our novel solution exploiting the BM25F model. The results of an empirical study conducted on a public benchmark indicate that the new approach in many cases outperforms the baseline approaches chosen. In addition, the proposed approach is easy to use and it is accurate enough to be worth the costs it may introduce in the corpus preprocessing and indexing.**

Keywords - ***Empirical Study; Probabilistic Approach; Traceability Recovery***

I. INTRODUCTION

Maintenance operations are carried out for several reasons and are typically classified as corrective, perfective, and adaptive [1]. Whatever is the maintenance operation, the greater part of the cost and effort is due to the comprehension of source code [2]. Pfleeger and Atlee [3] estimated that up to 60% of software maintenance is spent on the comprehension of source code. There are several reasons that make source code comprehension even more costly and complex and range from the size of the subject software to its overall quality. Other reasons are related to the knowledge of a subject system that is implicitly expressed in software artifacts (i.e., models, documentation, source code, e-mails, and so on) [4]. This knowledge is very difficult to retrieve and it is very often enclosed in non-source artifacts [5].

Among non-source artifacts, those composed of free-form natural language (e.g., documentation, wikis, forums, e-mails) are intended to be read by stakeholders with different experience and knowledge (e.g., managers, developers, testers, and end-users). This kind of artifacts often implicitly or explicitly references to other forms of artifacts, such as source code [6]. Linking e-mails and source code could improve software comprehension and could help to understand the justification behind decisions taken during the design and development [7]. Then, links between e-mails and source code are worthwhile within the entire software lifecycle and in software maintenance, in particular (e.g., [4], [8]).

Several approaches have been proposed to recover links among software artifacts (e.g., [9], [10], [11]). Only a couple of them are concerned with e-mails [12], [13] and can be classified as: rule-based and Information Retrieval (IR) based.

Rule-based. To detect latent links between emails and source code entities hand-code specific rules (i.e., sets of regular expressions) have to be specified. These rules are in turn triggered whenever they match with a portion of email text (e.g., [6]). For example, if the identifiers in the source code repository follows the CamelCase naming convention, we basically know that each identifier is either a single or a compound name (i.e., a sequence of unseparated single names). In the case of class names, all the single names start with a capital letter. Therefore, we can define a regular expression so that every time we find a string in an e-mail of the form `Foo`, `FooBar`, `FooBarXYZ`, etc., we can mark it as a link between the source code and the e-mail. This kind of approach is computationally lightweight for small/medium corpora (e.g., repositories with a small number of e-mails) and easy to implement. Conversely, they lack of flexibility since they are strictly programming-language-dependent. Even more, they do not provide any ranking score associated with the discovered link (i.e., information about a link is binary: a link is either present or not).

IR-based. These approaches reformulate the problem as a particular instance of the more general document retrieval problem. They use IR techniques to compare a set of source artifacts (software entities) with a set of target artifacts (e-mails). Each source code entity (e.g., the class name) is used as the query to retrieve the set of most relevant e-mails. Candidate links are then devised by inspecting the ranked list of retrieved e-mails. Relevance between any pair of source and target artifacts (i.e., source code entity and email) can be determined by their textual/lexical similarity, which is computed by using a specific IR model in conjunction with a particular term-weighting score (e.g., cosine similarity using *tf-idf* vector space model) [14]. The main advantage of IR-based approaches is that they are more flexible and associate each discovered link with a ranking score.

In this paper, we propose an IR-Based approach that refines and improves existing solutions by leveraging the parts of which an e-mail is composed of, namely the header, the current message (from here on, body), and the sentences from previous messages (quote). The relevance of these parts has been weighted by means of a text retrieval probabilistic model. In particular, we implemented our novel solution exploiting the BM25F model [15], [16]. This model is based on a term weighting scheme which takes into account the fact that semi-structured documents from a corpus can be composed of fields [17]. These fields differently contribute to the representation of documents and then to the accuracy of the links retrieved. To assess the validity of our proposal, we have conducted an empirical study on the public benchmark proposed by Bacchelli *et al.* [12].

Structure of the paper. We illustrate our approach in Section II. In Section III, we present the design of the empirical evaluation, while we discuss the achieved results in Section IV. Final remarks conclude the paper.

 ISBN: 978-1-61208-304-9

II. The Approach

IR-based traceability recovery approaches reformulate traceability recovery as a document retrieval problem. We refine and improve such solutions by leveraging header, body, and quote of e-mails. We describe the steps of our approach in the following subsections.

A. Creating a Corpus

Each e-mail results in one document in the corpus. Each document has three well defined fields: header, body, and quote. The header field contains the subject of an e-mail, while the body the sentences of the current message. All the sentences from previous messages are within the quote field. In particular, it includes a chain of messages (e.g., ideas, opinions, issues, or possible solutions) exchanged among stakeholders (mostly developers) linked in the sequence in which they espoused that discussion. We also consider the quote because IR approaches produce better results when a huge amount of lexical information is available [14]. Moreover, the body and the quote fields are separately considered since the lexical information within the body is on the current focus of a discussion, while the quote field includes text that might provide useful information on the entire discussion thread.

