

# Automatic Labeling of Software Requirements Clusters

Nan Niu\*, Sandeep Reddivari\*, Anas Mahmoud\*, Tanmay Bhowmik\*, and Songhua Xu†

\* Department of Computer Science and Engineering, Mississippi State University, MS 39762, USA

† Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

niu@cse.msstate.edu, {srr159, amm560, tb394}@msstate.edu, xus1@ornl.gov

**Abstract**—Clustering is of great practical value in retrieving reusable requirements artifacts from the ever-growing software project repositories. Despite the development of automated cluster labeling techniques in information retrieval, little is understood about automatic labeling of requirements clusters. In this paper, we review the literature on cluster labeling, and conduct an experiment to evaluate how automated methods perform in labeling requirements clusters. The results show that differential labeling outperforms cluster-internal labeling, and that hybrid method does not necessarily lead to the labels best matching human judgment. Our work sheds light on improving automated ways to support search-driven development.

**Keywords**-clustering; labeling; requirements; software reuse;

## I. INTRODUCTION

Reuse of existing software artifacts and knowledge is often viewed as the best approach to achieve tremendous increases in developers' productivity, improvements of software quality, and reduced development cost [1, 2]. While retrieving reusable code attracts much attention in search-driven development (e.g., [3, 4, 5]), researchers have begun to support software reuse by retrieving high-level artifacts (HLAs), such as requirements [6] and design models [7]. As a matter of fact, source code and HLAs are complementary in that showing the similarity between the source code being developed and related HLAs like requirements helps developers improve the quality of source code identifiers [8]. This in turn increases the likelihood of the right artifacts being reused in the right contexts.

In this paper, we focus on *requirements* retrieval, through which the developer can work on the abstractions closer to the software system's initial concepts [9, 10]. As the number of available requirements-level artifacts grows, it is often difficult to find relevant information to support pragmatic reuse tasks [1]. One way to address the challenge is *clustering* [11] — the automatic division of data into classes/clusters, i.e., a set of requirements that are in some way characterized by an internal coherence and/or an external isolation.

The literature on requirements clustering is emerging. Researchers have applied clustering to support a variety of software engineering activities, including feature identification [12, 13], automated tracing [14], and system modularization [15]. An important task in analyzing clustering results is to *label* the clusters to help summarize the data.

From pragmatically managing complexity's point of view, all the requirements within a cluster can be abstracted and treated as a single unit. This makes labeling particularly important for developers to efficiently search and understand the artifacts to be reused.

In existing approaches, requirements clusters are labeled either manually (e.g., [13]) or semi-automatically (e.g., [14]). Automatic labeling of requirements clusters is still not well understood, despite the continuous development and improvement of automated labeling techniques in related fields like document and source code clustering. It is our conjecture that taking full advantage of automated labeling can improve the scalability and comprehensibility of clustering-based requirements reuse. The goal of our work is to parallel today's search-driven development by investigating the role automated labeling plays in requirements clustering. To that end, this paper makes two contributions:

- a review of the automated cluster-labeling methods (Section II); and
- an experiment that tests three labeling methods over three requirements datasets (Section III).

We summarize the paper and shed light on future research avenues in Section IV.

## II. BACKGROUND AND RELATED WORK

In labeling a cluster of documents (and similarly, a cluster of webpages or a cluster of requirements), automatic methods attempt to characterize the cluster items rather than understand them. For this reason, a list of terms, instead of concise terms, is typically generated as the cluster's descriptors. We distinguish two basic categories of cluster labeling [11]: *cluster-internal labeling* selects labels by relying solely on the contents of the cluster of interest, whereas *differential cluster labeling* labels a cluster by comparing the terms in one cluster with the terms occurring in other clusters. In addition, *hybrid labeling* generates labels by combining both intra-cluster and inter-cluster features. Table I classifies the methods surveyed in this paper.