B. Corpus Normalization

The corpus is normalized: *(i)* deleting non-textual tokens (i.e., operators, special symbols, numbers, etc.), *(ii)* splitting terms composed of two or more words (e.g., `first_name` and `firstName` are both turned into `first` and `name`), and *(iii)* eliminating all the terms within a stop word list (including the keywords of the programming languages: Java, C, ActionScript, and PHP) and with a length less than three characters. We applied these normalization rules because they have been widely applied in IR-based traceability recovery approaches (e.g., [11]).

Splitting identifiers could produce some noises in the corpus. For example, if the name of a class is `FileBuffer`, it is possible that a software engineer talks about FileBuffer in an e-mail rather than `File` and `Buffer`. However, if the identifiers are not split the things could go from bad to worse: the class name is not in the text of the e-mails (e.g., [12] and [13]), while that name is used as the query. To deal with this issue, we apply the same normalization process on both the corpus and the queries and use the "AND" operator to formulate each query.

Differently from the greater part of the traceability recovery approaches (e.g., [9], [11]), we did not apply any stemming technique [14] to reduce words to their root forms (e.g., the words `designing` and `designer` have `design` as the common radix). This is because we experimentally observed that the use of a Porter stemmer [18] led to worse results. Also, in [12] the stemming was not used for similar reasons.

C. Corpus Indexing

We adopt here a probabilistic IR-based model, namely BM25F [15]. This model extends BM25 [16] to handle semistructured documents from a corpus. The BM25 model was originally devised to pay attention to term frequency and document length, while not introducing a huge number of parameters to set [19]. BM25 showed very good performances [16] and then widely used specially in web document retrieval applications [17], [20]. BM25F was successively proposed to build a term weighting scheme considering the fact that documents from a corpus can be composed of fields (e.g., [17]). Each document is in the corpus and contains information on the contained fields. Then, the fields of a document differently contribute to the document representations. We used BM25F because it has been successfully used on very large corpuses [20] in terms of both scalability and quality of retrieved documents [21]. The use of other probabilistic models could lead to different results. This point is subject of future work.

In both "vector space" and "probabilistic" IR methods, an information retrieval scheme is built for considering each document as a point in a multi-dimensional geometrical space. Therefore, BM25F is based on the bag-of-words model, where each document in the corpus is considered as a collection of words disregarding all information about their order, morphology, or syntactic structure. A word could appear in different fields of the same document. In this case, that word is differently considered according to the field in which it appears. Applying BM25F, each e-mail in the corpus is represented by an array of real numbers, where each element is associated to an item in a dictionary of terms. BM25F does not use a predefined vocabulary or grammar, so it can be easily applied to any kind of corpora.

BM25F works on the occurrence of each term in the fields of all the documents in the corpus. These occurrences are used to build a *term-by-document* matrix. In the current instantiation of this step we modified the original definition of BM25F to better handle the problem at hand. In the model, a generic entry of the table is computed as follows:

$$idf(t,d) = \log\left(\frac{N - df(t) + 0.5}{df(t) + 0.5} + 1\right) * weight(t,d) \quad (1)$$

where N is the total number of documents in the corpus, while df is the number of documents where the term t appears. The weight of the term t with respect to the document d is computed by $weight(t,d)$ as follows:

$$weight(t,d) = \sum_{c \ in \ d} \frac{occurs^{d}_{t,c} * boost_c}{((1 - b_c) + b_c * \frac{l_c}{avl_c})} \quad (2)$$

l_c is the length of the field c in the document d; avl_c is the average length of the field c in all the documents; and b_c is a constant related to the field length; and $boost_c$ is the boost factor applied to the field c. $occurs^{d}_{t,c}$ is the number of terms t that occur in the field c of the document d. This equation is dependent on the field and document relevance and it is similar to a mapping probability. This is because BM25F is considered a probabilistic IR-based model. Regarding the constants of (1), we chose 0.75 as the value for b_c, while 1 is the boost value applied to each field (i.e., header, body, and quote). These values were experimentally chosen and are customary in the IR field [21].

In the original definition of BM25F [20], if a term occurs in over half the documents in the corpus, the model gives a negative weight to the term. This undesirable phenomena is well established in the literature [14]. It is rare in some

 ISBN: 978-1-61208-304-9

applicative contexts, while it is common in others as for an example in the recovery of links between e-mails and source code. In such a context, in fact, e-mails quote sentences from previous messages and then the difference among e-mails (in the same discussion thread) is not that great with respect to the terms contained. To deal with this concern, we modified the computation of idf. The adopted solution is that shown in the equation (1), which is based on that suggested in [22]. The main difference with respect to the canonical computation of idf is that 1 is added to the argument of the logarithm.