We describe the main idea of each labeling method and provide known area(s) in which the method is applied. Our

Table I  
CLASSIFICATION OF AUTOMATED LABELING METHODS

Cluster-Internal	Differential			Hybrid		
TF	IDF	$\chi^2$	IG	TF-IDF	FPW	LSI

Table II  
COMPARISON OF AUTOMATED LABELING METHODS

Domain	Study	Role of Labeling	Methods	Comparison Result
Source Code Clustering	[24] [20]	Software architecture recovery Reverse eng. & re-engineering	TF vs. IDF $\chi^2$ vs. IDF	IDF generates more meaningful labels than TF. $\chi^2$ consistently chooses more meaningful labels than IDF.
Information Retrieval & Web Search	[21] [17]	Searching and browsing Interactive browsing	$\chi^2$ vs. FPW TF vs. TF-IDF	FPW produces more meaningful labels than $\chi^2$ . TF-IDF provides more meaningful labels than TF.

surveyed application areas include: information retrieval, Web search, and source code clustering. In the following descriptions, we use “documents” to refer to the data items to be clustered, although strictly speaking, it is the webpages and source code profiles [16] that are being grouped in Web search and software clustering respectively.

**Term Frequency (TF).** The TF weighting scheme assigns the weight to be equal to the number of occurrences of term  $t$  in document  $d$ , denoted by  $tf_{t,d}$ . In particular:

$$w_{t,d} = \begin{cases} 1 + \log(tf_{t,d}) & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The terms with high TF values are candidate descriptors for the cluster. The TF-based labeling has been applied in information retrieval [17].

**Inverse Document Frequency (IDF).** IDF is a measure of the general importance of the term obtained by weighting how concentrated or spread out the term  $t$  appears in the corpus. Specifically:

$$idf_t = \log \frac{N}{1 + df_t} \quad (2)$$

where  $N$  is the total number of documents in a collection, and  $df_t$  represents document frequency whose value is the number of documents in which term  $t$  occurs. The terms with high IDF values can discriminate one cluster from another, and therefore serve as candidate cluster labels. The IDF labeling scheme has been applied in Web search [18] and source code clustering [19, 20].

**Chi-Square Selection ( $\chi^2$ ).** The main idea is to use  $\chi^2$  test for each term at each node in a hierarchy by starting at the root and recursively moving down the hierarchy. If one cannot reject the hypothesis that a term is equally likely to occur in all of the children of a given node, then the term is assigned to the current node’s bag of words and removed from all nodes under the current node. The  $\chi^2$  value is calculated as follows:

$$\chi^2 = \sum \frac{(O - E)^2}{E} \quad (3)$$

where  $O$  is the observed frequency of the term and  $E$  is the expected frequency of the term.  $\chi^2$  labeling has found its applications in information retrieval [21].

**Information Gain (IG).** This is an information-theoretic measure that quantifies the degree of dependence of two random variables. The IG,  $gain_r(w, C)$ , of a word  $w$  in a cluster  $C$  is computed on the basis of entropy and mutual information described in [13]. The detailed IG calculations

can be found in [11]. During differential cluster labeling, one calculates the IG of each term in the cluster and selects the  $k$  terms with the highest IG value. IG-based labeling has been used in Web search [22].

**TF-IDF** combines formulas (1) and (2) (i.e.,  $tf\text{-}idf_{t,d} = tf_{t,d} \times idf_t$ ) in assigning the weight of using term  $t$  to label the cluster. This scheme has been used extensively in information retrieval [17].

**Frequent & Predictive Words (FPW)** computes the value:

$$p(w|C) \times \frac{p(w|C)}{p(w)} \quad (4)$$

by considering two factors: 1) frequency,  $p(w|C)$ ; and 2) predictiveness,  $p(w|C)/p(w)$ . Predictiveness is similar to IDF in that it distributes more weight to the words occurring frequently in a given cluster and less weight to the words occurring frequently in all of the clusters:  $p(w|C)$  is the term frequency in a given cluster and  $p(w)$  is the term frequency in a more general category or in the whole collection. FPW has been considered in information retrieval [21].

**Latent Semantic Indexing (LSI)** was adapted by Kuhn *et al.* in a novel way to label source code clusters [23]. The key idea is to reverse the usual search process where a search query of terms is used to find documents, and instead, one uses the documents in a cluster as search query to find the most similar terms. To label a cluster, one simply takes the top- $n$  most similar terms, where  $n=7$  in the case studies reported in [23].

Even though the set of surveyed labeling methods is by no means exhaustive, it helps to reveal the growing interest in investigating automated ways to characterize and summarize clusters. Also note that the method’s application areas reported here are rather representative than exhaustive.

Despite the increased research on label generation, there has been only scattered work on comparing the effectiveness of different labeling methods. Table II summarizes the four comparative studies from related domains. Although the role of automated labeling varies in these studies, some common aspects exist. These include using the cluster labels prepared manually by human expert(s) as the answer set, and measuring the effectiveness of labeling methods by qualitative comparisons, such as “meaningfulness” evaluated subjectively by the researchers. Nevertheless, some general conclusions that can be drawn from Table II seem to be that: 1) hybrid methods (e.g., FPW) outperform differential methods (e.g.,  $\chi^2$ ), and 2) differential methods in turn outperform

Table III  
CHARACTERISTICS OF THE EXPERIMENTAL DATASETS

Data-set	Input & Preprocessing			Output	
	# of req.s	# of words per req.	# of non-stop words per req.	# of clusters	# of req.s per cluster
iTrust	41	375.2	75.8	5	8.2
eTour	58	103.1	43.8	8	7.3
CM-1	235	22.3	5.2	27	8.7

internal methods (e.g., TF). Such findings make intuitive sense since *i*) the whole (synthesized scheme) should work better than the parts (individual schemes), and *ii*) considering how dissimilar a cluster (abstracted as a single unit) is to other clusters (units) should result in more discriminative (and thus more “meaningful”) labels than paying attention only to the inner parts of the cluster.

### III. EXPERIMENTAL EVALUATION

The previous section provides some understanding of automated cluster labeling methods and their performances. In order to advance this understanding, we carry out an experiment to examine how automated methods perform in labeling *requirements clusters*.

#### A. Datasets and Procedure

Three requirements datasets are used in our experiment. iTrust<sup>1</sup> is a medical application, developed by software engineering students at North Carolina State University (USA), which provides patients with a means to keep up with their medical records and to communicate with their doctors. eTour<sup>2</sup> is an electronic touristic guide developed by the final year students at the University of Salerno (Italy). CM-1<sup>3</sup> contains a complete set of high-level requirements for a NASA scientific instrument. Table III shows the characteristics of these datasets.

The *independent variable* in our experiment is automated cluster labeling. We take three values, namely TF,  $\chi^2$ , and FPW, to instantiate the independent variable for a couple of reasons. First, they represent different method categories (cf. Table I). Second, some pairwise comparisons between them exist (cf. Table II). It is our intention to test whether the results can be generalized to the requirements datasets.

Two *controlled variables* are worth discussing here: preprocessing and the underlying algorithm used to produce requirements clusters. As for preprocessing, we extend our source code indexer [16] to handle requirements. Three specific steps are involved: tokenizing, filtering (i.e., removing stop words like a and the), and stemming (i.e., reducing a word to its inflectional root: “patients” → “patient”).

As far as the clustering algorithm is concerned, the seminal work by Duan and Cleland-Huang [14] compared three algorithms: agglomerative hierarchical, *k*-means, and bisecting divisive, and reported that at reasonable clustering

<sup>1</sup><http://agile.csc.ncsu.edu/iTrust>

<sup>2</sup><http://www.cs.wm.edu/semeru/tefse2011>

<sup>3</sup><http://promise.site.uottawa.ca/SERepository/datasets/cm1.desc>

Table IV  
ILLUSTRATION OF LABELS FOR AN iTRUST’S REQUIREMENTS CLUSTER

TF (Internal)	$\chi^2$ (Differential)	FPW (Hybrid)	Manually Prepared
patient	hospit	record	record
repres	health	health	patient
personal	patient	prescript	HCP*
user	record	lhcp <sup>†</sup>	authenticate
system	medic	personnel	monitor
medic	edit	repres	medical
health	weight	laboratori	disable
weight	report	monitor	health
pedomet	authent	agent	agent
view	repres	identif	hospital

\* HCP: Health Care Professional. <sup>†</sup> lhcp: Licenced HCP.

granularity of 5-6 clusters or higher, no significant difference was observed between the requirements clusters produced by the three algorithms. This is an important finding because there is unlikely to be a clear winner among the many different clustering algorithms when it comes to requirements clustering. For this reason, we choose single linkage, an agglomerative clustering method, for its simplicity and robustness [25]. The two “output” columns of Table III show the clustering results. Note that, following [14], we also take human usability of the clusters into consideration, so the resulting cluster’s size is defined to be  $7 \pm 2$ .