D. Query Formulation

In the traceability recovery field, source artifacts are used as the query [9]. The number of queries is then equal to the number of source artifacts. In this work, we used source code entities as the source artifacts and applied the following two instantiations for Query Formulation: *(i)* class names and *(ii)* class and package names. We opted here for that solution because we wanted to compare our novel approach with some baselines (included those in [12]) on a public benchmark [13]. In addition, this solution allowed us to automatically formulate "semi-structured" queries directly parsing the source code[1].

The queries are normalized in the same way as the corpus. When the textual query is composed of more than one term (e.g., `ArgoStatusBar`), the boolean operator "AND" is used with the individual terms of that query (`Argo`, `Status`, and `Bar`). This implies that all the individual terms have also to exist anywhere in the text of a document.

E. Ranking Documents

For a probabilistic IR method, the similarity score between a query with a document in the corpus is not computed by the cosine similarity, but by a different formula motivated by the probability theory [14]. In this work, we used a formula based on a non-linear saturation to reduce the effect of term frequency. This means that the term weights do not grow linearly with term frequency but rather are saturated after a few occurrences:

$$score(q,d) = \sum_{t \ in \ q} idf(t) * \frac{weight(t,d)}{k_1 + weight(t,d)} \qquad (3)$$

where q is the textual query and d is a document in the corpus. The values for $idf(t)$ and $weight(t,d)$ are computed as shown in the equations (1) and (2), respectively. The parameter k_1 usually assumes values in the interval $[1.2, 2]$. We used 2 as the value because experiments suggested that it is a reasonable value [14] to maximize retrieval performances.

F. Examining Results

A set of source artifacts is compared with set of target artifacts (even overlapping). Then, all the possible pairs (candidate links) are reported in a ranked list (sorted in descending order). The software engineer investigates the ranked list of candidate links to classify them as actual or false links.

[1]Different kinds of queries can be formulated automatically or not. For example, the source code content could be also used as the query. We experimentally observed that the use of this kind of query leads to worse results with respect to the other two kinds of query we prose here. Therefore, we did not consider this instantiation for the Query Formulation step. This result is in line with that of Bacchelli *et al*. [12].

III. Empirical Evaluation

Based on the above instantiation of our approach, we implemented an Eclipse plug-in, named Linking e-mAils and Source COde (LASCO). This plug-in has been described in a previous tool demo paper [23]. To asses both the approach and the plug-in, we conducted an empirical study (i.e., an experiment). The presentation of that study is based on the guideline suggested in [24].

A. Definition

As suggested in [24], the goal of our study has been defined using the Goal Question Metrics (GQM) template [25]: ***Analyze*** *traceability recovery links between e-mails and source code* ***for the purpose of*** *evaluating the use of BM25F on header, body, and quote* ***with respect to*** *the accuracy of the retrieved links* ***from the point of view of*** *the researcher and the practitioner* ***in the context of*** *open source systems.*

To this end, we then formulated and investigated the following research question: ***Does our proposal outperform baseline approaches based on text search or text retrieval methods?*** We considered in this study the following baselines:

1. BM25F with the "OR" operator: We apply the BM25F model and the "OR" operator in the step Query Formulation. The Corpus Indexing step is executed by considering the e-mails as composed of header, body, and quote. The only difference with respect to our proposal is that the "OR" operator is used against the "AND" operator;
2. BM25F considering body and quote together: We apply the BM25F model and the operators "AND" and "OR". Furthermore, the Corpus Indexing step is performed considered two fields: *(i)* header and *(ii)* body and quote together;
3. Lucene with "AND" and "OR" operators: In the Corpus Indexing step, we use Lucene. It uses a combination of Vector Space Model (VSM) and the Boolean model to determine how relevant a document is to a query. We here apply both the operators "AND" and "OR". Since Lucene is based on VSM, more times a query term appears in a document relative to the number of times the term appears in all the documents in the corpus, the more relevant that document to the query is;
4. VSM: It represents the documents in the corpus as term vectors, whose size is the number of terms present in the vocabulary. Term vectors are aggregated and transposed to form a *term-document matrix*. To take into account the relevance of terms in each document and in all the corpus, many weighting schema are available. In our empirical evaluation, we employed the *tf-idf* (term frequency - inverse document frequency) weighting;
5. LSI: Even for a corpus of modest size, the term-document matrix is likely to have several tens of thousand of rows and columns, and a rank in the tens of thousands as well. LSI is an extension of VSM developed to overcome the synonymy and polysemy problems [26]. SVD (Singular Value Decomposition) is used to construct a low-rank approximation matrix to the term-document matrix [27]. In LSI there is no way to enforce Boolean conditions [14];
6. Lightweight linking technique (LLT) - case sensitive (CS): To reference software entities from e-mails, the names of the software entities are used as text search queries. There exists a link between a software entity and an e-mail, when

 ISBN: 978-1-61208-304-9

there is a case sensitive match on the entity name;
7. LLT - mixed approach (MA): In case the name of software entities are compounded words, they are split (e.g., `ClassName` becomes `Class Name`). The compounded words are then used for the case sensitive match on the entity name, otherwise it is used a regular expression based on class and package name;
8. LLT - MA with regular expression (RE): This approach is based on that above. A different regular expression is used to better handle non-Java systems. Further details about Lightweight linking techniques can be found in [12].

The baselines from 1 to 5 are different instantiations of the recovery process shown in Section II, while the others are lightweight approaches based on regular expressions. In all the IR-based baseline approaches, with the exception of the first and second one, the corpus was indexed considering together header, body, and quote.

B. Planning

1) Context: Many IR-based traceability recovery approaches depend on users' choices: the software engineer analyzes a subset of the ranked list to determine whether each traceability link has been correctly retrieved. It is the software engineer who makes the decision to conclude this process. The lower the number of false traceability links retrieved, the better the approach is. The best case scenario is that all the retrieved links are correct. IR-based traceability recovery methods are far from this desirable behavior [9]. In fact, IR-based traceability recovery approaches might retrieve links between source and target artifacts that do not coincide with correct ones: some are correct and others not. To remove erroneously recovered links from the candidate ones, a subset of top links in the ranked list (i.e., retrieved links) should be presented to the software engineer. This is possible by selecting a threshold to cut the ranked list (e.g., [11], [28]).

There are methods that do not take into account the similarity between source and target artifacts: *Constant Cut Point*, it imposes a threshold on the number of recovered links, and *Variable Cut Point*, it consists in specifying the percentage of the links of the ranked list to be considered correctly retrieved. Alternative possible strategies for threshold selection are based on the similarity between source and target artifacts: *Constant Threshold*, a constant threshold is chosen, *Scale Threshold*, a threshold is computed as the percentage of the best similarity value between two vectors, and *Variable Threshold*, all the links among those candidate are retrieved links whether their similarity values are in a fixed interval. In our experiment, we used the Constant Threshold method. This is the standard method used in the literature [9]. We applied this method employing thresholds assuming values between 0 and 1. The increment used was 0.01.

For each software entity, the Query Formulation step was instantiated using either the original class name or the concatenation of class and package names. Many of the design choices have been taken because our main goal was to compare the results of our solution with those presented in [12].

2) Variable selection: The traceability links retrieved by applying both our approach and the baselines are analyzed in terms of *correctness* and *completeness*. Correctness reflects the fact that an approach is able to retrieve links that are correct. To measure the correctness, we used (as customary) *precision* ($precision = \frac{|TP|}{|TP|+|FP|}$). On the other hand, completeness reflects how much the set of retrieved links is complete with respect to the all actual links. *Recall* is used to measure this aspect ($recall = \frac{|TP|}{|TP|+|FN|}$). where TP (true positives) is the set of links correctly retrieved. The set FN (false negatives) contains the correct links not retrieved, while FP (false positives) the links incorrectly presented as correct.

When the e-mails in the benchmark do not have any reference to source code artifacts, the union of TP and FN is empty (i.e., $|TP|+|FN|=0$). In all these cases, we cannot calculate the values for the recall measure. The values for precision could not be computed in case the approach found no link between an e-mail and the source code. Similar to [12], we avoided these issues calculating the average of $|TP|$, $|FP|$, and $|FN|$, on the entire dataset. We then computed the average values for precision and recall. Precision and recall assume values in the interval $[0,1]$. The higher the precision value, the more correct the approach is. Similarly, the higher the recall value, the better the approach is.

To get a trade-off between correctness and completeness, we applied the balanced F-measure (i.e., $F_1 = \frac{2*precision*recall}{precision+recall}$). F_1 was used to estimate the *accuracy* of the approach. This is the main criterion we considered in the study. This measure has values in the interval $[0,1]$. When comparing two approaches, the one with higher F_1 value is considered the best, namely the most accurate.

3) Instrumentation: To estimate our approach and to compare it with the baselines, we used the benchmark proposed in [13]. For each system and all the threshold values, we computed the values of precision, recall, and F_1. To compare our approach with the baselines, we selected the constant threshold that produced the best accuracy.

IV. Results and Discussion

A. Results

The results achieved by applying our approach are shown in Table I. The table also reports the results achieved by applying the "OR" operator. The results are grouped according to the two different instantiations of the step Query Formulation: *(i)* class name and *(ii)* class and package names. The last row reports the average values for each measure. Better average accuracy was achieved using class and package names and the "AND" operator (F_1 = 0.44). With respect to each individual system, we obtained the higher accuracy for Habari, namely the system implemented in PHP (F_1 = 0.59). On that system, the higher value of correctness was also obtained (precision = 0.77). It is worth mentioning that the results for that system are the same both using class name alone and class and package names together. This is because PHP 5 did not have packages. Namespaces (i.e., packages) where only introduced in PHP 5.3. The same held for Augeas (the C software system).