#### B. Results and Analysis

In our experiment, the clusters are generated once and for all, and then only labeling differs. The *dependent variable* is concerned with automatic labeling’s effectiveness. We devise the answer set based on human expert’s manual labeling. In our study, a researcher familiar with the datasets acts as the expert. Although this poses a clear limitation, our focus is on comparing the labels automatically generated by different methods, given that all the other variables are set, including those involved in preprocessing, the underlying clustering algorithm, and preparing the answer set.

Table IV illustrates the labels for one of iTrust’s requirements clusters. From all the candidate labels generated automatically, we select the top-10 ranked terms, a heuristic commonly used in labeling evaluations, e.g., [21, 11]. Similarly, the expert provides 10 labels for each cluster, with which the automatic labels are compared. For our evaluation, *kappa statistic*, an agreement measure of multi-judgments [11], is adopted. Kappa statistic measures the actual observed agreement,  $P(A)$ , excluding the expected agreement,  $P(E)$ , if agreement is due strictly to chance. Mathematically,

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \quad (5)$$

Kappa statistic returns a value in  $[0, 1]$ , where  $\kappa = 0$  shows no agreement and  $\kappa = 1$  suggests complete agreement. Note that the Kappa measure is rather set-based; we are investigating measures that incorporate the label rankings for

Table V  
KAPPA STATISTICS BETWEEN THE AUTOMATIC AND MANUAL LABELS

Data-set	TF vs. Manual		$\chi^2$ vs. Manual		FPW vs. Manual	
	$\kappa$	Agreement	$\kappa$	Agreement	$\kappa$	Agreement
iTrust	.38	fair	.77	substantial	.59	moderate
eTour	.57	moderate	.70	substantial	.73	substantial
CM-1	.36	fair	.83	almost perfect	.55	moderate

improving the evaluation. To interpret the  $\kappa$  statistical significance in the current study, we use the following magnitude guideline [11]: if  $\kappa$  is in  $[0, 0.20]$ ,  $[0.21, 0.40]$ ,  $[0.41, 0.60]$ ,  $[0.61, 0.80]$ , and  $[0.81, 1]$ , then the agreement is slight, fair, moderate, substantial, and almost perfect respectively. Take the labels generated by TF and human expert in Table IV as an example, the  $\kappa$  value is 0.35, which indicates the agreement level is fair between these label sets.

Table V shows the average  $\kappa$  values by comparing the requirements clusters' labels in our datasets. The results confirm that cluster-internal methods, such as TF, produce the least satisfactory sets of labels. However, the best label sets in our experiment result from  $\chi^2$ , a differential cluster labeling method, rather than a hybrid method FPW. The finding may imply that combining poor descriptors is worse than combining nothing at all. Testing this conjecture requires more detailed investigation.

Several factors can affect the validity of our study. As for construct validity [26], for example, the interpretation of label's "meaningfulness" may vary among experts, but considering 10 terms for each cluster in the evaluation helps mitigate the threat. We believe the main strength of our experimental design is its high internal validity [26], as all the factors potentially affecting the label-agreement measure are under our direct control. The results of our study may not generalize to other requirements datasets – a threat to external validity [26]. Besides, different preprocessings or clustering algorithms can lead to different results.

#### IV. CONCLUSIONS

Retrieving and clustering requirements artifacts from project repositories have become more important search-driven development activities due to a shorter cognitive distance involved in software reuse [9]. There currently exists only limited understanding about how to best label the requirements clusters in an automated manner. In this paper, we have reviewed three categories of automated labeling techniques and compared their performances in labeling document, webpage, and source code clusters. We further carried out an experiment to evaluate three labeling methods on the requirements datasets. The results indicated that  $\chi^2$  produced superior labels than the TF and FPW methods.

From our initial implementation and evaluation, we feel that automatic labeling has rich value in supporting reusable requirements retrieval, and search-driven development in general. In the future, we plan to study the interplay of clustering algorithms and labeling methods, investigate semantics-enabled retrieval techniques [27], and to exploit

novel and automated ways [28, 29] to improve automatic labeling.

#### ACKNOWLEDGEMENT

Songhua Xu performed this research as a Eugene P. Wigner Fellow and staff member at the Oak Ridge National Laboratory, managed by UT-Battle, LLC, for the U.S. Department of Energy under Contract DE-AC05-00OR22725.