Table II shows the results achieved by indexing the corpus using: *(i)* header and *(ii)* body and quote together. With respect to accuracy, better results were achieved using the operator "AND" and class and package names. The best average accuracy value was 0.41. Among the analyzed software systems, the best accuracy was obtained for Habari (F_1 = 0.61).

 ISBN: 978-1-61208-304-9

TABLE I. BM25F RESULTS INDEXING THE CORPUS USING: *(i)* HEADER, *(ii)* BODY, AND *(iii)* QUOTE

	Class Name + "AND"			Class Name + "OR"			Class and Package Names + "AND"			Class and Package Names + "OR"		
System	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1
ArgoUML	0.32	0.53	0.40	0.05	0.55	0.10	0.41	0.72	0.52	0.04	0.48	0.07
Freenet	0.23	0.49	0.31	0.03	0.23	0.06	0.30	0.52	0.39	0.02	0.40	0.05
JMeter	0.32	0.41	0.36	0.10	0.41	0.16	0.49	0.62	0.55	0.06	0.43	0.10
Away3D	0.31	0.51	0.39	0.15	0.24	0.18	0.39	0.44	0.41	0.12	0.24	0.16
Habari	0.77	0.48	0.59	0.29	0.35	0.32	0.77	0.48	0.59	0.29	0.35	0.32
Augeas	0.12	0.27	0.16	0.04	0.32	0.08	0.12	0.26	0.16	0.04	0.32	0.08
Average value	0.35	0.45	0.37	0.11	0.35	0.15	0.41	0.51	0.44	0.10	0.37	0.13

TABLE II. BM25F RESULTS INDEXING THE CORPUS USING: *(i)* HEADER AND *(ii)* BODY AND QUOTE TOGETHER

	Class Name + "AND"			Class Name + "OR"			Class and Package Names + "AND"			Class and Package Names + "OR"		
System	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1
ArgoUML	0.34	0.51	0.41	0.07	0.58	0.12	0.40	0.46	0.43	0.05	0.55	0.09
Freenet	0.22	0.54	0.31	0.08	0.45	0.14	0.29	0.62	0.40	0.07	0.5	0.13
JMeter	0.29	0.45	0.36	0.14	0.41	0.21	0.34	0.66	0.45	0.12	0.45	0.19
Away3D	0.29	0.76	0.42	0.21	0.24	0.22	0.37	0.44	0.40	0.16	0.23	0.19
Habari	0.74	0.52	0.61	0.46	0.45	0.46	0.74	0.52	0.61	0.46	0.45	0.46
Augeas	0.11	0.35	0.17	0.10	0.17	0.13	0.11	0.35	0.17	0.06	0.35	0.10
Average value	0.33	0.52	0.38	0.18	0.38	0.21	0.38	0.5	0.41	0.15	0.42	0.19

TABLE III. LUCENE RESULTS

	Class Name + "AND"			Class Name + "OR"			Class and Package Names + "AND"			Class and Package Names+ "OR"		
System	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1
ArgoUML	0.32	0.50	0.39	0.06	0.50	0.11	0.39	0.47	0.43	0.03	0.53	0.06
Freenet	0.20	0.59	0.30	0.07	0.47	0.11	0.27	0.64	0.38	0.05	0.56	0.10
Jmeter	0.27	0.46	0.34	0.10	0.36	0.15	0.34	0.70	0.46	0.07	0.49	0.13
Away3D	0.29	0.77	0.42	0.17	0.22	0.19	0.37	0.44	0.40	0.13	0.24	0.17
Habari	0.61	0.55	0.58	0.45	0.40	0.43	0.61	0.55	0.58	0.45	0.40	0.42
Augeas	0.10	0.27	0.15	0.05	0.21	0.08	0.10	0.27	0.15	0.05	0.20	0.08
Average value	0.30	0.52	0.36	0.15	0.36	0.18	0.35	0.51	0.40	0.13	0.40	0.16

TABLE IV. RESULTS BY BACCHELLI *et al.* [12]

	VSM with $tf-idf$			LSI			LLT - case sensitive			LLT - mixed approach			LLT - mixed approach with RE		
System	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1
ArgoUML	0.25	0.34	0.29	0.60	0.48	0.53	0.27	0.68	0.38	0.64	0.61	0.63	0.35	0.68	0.46
Freenet	0.15	0.25	0.19	0.62	0.43	0.51	0.17	0.70	0.27	0.59	0.59	0.59	0.27	0.69	0.39
JMeter	0.21	0.34	0.26	0.52	0.40	0.45	0.15	0.73	0.25	0.59	0.65	0.62	0.30	0.72	0.42
Away3D	0.35	0.31	0.33	0.35	0.33	0.34	0.32	0.74	0.44	0.40	0.54	0.46	0.41	0.72	0.52
Habari	0.34	0.39	0.36	0.36	0.41	0.38	0.40	0.41	0.41	0.83	0.09	0.17	0.49	0.38	0.43
Augeas	0.10	0.20	0.14	0.10	0.28	0.14	0.09	0.72	0.15	0.14	0.02	0.04	0.15	0.64	0.24
Average value	0.23	0.31	0.26	0.43	0.39	0.39	0.23	0.66	0.32	0.53	0.42	0.42	0.33	0.64	0.41

The results achieved with Lucene are shown in Table III. The best average accuracy value was reached using the operator "AND" and class and package names (i.e., 0.40). The better accuracy was achieved for Habari (F_1 = 0.58).

Table IV summarizes the results presented in [12], instantiating Query Formulation step with class name. As mentioned before, the results for class and package names together are not reported for VSM and LSI because the authors observed that better results were achieved using only class names. Table IV also shows the results for the lightweight linking techniques.

The results indicate that our proposed technique is more accurate than BM25F using two fields (header and body and quote together) on all the Java systems with the exception of Freenet (the F_1 values were 0.39 and 0.40, respectively). On the non-Java systems, the use of BM25F indexing the corpus with three or two fields did not produce remarkable differences in accuracy (see Table I and Table II).

Our approach using class and package names as the queries is more accurate than VSM. Similar results were achieved for Lucene using both the operators and class name and class and package names together as the queries. Indeed, our proposal did not outperform Lucene only on Away3D when using the "AND" operator and class name as the query. The F_1 values were 0.41 and 0.42, respectively.

As far as LSI, our approach is more accurate on all the non-Java system and Jmeter. For ArgoUML the difference in favor of LSI was negligible (the F_1 values was 0.52 with respect to 0.53). A larger difference in accuracy was obtained for Freenet.

Our proposal outperformed LLT-CS in accuracy on all the systems with the exception of Away3D (the F_1 values were 0.44 and 0.41, respectively). BM25F with three fields was on average more accurate than LLT MA with and without RE (see the average values of F_1). With respect to LLT MA, we achieved better F_1 values results on Habari and Augeas (0.59 vs. 0.17 and 0.16 vs. 0.04, respectively). On the Java systems LLT MA was more accurate than our approach. For LLT MA RE, we reached better results on the Java systems and Habari.

Regarding the correctness and completeness of the retrieved links, we can observe an interesting pattern in the data: our approach mostly allowed obtaining a more complete set of retrieved links that are correct. This result is desirable when you are interested in the recovery of links among software artifacts (e.g., [9]).

B. Discussion

1) IR-Based recovery: For Java systems, LSI outperformed other approaches based on IR techniques with respect to the accuracy of the retrieved links. A reason is that each e-mail in the corpus quotes a large number of sentences from previous

messages. This is the best scenario for using LSI [14]. In fact, this technique is used to disclose the latent semantic structure of a corpus and to recognize its topics, so dealing with synonymy and polysemy problems. Further, each document in the corpus has a large size as compared with the entities used as the queries. This might also represent another possible reason for having achieved better results on Java systems. The considerations above and the fact that LSI outperformed our approach in terms of accuracy only on Freenet (this difference was 0.04, while this difference was in favor of our approach on ArgoUML and JMeter and was 0.01 and 0.1, respectively) suggest that BM25F represents an alternative also when dealing with large documents in the corpus.

In the case that e-mails in the corpus quote a small number of sentences from previous e-mails our approach outperformed other baseline approaches based on IR techniques. This happened for all the non-Java systems. For the Habari system, the e-mails were very short and then BM25F made the difference also considering the information in the body and quote together.

For the system implemented in C (i.e., Augeas), the application of the IR-Based approaches mostly produced worse results in terms of correctness, completeness, and accuracy. As also suggested in [12], a possible justification is related to the names of the entities. However, our approach outperformed the IR based baselines. Again, indexing the e-mails considering two or three fields did not produce remarkable differences.

The instantiation of the Query Formulation step with class and package names improved the correctness and completeness when our technique was used. Then, it is possible that the choice of the source artifact can make the difference in the accuracy of the links recovered.

The use of a stemming technique in the Normalization step produced worse results. Then, this technique seems useless in the recovery of links between source code and e-mails, when using BM25F (with two and three fields) and Lucene. On these instances, the use of the "AND" operator led to better results in terms of accuracy and correctness of the retrieved links with respect to the "OR" operator. This result held for all the systems. For completeness, the results achieved with the "AND" operator were mostly better than those achieved with the "OR" operator. Only in four cases the use of the "OR" operator led to better recall values.