#### REFERENCES

- [1] R. Holmes and R. J. Walker, "Supporting the investigation and planning of pragmatic reuse tasks," in *ICSE*, 2007, pp. 447–457.
- [2] L. Heinemann and B. Hummel, "Recommending API methods based on identifier contexts," in *SUITE*, 2011, pp. 1–4.
- [3] S. Bhatia, S. Tuarob, P. Mitra, and C. L. Giles, "An algorithm search engine for software developers," in *SUITE*, 2011, pp. 13–16.
- [4] O. Panchenko, H. Plattner, and A. Zeier, "What do developers search for in source code and why," in *SUITE*, 2011, pp. 33–36.
- [5] E. Hill *et al.*, "Investigating how to effectively combine static concern location techniques," in *SUITE*, 2011, pp. 37–40.
- [6] A. Mahmoud and N. Niu, "An experimental investigation of reusable requirements retrieval," in *IRI*, 2010, pp. 330–335.
- [7] B. Bislimovska *et al.*, "Content-based search of model repositories with graph matching techniques," in *SUITE*, 2011, pp. 5–8.
- [8] M. Gethers *et al.*, "CodeTopics: which topic am I coding now?" in *ICSE*, 2011, pp. 1034–1036.
- [9] C. W. Krueger, "Software reuse," *ACM Comput. Surv.*, vol. 28(2), pp. 131–183, 1992.
- [10] J. L. Cybulski *et al.*, "Reuse of early life-cycle artifacts: workproducts, method and tools," *Annals of SE*, vol. 5(1), pp. 227–251, 1998.
- [11] C. D. Manning, P. Raghavan, and H. Schtze, *An Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [12] K. Chen *et al.*, "An approach to constructing feature models based on requirements clustering," in *RE*, 2005, pp. 31–40.
- [13] N. Niu and S. Easterbrook, "On-demand cluster analysis for product line functional requirements," in *SPLC*, 2008, pp. 87–96.
- [14] C. Duan and J. Cleland-Huang, "Clustering support for automated tracing," in *ASE*, 2007, pp. 244–253.
- [15] Z. Li *et al.*, "Does requirements clustering lead to modular design?" in *REFSQ*, 2009, pp. 233–239.
- [16] A. Mahmoud and N. Niu, "Source code indexing for automated tracing," in *TEFSE*, 2011, pp. 3–9.
- [17] P. Treeratpituk and J. Callan, "Automatically labeling hierarchical clusters," in *Int'l Conf. Digital Govern. Research*, 2006, pp. 167–176.
- [18] P. Tonella, F. Ricca, E. Pianta, and C. Girardi, "Using keyword extraction for web site clustering," in *WSE*, 2003, pp. 41–48.
- [19] O. Maqbool and H. A. Babri, "Interpreting clustering results through cluster labeling," in *ICET*, 2005, pp. 429–434.
- [20] F. Siddique and O. Maqbool, "Analyzing term weighting schemes for labeling software clusters," in *CSMR*, 2011, pp. 85–88.
- [21] A. Popescul and L. H. Ungar, "Automatic labeling of document clusters," (unpublished manuscript), 2000.
- [22] F. Geraci, M. Pellegrini, M. Maggini, and F. Sebastiani, "Cluster generation and labeling for web snippets: a fast, accurate hierarchical solution," *Internet Mathematics*, vol. 3(4), pp. 413–443, 2007.
- [23] A. Kuhn, S. Ducasse, and T. Girba, "Semantic clustering: identifying topics in source code," *IST*, vol. 49(3), pp. 230–243, 2007.
- [24] O. Maqbool and H. A. Babri, "Automated software clustering: an insight using cluster labels," *JSS*, vol. 79(11), pp. 1632–1648, 2006.
- [25] —, "Hierarchical clustering for software architecture recovery," *TSE*, vol. 33(11), pp. 759–780, 2007.
- [26] R. K. Yin, *Case Study Research: Design and Methods*. Sage Publications, 2003.
- [27] A. Mahmoud and N. Niu, "Using semantics-enabled information retrieval in requirements tracing," in *COMPSAC*, 2010, pp. 246–247.
- [28] S. Haiduc *et al.*, "On the use of automated text summarization techniques for summarizing source code," in *WCRE*, 2010, pp. 35–44.
- [29] S. Xu *et al.*, "Keyword extraction and headline generation using novel word features," in *AAAI*, 2010, pp. 1461–1466.