The use of source code (program statements and/or source code comment) as the query was also analyzed. The results revealed that this kind of instantiation for the Query Formulation step led to worse results with respect to the other two kinds of queries considered here. This result is in line with that shown in[12] and has the following implication: it is better to use class name and class and package names as the queries.

We also performed an analysis to get indications on whether BM25F might introduce scalability issues. We used a laptop equipped by a processor Intel Core i7-2630QM with 4 GB of RAM and Windows Seven Home Premium SP-1 64bit as operating system. This analysis was performed on each system and the baseline processes implemented for our experiment (see Section III-A). The results indicated that the time to build, normalize, and index the e-mails of the entire benchmark was twice when using three fields (i.e., 5033 milliseconds) with respect to the use of two fields (i.e., 2668 milliseconds). For Lucene, the average execution time on all the systems in the benchmark was 2660 milliseconds. For the Query Formulation step, nearly the same pattern was observed. Further details are not provided for space reason.

2) Lightweight Approaches: Regarding the accuracy of the retrieved links, LLT MA outperformed the other lightweight techniques and our approach on the Java systems. On the non-Java system with the exception of Away 3D, LLT MA did not outperform our approach and the differences in the F_1 values were significant (0.59 vs. 0.17 and 0.16 vs. 0.04, respectively). The difference on Away3D was small (F_1 values were 0.41 and 0.44, respectively). Similarly, LLT MA did not outperform LLT MA RE on the non-Java systems. The achieved results suggest that our approach and LLT MA RE are more independent from the kind of documents in the corpus. Since our approach was more accurate, we can then conclude that it is the best and can be applied without making any assumption on the mailing list and the programming language of the understudy system. The same did not hold for lightweight techniques based on regular expressions because they heavily rely on common conventions and intrinsic syntactical characteristics of the corpus [12].

C. Lesson Learned

The accuracy of our approach increased when e-mails contain a huge amount of text and the entity names are carefully chosen and naming conventions are used. Furthermore, when e-mails did not contain a huge amount of text, the application of BM25F on two or three fields did not produce noteworthy differences. Then, BM25F on header, body, and quote with the operator "AND" is the best alternative.

We experimentally observed that, in terms of accuracy, our approach outperformed on 5 out of 6 systems the lightweight technique that is more independent from the kind of e-mails in the corpus (i.e., LLT MA RE) [12]. To apply our approach, any assumption on the system understudy has to be made and any particular configuration setting is required. Therefore, our approach is easier to use than lightweight approaches and it is accurate enough to be worth the costs it may introduce in the corpus preprocessing and indexing phases. Furthermore, IR-based approaches, such as the one we introduce here, are more scalable. They are more efficient than lightweight techniques when the number of e-mails in the corpus increases. Finally, lightweight techniques return documents without any ranking: an e-mail either matches or not a regular expression. As a consequence, all the retrieved links have to be analyzed. In addition, incremental processes cannot be used to keep only relevant links (e.g., [29]).

1) Pieces of evidences: We distilled our findings and lesson learned into the following pieces of evidence (PoE):
PoE1. Accuracy increases when using class and package names as the queries;
PoE2. Applying our approach on three fields (i.e., header, body, and quote) improves the results when the corpus contains e-mails with a huge amount of text and the entity names are carefully chosen by developers;
PoE3. Using the "AND" operator leads to better results in terms of correctness, completeness, and accuracy;
PoE4. The corpus normalization by using stemming techniques reduces the accuracy of the recovered links;

 ISBN: 978-1-61208-304-9

PoE5. Our approach scales reasonable well also when the number of documents in the corpus increases;
PoE6. Our approach is more independent from the mailing list than lightweight approaches.

D. Threats to Validity

To comprehend the strengths and limitations of our study, we present here the threats that could affect the validity of the results and their generalization. Although our efforts in mitigating as many threats as possible, some threats are unavoidable. A possible threat is related to the used benchmark that is built on human judgement. The use of open source software represents another threat to validity. Although many large companies are using open source software in their own work or as a part of their marketed software, it will be worth replicating the study on real project. These replications will help us to confirm or contradict the achieved results. The instantiation of Query Formulation is another possible threat. We used class names or class and package names to compare our approach with those in [12].

V. Conclusion

We proposed, implemented [23], and evaluated an approach to recover links between e-mails and source code. The approach is based on text retrieval techniques combined with the BM25F probabilistic model. To assess the validity of our proposal, we conducted an empirical evaluation using a public benchmark [13]. Based on this benchmark, we performed a comparison between our approach and 8 baselines. The results indicated that our approach in many cases outperformed the IR-based baseline approaches and the lightweight techniques proposed in [12].

References

[1] E. B. Swanson, "The dimensions of maintenance," in *Proc. of International Conference on Software Engineering*. IEEE CS Press, 1976, pp. 492–497.

[2] A. V. Mayrhauser, "Program comprehension during software maintenance and evolution," *IEEE Computer*, vol. 28, pp. 44–55, 1995.

[3] S. Pfleeger and J. Atlee, *Software Engineering - Theory and Practice*. Pearson, 2006.

[4] A. De Lucia, F. Fasano, C. Grieco, and G. Tortora, "Recovering design rationale from email repositories," in *Proc. of the International Conference on Software Maintenance*. IEEE, 2009, pp. 543–546.

[5] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, "Advancing candidate link generation for requirements tracing: The study of methods," *IEEE Trans. Software Eng.*, vol. 32, no. 1, pp. 4–19, 2006.

[6] A. Bacchelli, M. Lanza, and M. D'Ambros, "Miler: a toolset for exploring email data," in *Proc. of the International Conference on Software Engineering*. ACM, 2011, pp. 1025–1027.

[7] A. Bacchelli, T. Dal Sasso, M. D'Ambros, and M. Lanza, "Content classification of development emails," in *Proceedings of the 2012 International Conference on Software Engineering*. IEEE Press, 2012, pp. 375–385.

[8] N. Bettenburg, B. Adams, A. E. Hassan, and M. Smidt, "A lightweight approach to uncover technical artifacts in unstructured data," *Proc. of the International Conference on Program Comprehension*, vol. 0, pp. 185–188, 2011.

[9] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Recovering traceability links in software artifact management systems using information retrieval methods," *ACM Trans. Softw. Eng. Methodol.*, vol. 16, no. 4, 2007.

[10] S. K. Sundaram, J. H. Hayes, A. Dekhtyar, and E. A. Holbrook, "Assessing traceability of software engineering artifacts," *Requir. Eng.*, vol. 15, no. 3, pp. 313–335, 2010.

[11] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering traceability links between code and documentation," *IEEE Trans. Software Eng.*, vol. 28, no. 10, pp. 970–983, 2002.

[12] A. Bacchelli, M. Lanza, and R. Robbes, "Linking e-mails and source code artifacts," in *Proc. of International Conference on Software Engineering*. ACM, May 2010, pp. 375–384.

[13] A. Bacchelli, M. D'Ambros, M. Lanza, and R. Robbes, "Benchmarking lightweight techniques to link e-mails and source code," in *Proc. of Working Conference on Reverse Engineering*. IEEE Computer Society, 2009, pp. 205–214.

[14] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[15] S. Robertson, H. Zaragoza, and M. Taylor, "Simple bm25 extension to multiple weighted fields," in *Proc. of International Conference on Information and Knowledge Management*. ACM, 2004, pp. 42–49.

[16] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: Bm25 and beyond," *Found. Trends Inf. Retr.*, vol. 3, pp. 333–389, April 2009.

[17] K. Y. Itakura and C. L. Clarke, "A framework for bm25f-based xml retrieval," in *Proc. of International Conference on Research and Development in Information Retrieval*. ACM, 2010, pp. 843–844.

[18] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[19] K. S. Jones, S. Walker, and S. E. Robertson, "A probabilistic model of information retrieval: development and comparative experiments," *Inf. Process. Manage.*, vol. 36, pp. 779–808, November 2000.

[20] J. R. Pérez-Agüera, J. Arroyo, J. Greenberg, J. P. Iglesias, and V. Fresno, "Using bm25f for semantic search," in *Proc. of the International Semantic Search Workshop*. ACM, 2010, pp. 2:1–2:8.

[21] J. Pérez-Iglesias, J. R. Pérez-Agüera, V. Fresno, and Y. Z. Feinstein, "Integrating the Probabilistic Models BM25/BM25F into Lucene," *CoRR*, vol. abs/0911.5046, 2009.

[22] L. Dolamic and J. Savoy, "When stopword lists make the difference," *J. Am. Soc. Inf. Sci. Technol.*, vol. 61, no. 1, pp. 200–203, Jan. 2010.

[23] L. Mazzeo, A. Tolve, R. Branda, and G. Scanniello, "Linking e-mails and source code with lasco," in *Proc. of the European Conference on Software Maintenance and Reengineering*. IEEE Computer Society, 2010.

[24] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering - An Introduction*. Kluwer, 2000.

[25] V. Basili, G. Caldiera, and D. H. Rombach, *The Goal Question Metric Paradigm, Encyclopedia of Software Engineering*. John Wiley and Sons, 1994.

[26] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society of Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[27] J. K. Cullum and R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. 1*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002.

[28] A. Marcus and J. I. Maletic, "Recovering documentation-to-source-code traceability links using latent semantic indexing," in *Proc. of the International Conference on Software Engineering*. IEEE CS Press, 2003, pp. 125–137.

[29] A. De Lucia, R. Oliveto, and P. Sgueglia, "Incremental approach and user feedbacks: a silver bullet for traceability recovery," in *Proc. of the International Conference on Software Maintenance*. IEEE Computer Society, 2006, pp. 299–309.

 ISBN: 978-1-61208-304-9